ABSTRACT

Title of Dissertation:	A BIG-DATA-DRIVEN FRAMEWORK FOR SPATIOTEMPORAL TRAVEL DEMAND ESTIMATION AND PREDICTION
	Songhua Hu, Doctor of Philosophy, 2023
Dissertation directed by:	Professor, Paul Schonfeld Department of Civil and Environmental Engineering

Traditional travel demand models heavily rely on travel surveys, simplify future demand forecasting, and show low sensitivity in response to spatiotemporal dynamics. This study proposes a deep-learning-driven framework based on mobile device location data (MDLD) for estimating and predicting large-scale travel demand at both individual and aggregated levels. This study first introduces how raw MDLD should be parsed to distill trip rosters and estimate population flow. Based on derived information, this study reexamines relations between average population flow and its determinants such as built environment, socioeconomics, and demographics, via a set of explainable machine learning (EML) models. Different interpretation approaches are employed and compared to understand nonlinear and interactive relations learned by EML models. Next, this study proposes a *Multi-graph Multi-head Adaptive Temporal Graph Convolutional Network* (Multi-ATGCN), a general deep learning framework that fuses multi-view spatial structures, multi-head temporal patterns, and various external effects, for multi-step citywide population flow forecasting. Multi-ATGCN is designed to comprehensively address

challenges such as complex spatial dependency, diverse temporal patterns, and heterogeneous external effects in spatiotemporal population flow forecasting. Last, at an individual level, this study proposes a *Hierarchical Activity-based Framework* (HABF) for simultaneously predicting the activity, departure time, and location of the origin and destination of the next trip, incorporating both internal (individual characteristics) and external (calendar, point-of-interests (POIs)) information. For each individual, HABF first predicts activities via an *Interpretable Hierarchical Transformer* (IHTF). IHTF can efficiently handle big data benefiting from its transformer-based design to avoid recursion. Meanwhile, loss functions used in semantic segmentation are introduced into IHTF to address imbalanced distributions of activity types. Then, a local plus global probabilistic generator is designed to generate locations based on predicted activities and historical places, allowing individuals to visit new or historically-sparse places. Analyses are performed on several real-world datasets to demonstrate the model's capability in forecasting large-scale high-resolution human mobility in a timely and credible manner. Altogether, this study provides sound evidence, practically and theoretically, of the feasibility and reliability of realizing data-driven travel demand estimation and prediction at different spatiotemporal resolutions and scales.

A BIG-DATA-DRIVEN FRAMEWORK FOR SPATIOTEMPORAL TRAVEL DEMAND ESTIMATION AND PREDICTION

by

Songhua Hu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2023

Advisory Committee:

Professor Paul Schonfeld, Chair
Professor Anna Alberini, Dean's Representative
Professor Ali Haghani
Assistant Professor Chenfeng Xiong
Assistant Professor Yiqun Xie

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Dr. Paul Schonfeld, for his unwavering guidance, support, encouragement, and protection throughout the final year of my Ph.D. program. Dr. Schonfeld has been an invaluable mentor, always available to offer expert advice and wise counsel when I needed it most. I am truly grateful for his generosity with his time, patience, and for sharing his vast knowledge and experience with me, which have greatly advanced my research career. It has been an honor and privilege to learn from him and to be his student.

I extend very special thanks to my dissertation committee members, Dr. Anna Alberini, Dr. Ali Haghani, Dr. Chenfeng Xiong, and Dr. Yiqun Xie, for their constructive comments and critical feedback that enhanced the rigor and quality of my dissertation. I am particularly grateful to Dr. Alberini, who not only served as the dean's representative but also provided generous help to support me in continuing my academic career. I would like to especially thank Dr. Haghani for his guidance in the courses and his insightful comments on the proposal, which were instrumental in shaping my dissertation. I am deeply thankful to Dr. Xiong for his academic and personal support during my Ph.D. study. Our discussions and collaborations have been a source of great joy and intellectual growth for me. I also deeply appreciate Dr. Xie’s expertise in deep learning-based spatiotemporal modeling. His interesting courses and constructive suggestions have greatly influenced this dissertation.

I would like to acknowledge the mentorship of my former supervisor, Dr. Lei Zhang, who played a significant role in the early stages of my Ph.D. study. I am grateful for his support and for setting me on the path to an independent researcher.

I would also like to express my appreciation for the support and friendship of my colleagues at the University of Maryland, who have provided encouragement and shared brilliant ideas that have made my academic journey full of laughter and joy.

Lastly, I owe my deepest thanks to my family and my girlfriend, Yingrui Zhao, for their unwavering love, encouragement, and sacrifices. Their constant support and understanding enabled me to pursue my academic goals with confidence and determination. I could not have completed this journey without their support.

Table of Contents

Acknowledgments ii
Table of Contents iv
List of Tables vii
List of Figures viii
List of Abbreviations x
1 Chapter 1: Introduction 1
1.1 Background 1
1.2 Research framework & objectives 4
1.3 Contributions 6
1.4 Organization 8
2 Chapter 2: Literature Review 9
2.1 Traditional travel demand modeling 9
2.1.1 Four-step model 9
2.1.2 Activity-based model 10
2.1.3 Travel demand and underlying factors 12
2.2 Mobile device location data (MDLD) in travel demand modeling 13
2.2.1 Mobile device location data (MDLD) 13
2.2.2 Travel demand estimation using MDLD 15
2.3 Spatiotemporal travel demand modeling 19
2.3.1 Cross-sectional modeling and nonlinearity 20
2.3.2 Explainable machine learning (EML) 21
2.3.3 Temporal modeling (Time-series analysis) 22
2.3.4 Spatial modeling 25
2.3.5 Spatiotemporal modeling 27
2.3.6 Population flow forecasting and Next-location forecasting 28
2.4 Discussion 30
3 Chapter 3: Extracting travel demand from MDLD 33
3.1 Raw data cleaning & Data statistics 33
3.2 Home&Work identification 34
3.3 Trip identification 35
3.4 Mode imputation 37
3.5 Population weighting 38
3.6 Result validation 39
3.7 A real-world application 41
4 Chapter 4: Revisiting travel demand and underlying factors 45
4.1 Variable and data description 46
4.1.1 Prediction target 46
4.1.2 Feature engineering 49
4.2 Explainable machine learning (EML) models 52
4.2.1 Linear regression 53
4.2.2 Single decision tree 54
4.2.3 Tree ensembles: bagging and boosting 56
4.2.4 Advanced boosting trees 57
4.3 Interpretation approaches 59
4.3.1 Feature importance 60

4.3.2 Partial dependence plot (PDP) and Accumulated Local Effect (ALE)..... 62
4.3.3 SHapley Additive exPlanations (SHAP) 63
4.4 Experiment settings..... 65
4.5 Predictive performance 69
4.5.1 Performance across models..... 69
4.5.2 Performance across MDLD sampling rate..... 72
4.6 Feature importance..... 74
4.6.1 Feature importance of tree-based models 74
4.6.2 Comparison with regression coefficients..... 77
4.6.3 Robustness check of feature importance..... 78
4.7 Nonlinear relations..... 81
4.7.1 Global nonlinear relations: PDP and ALE..... 81
4.7.2 Local nonlinear interaction: SHAP 85
4.8 Discussion 87
5 Chapter 5: Population flow time series forecasting 89
5.1 Problem statement..... 92
5.2 Proposed approach: Multi-ATGCN..... 93
5.2.1 Multi-head temporal fusion..... 94
5.2.2 Multi-view adaptive graph learning..... 96
5.2.3 Zone-specific Mix-hop GCN (ZMGCN)..... 99
5.2.4 Graph convolutional recurrent neural network (GCRNN) 102
5.2.5 Multi-step output..... 103
5.3 Experiments 104
5.3.1 Data description 104
5.3.2 Baselines for comparison..... 107
5.3.3 Experiment settings..... 108
5.3.4 Implementation details of baselines..... 109
5.4 Baseline comparison 112
5.5 Model analysis 117
5.5.1 Performance across census tracts..... 117
5.5.2 Effects of the lower bound..... 118
5.5.3 Ablation study..... 120
5.5.4 Complexity analysis..... 121
5.5.5 Parameter study..... 122
5.5.6 Graph learning 123
5.6 Discussion..... 125
6 Chapter 6: Individual trip itinerary forecasting 128
6.1 Trip preprocessing 131
6.1.1 Activity labeling..... 131
6.1.2 Missing trips imputation 133
6.1.3 Short trips linking and clustering..... 135
6.1.4 Devices filtering..... 137
6.1.5 Before/After comparison 137
6.2 Hourly activity chain prediction 140
6.2.1 Objectives and challenges..... 140
6.2.2 Problem statement..... 143

6.2.3 Proposed approach: Interpretable hierarchical transformer (IHTF) 145
6.2.4 Model interpretation methods 155
6.2.5 Losses and metrics for imbalanced classification 156
6.2.6 Experiment settings 161
6.2.7 Model prediction 163
6.2.8 Model interpretation 172
6.3 Location generation 175
6.3.1 Time&Activity-aware chain-based probability 175
6.3.2 OD volume evaluation 178
6.3.3 Trip distance evaluation 179
6.3.4 Trip spatial distribution 182
6.4 Discussion 184
7 Chapter 7: Conclusion 187
7.1 Summary of key findings 187
7.1.1 Extracting travel demand from MDLD 187
7.1.2 Revisiting travel demand and underlying factors 187
7.1.3 Population flow time series forecasting 188
7.1.4 Individual trip itinerary forecasting 190
7.2 Future research directions 191
7.2.1 Augmenting model power in forecasting aggregated demand 191
7.2.2 Enhancing model accuracy in individual trip forecasting 196
7.2.3 Constructing an end-to-end MDLD-based travel demand model 197
References 199

List of Tables

Table 3-1 Quality metrics statistics of national MDLD 34
Table 3-2 A rule-based recursive algorithm to identify trips 36
Table 4-1 Summary of national POI information 50
Table 4-2 Summary of CBG-level target and features for EML models 50
Table 4-3 EML models for cross-sectional population inflow estimation 53
Table 4-4 Summary of interpretation techniques 59
Table 4-5 Hyperparameters tuning and best configurations for EML models 67
Table 4-6 Computational efficiencies of interpretation techniques 68
Table 4-7 EML model performance comparison (vanilla) 70
Table 4-8 EML model performance comparison (fine-tuned) 71
Table 5-1 Training dataset statistics for Multi-ATGCN 106
Table 5-2 Model performances comparison (population inflow forecasting) 113
Table 5-3 Ablation study for Multi-ATGCN 120
Table 5-4 Comparison of computation cost for population inflow forecasting 122
Table 5-5 Comparison of different adjacency matrices in Multi-ATGCN 124
Table 6-1 Correspondence table between activity type and NAICS code 131
Table 6-2 Example of a frequency lookup table 135
Table 6-3 Daily activities (%) comparison before/after trip preprocessing 138
Table 6-4 Trip statistics before/after trip preprocessing 139
Table 6-5 Illustration of a confusion matrix (Binary classification) 157
Table 6-6 Model performance comparison: IHTF vs. baselines 164
Table 6-7 Model performance across different loss functions (IHTF) 166
Table 6-8 Model performance comparison: workers vs. nonworkers (IHTF) 170
Table 6-9 Model performance comparison: trip rate (IHTF) 172
Table 6-10 Model performance in PMT and trip distance: workers vs. nonworkers 182
Table 6-11 Statistics of the distance errors between predicted and observed trips .. 184

List of Figures

Figure 1-1 Conceptual framework of big-data-driven travel demand models 4
Figure 3-1 Framework of extracting travel demand from MDLD 33
Figure 3-2 Illustration of trip identification 36
Figure 3-3 Methodological framework for mode imputation 37
Figure 3-4 Comparison of trip rates: MDLD vs. NHTS 2017 40
Figure 3-5 Comparison of trip temporal patterns: MDLD vs. NHTS 2017 40
Figure 3-6 Comparison of VMT per person: MDLD vs. NHTS 2017 41
Figure 3-7 Analytical framework for modeling travel demand during COVID-19 42
Figure 3-8 National mobility trends during the pandemic (multi-source) 44
Figure 4-1 Spatial distribution of CBG-level population inflow (a) and its log-transformed distribution plot (b) 47
Figure 4-2 Distribution of CBG-level population inflow before (a) and after (b) Box-Cox transformation 49
Figure 4-3 Analytical framework for EML models comparison 52
Figure 4-4 An illustration of a new split in a single decision tree 54
Figure 4-5 Contour plot of hyperparameter tuning for LightGBM 67
Figure 4-6 Prediction vs. Observation Plot across EML models 72
Figure 4-7 Model performance (a) and Prediction vs. Observation Plot (b) across different sampling rates 73
Figure 4-8 Impurity importance of tree-based models 74
Figure 4-9 Permutation importance of tree-based models (Shuffling vs. SHAP) 76
Figure 4-10 Standardized coefficients of linear regressions 78
Figure 4-11 Evolution of impurity importance varying across different MAPE 79
Figure 4-12 Sensitivity analysis of impurity importance 80
Figure 4-13 PDPs of the top 20 important features 81
Figure 4-14 ALE plots of the top 20 important features 84
Figure 4-15 SHAP interaction plots of the top 20 important features 86
Figure 5-1 Spatiotemporal patterns of population inflow 91
Figure 5-2 The Multi-ATGCN architecture 93
Figure 5-3 The ZMGCN architecture 101
Figure 5-4 Normalized time series of weekly average population inflow 106
Figure 5-5 Illustration of external variables in Baltimore 107
Figure 5-6 Forecasting results of the top and last three census tracts (24-step) 116
Figure 5-7 (Top 3) Model performance varying by POI counts 118
Figure 5-8 Model performance varying by lower bounds 119
Figure 5-9 Influence of different core parameters on model performance 123
Figure 5-10 Spatial patterns of four types of adjacency matrices 125
Figure 6-1 Spatial distribution of individual trip origins and destinations 129
Figure 6-2 Weekly evolution of the spatial distribution of individual trips 129
Figure 6-3 Hierarchical activity-based framework (HABF) 130
Figure 6-4 Illustration of activity labeling (a) and distribution of activities (b) 133
Figure 6-5 Illustration of short trip merging using DBSCAN 136
Figure 6-6 Distribution of daily activities before (a)/after (b) trip preprocessing 138
Figure 6-7 Tile plot of a device's hourly activities in two months 141
Figure 6-8 Daily evolution of trip counts by activity types 141

Figure 6-9 Distribution of hourly activity chains: weekday vs. weekend 142
Figure 6-10 Hourly trip counts by activity types: weekday vs. weekend 143
Figure 6-11 Illustration of processing activity chain time series for IHTF 145
Figure 6-12 IHTF architecture 147
Figure 6-13 Variable selection network architecture 150
Figure 6-14 Heatmap of home (a) and destinations (b) of observed devices 161
Figure 6-15 Predicted vs. real hourly trip counts using different loss functions 169
Figure 6-16 Predicted vs. real hourly trip counts: workers (a) vs. nonworkers (b) .. 170
Figure 6-17 Spatial distribution of trip rate: observation (a) vs. prediction (b) 171
Figure 6-18 Relative feature importance in IHTF 173
Figure 6-19 Attention weights to previous hours for predicting the next 24 hours .. 174
Figure 6-20 Attention weights to previous hours when predicting the first future hour across different attention heads 175
Figure 6-21 Scatter plot of CBG-level OD volume: prediction vs. observation 179
Figure 6-22 Spatial plot of CBG-level OD flow: prediction (a) vs. observation (b) 179
Figure 6-23 Predicted vs. observed hourly total miles traveled: workers (a) vs. nonworkers (b) 180
Figure 6-24 Predicted (a) vs. observed (b) trip distance distribution 181
Figure 6-25 Heatmap for workers' trips: prediction (a) vs. observation (b) 183
Figure 6-26 Heatmap for nonworkers' trips: prediction (a) vs. observation (b) 183
Figure 7-1 Population flow time series by travel modes (a) and activities (b) 193
Figure 7-2 Spatial distribution of OD flow by activities 193
Figure 7-3 Prediction outcomes without (a)/with (b) walk-forward validation 195
Figure 7-4 Conceptual framework of connecting MDLD with traffic simulators 198

List of Abbreviations

Abbreviation	Description
ACS	American community survey
AI	Artificial intelligence
ALE	Accumulated local effect
ARIMA	Autoregressive integrated moving average
ARMA	Autoregressive moving average
AWS	Amazon web services
CBG	Census block group
CDR	Call detail record
ChebNet	Chebyshev spectral network
CNN	Convolutional neural network
COVID-19	Coronavirus disease 2019
DB1B	Airline origin and destination (DB1B) survey
DCRNN	Diffusion convolutional recurrent neural network
DTA	Dynamic traffic assignment
DTALite	An open-source dynamic traffic assignment package
EML	Explainable machine learning
EMR	Amazon Elastic MapReduce
FN	False negative
FNN	Fully-connected neural network
FP	False positive
FSM	Four-step model
GAM	Generalized additive model
GBDT	Gradient boosting decision trees
GCN	Graph convolutional network
GCRNN	Graph convolutional recurrent neural network
GIS	Geographic information systems
GNN	Graph neural network
GPS	Global positioning system
GRN	Gated residual network
GRU	Gated recurrent unit
HABF	Hierarchical activity-based framework
HMM	Hidden Markov model
ICE	Individual conditional expectation
IHTF	Interpretable hierarchical transformer
ITA	Incremental traffic assignment
LARS	Least angle regression
LIME	Local interpretable model-agnostic explanations
LSTM	Long short-term memory
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MATSim	An agent-based transport simulation framework

MAU	Monthly active users
MDLD	Mobile device location data
MG	Montgomery
MLP	Multi-layer perceptron
MSE	Mean square error
MTGNN	Multivariate temporal graph neural network
Multi-ATGCN	Multi-graph Multi-head Adaptive Temporal Graph Convolutional Network
NAICS	North American industry classification system
NHTS	National household travel survey
NTD	National transit database
OD	Origin-destination
OLS	Ordinary least squares
OSM	Open street map
PA	Production-attraction
PMT	Person miles traveled
PDP	Partial dependence plot
POI	Point-of-interest
QAQC	Quality assurance and quality control
RAU	Regularly active users
RF	Random forest
RMSE	Root mean square error
RNN	Recurrent neural network
S2S	Sequence-to-sequence
SafeGraph	A location data company
SARIMA	Seasonal autoregressive integrated moving average
SHA	Successive halving algorithm
SHAP	Shapley Additive explanations
SMOTE	Synthetic minority over-sampling technique
St.d.	Standard deviance
SVD	Singular value decomposition
SVM	Support vector machine
T-100	T-100 domestic market data
TAZ	Traffic analysis zone
TCN	Temporal convolution network
TGCN	Temporal graph convolution network
TN	True negative
TP	True positive
TVT	Traffic volume trends
VISSIM	A microscopic traffic flow simulation software
VMT	Vehicle miles traveled
ZMGCN	Zone-specific Mix-hop GCN

1 Chapter 1: Introduction

1.1 Background

Travel demand modeling has been investigated at least since the early 1950s [1, 2]. Traditional travel demand models require onerous surveys for collecting detailed travel behaviors and traveler information, which are costly for administrators, laborious for investigators, and burdensome for participants [3-5]. As a result, surveys suffer from limited sample sizes, low update frequencies, under-reporting biases, and response fatigue [6, 7]. These inherent limitations inevitably restrict the capability of traditional models in replicating base scenarios, predicting scenarios in the future, and responding to sophisticated policies and unexpected interventions [5, 7, 8].

The burden of data collection is further exacerbated when the paradigm shifts from aggregated four-step models to disaggregated activity-based models [9-11], which impels scholars to find alternatives for assessing travel demand [12-14]. Over the past decades, with the advancement of geo-tracking technologies and the prevalence of mobile devices, mobile device location data (MDLD) have become the widespread source for estimating travel demand [15, 16]. Unlike surveys, MDLD are collected continuously and unobtrusively, enjoying the merits of lower cost, higher penetration rate, higher frequency, and finer spatiotemporal resolution. Due to these attributes, considerable efforts have been devoted to parsing MDLD to characterize travel demand [17-20], providing critical evidence that MDLD can essentially complement traditional travel surveys.

The game-changing data, in conjunction with novel methods, have facilitated opportunities to build a data-driven travel demand model. The overwhelming penetration rates of mobile devices offer the chance to measure population-representative travel demand. Measures can be used to reexamine well-founded knowledge accumulated from small-sample surveys [12]. A fundamental topic is identifying factors associated with travel demand and uncovering underlying relations [21]. In addition, the continuous MDLD stream allows the modeling of travel demand continuously instead of by snapshot [13, 22]. For example, population flow can be continually derived from MDLD [17, 18], thus accumulating sufficient time series to train forecasting models. Forecasted outcomes can be fed into traffic simulators to generate future link-level traffic states. More importantly, the rich individual travel details in MDLD provide great potential to model travel demand in a bottom-up way [23], which is highly compatible with the disaggregated model paradigm [24, 25].

Although MDLD provide a great possibility to revolutionize the existing travel demand models, limited studies [17, 18, 25] have successfully realized a big-data-driven, dynamic travel demand model using MDLD. A complete data-driven dynamic travel demand model requires not only estimating travel demand using big data but also forecasting future travel demand in a timely and credible manner. Several challenges should be overcome in travel demand forecasting. At an aggregated level, population flow time series forecasting involves massive data amounts with high dimensionality, spatiotemporal dependency, zone-specific patterns, and complex dynamics triggered by external factors [26-32]. At an individual level, methods for simultaneously predicting activity, time, and location

for individuals with or without observations remain unaddressed due to individual heterogeneity, multi-task learning, and the cold start issue [33, 34]. Classical statistical methods have difficulties handling these challenges while emerging deep learning methods provide a potential solution [35, 36]. In addition, although MDLD contain extensive travel information, it often lacks important contextual socioeconomic and demographic information due to the passive collection nature and privacy concerns. The information-poor predicament curtails model capability in responding to policies, interventions, or individual changes [20, 24]. Thus, a unified and higher-level knowledge fusion and discovery framework should be developed for handling multisource data, including travel surveys, MDLD, and other related information.

This research is intended to provide a unified and comprehensive guide for constructing a data-driven travel demand estimation and prediction framework integrating a variety of big data resources and novel techniques. This includes 1) a normative pipeline for parsing MDLD to derive trip rosters and population flow matrices, 2) a set of explainable machine learning (EML) methods for delineating the nonlinear relations between travel demand and its determinators, 3) a set of spatiotemporal neural networks for forecasting population flow time series, and 4) a hierarchical activity-based framework (HABF) for individual trip itineraries forecasting. By performing analyses on both nationwide and citywide human mobility data, this study promises to address various theoretical and applied challenges, enhance individual and aggregated prediction power, integrate multi-source multi-dimensional data, and respond to a wide range of scenarios.

1.2 Research framework & objectives

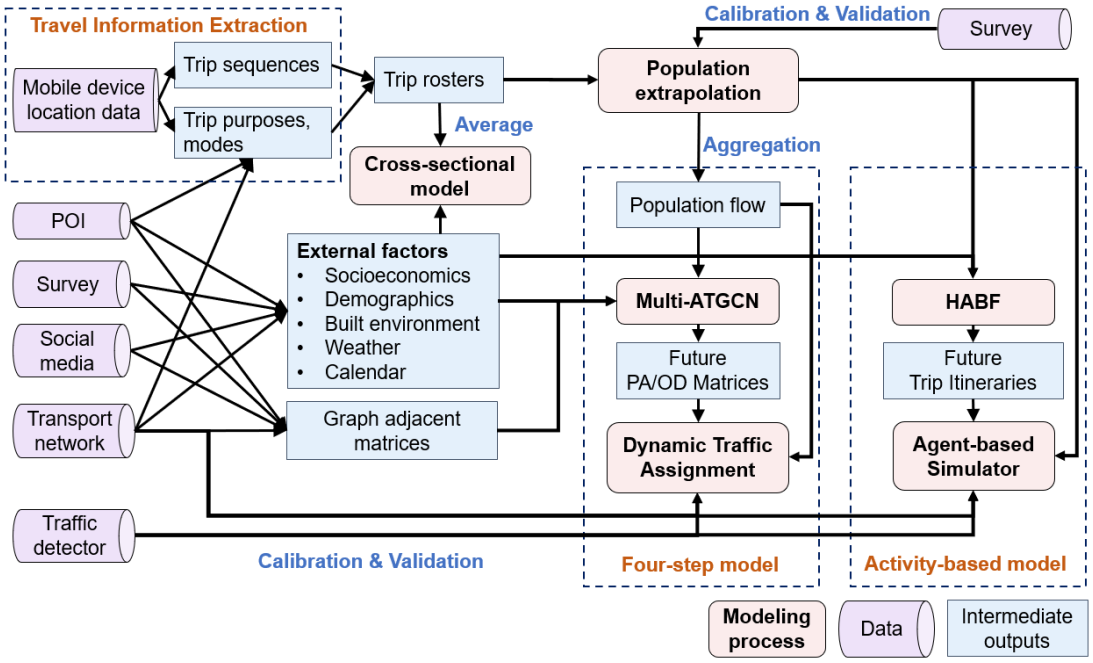


Figure 1-1 Conceptual framework of big-data-driven travel demand models

The conceptual framework of a big-data-driven travel demand model is summarized in **Figure *1-1***. First, travel demand measures, including aggregated population flow matrices and individual trip rosters, are extracted from MDLD, extrapolated to the entire population, and compared with surveys for validation. Next, several cross-sectional EML models are built on top of these measures, in tandem with other data sources, to understand factors associated with travel demand. Last, several forecasting models are constructed to forecast future travel demand time series, including a multi-view temporal graph neural network (TGCN) to forecast citywide population inflow and a hierarchical activity-based framework (HABF) to forecast future individual trip itineraries. Note that to complete a whole cycle of travel demand models, traffic simulation is required. Forecasted travel demand can be fed into traffic simulators, such as dynamic traffic assignment and agent-based simulators, to obtain link-level

traffic measures. The simulation, however, is not included in this study due to the lack of time and resources. Specifically, this research mainly attempts to achieve the following objectives:

1) The first goal of this research is to extract travel demand information from MDLD and demonstrate its reliability. Two measures are produced, including population flow matrices and trip rosters, corresponding to intermediate inputs of four-step models and activity-based models, respectively. Outcomes are extrapolated to the entire population via some weighting processes and are compared with different surveys to ensure product quality. Note that this step is not intended to entirely replace travel surveys with MDLD, but to justify that MDLD can substantially complement surveys by providing solid travel demand information.

2) Based on extracted travel demand measures, this study reexamines the well-founded relations between travel demand and underlying factors such as built environment, socioeconomics, and demographics. The purpose of this step is twofold. First, by using travel demand measured by MDLD, this step attempts to validate previous findings that are derived from small-sample surveys. Second, by using EML models with the capability of handling and interpreting nonlinearities, this step aims to uncover more details in relations revealed by linear statistical models.

3) Aided by continuously collected MDLD, abundant travel demand time series are available. Another main objective of this study is to radically enhance the accuracy of travel demand forecasting by learning patterns from historical data, aggregately and individually. At an aggregated level, this study introduces a multi-view TGCN to handle spatiotemporal dependencies, diverse temporal patterns, and

complex external effects in population flow time series. At an individual level, this study proposes a hierarchical activity-based framework to simultaneously predict the activity, time, and location of the next trip for each person. By integrating a range of deep learning techniques, this step is intended to inform the best practice for travel demand forecasting, considering both accuracy and efficiency in handling large-scale multi-source spatiotemporal data at both individual and aggregated levels.

1.3 Contributions

This study is among the first to propose, implement, and validate a big-data-driven travel demand estimation and prediction framework with the capability of assimilating massive MDLD, meshing it with other data, and achieving end-to-end travel demand estimation and prediction at both aggregated and disaggregated levels. Specifically, the major contributions can be viewed as follows:

1) This dissertation effectively extracts nationwide travel demand information from MDLD with reliable quality validated by a range of travel surveys. This is consequential since compared with surveys, MDLD allow for the characterization of human mobility at an unprecedented spatiotemporal resolution, with near-real-time updating frequency, covering a large population and geographic area, and in a continuous and unobtrusive collection manner [7, 13, 22]. These natures contribute to a more timely, accurate, population-representative, and cost-effective option for estimating travel demand. Hence, the limitations of customary travel demand models [5, 10] caused by surveys are overcome in this dissertation.

2) This research examines relations among travel demand and related factors at a nationwide near-population level. Previous knowledge of such relations is

accumulated from small-sample surveys [37-39]. The adoption of population-representative travel demand measures contributes to fewer sampling biases and more reliable model estimations. Moreover, this research introduces a variety of EML models in conjunction with model-agnostic interpretation approaches to comprehensively understand nonlinear and interactive relations [40-42]. This cross-sectional analysis is nontrivial as it provides versatile functions to visualize and understand complex relations among travel demand and external environments, offering new perspectives for transportation modelers and urban planners.

3) This study introduces a novel multi-view TGCN framework for citywide population inflow time series forecasting. The proposed framework is more comprehensive than previous studies as it integrates multi-head temporal patterns, multi-view spatial structures, and multi-dimensional auxiliary effects. Moreover, the TGCN framework combines adaptive learning with mix-hop convolution to process graphs given (or without) prior knowledge of their structures, providing a guide to handling spatiotemporal data with complex temporal dynamics and unclear graph structures. Last, the TGCN framework handles external variables separated by static or temporal dimensions, which can decrease the risk of polluting the target time series while remaining the model's capability in learning external information.

4) This research is also the first to establish a hierarchical trip itinerary forecasting framework that can simultaneously output the next activity, departure time, and location for each individual. Several challenges are well addressed. First, the joint prediction of activities, time, and places is accomplished following activity-based models, that is, first determining the activity chain for each individual and then

generating locations based on predicted activities and historical visits. Second, the imbalanced distribution of activities (i.e., some types of activities occur more frequently than others) is addressed by introducing a set of semantic segmentation losses [43]. Third, the cold start issue, i.e., the difficulty in predicting places without historical visits [44], is solved by combining the local plus global probability.

1.4 Organization

This research is organized as follows. The second chapter is dedicated to the literature review. The traditional travel demand models and emerging travel-related data sources are summarized, with a particular emphasis on MDLD. Also, cutting-edge techniques used in travel demand forecasting are comprehensively reviewed.

The third chapter provides a brief introduction to the pipeline of parsing MDLD for inferring travel demand. Then, in the fourth chapter, a set of EML models is established to delineate the relations among MDLD-based census block group (CBG)-level travel demand and underlying factors.

The fifth chapter focuses on population inflow time series matrices forecasting. A Multi-graph Multi-head Adaptive Temporal Graph Convolutional Network (Multi-ATGCN) is proposed for multi-step citywide population inflow forecasting. The sixth chapter is devoted to individual trip itinerary forecasting. A hierarchical activity-based framework (HABF) is proposed to jointly predict the activity, departure time, and location of the next trip.

The last chapter summarizes key findings, limitations, and future directions.

2 Chapter 2: Literature Review

2.1 Traditional travel demand modeling

Although recent years have witnessed the advent and prevalence of new data sources and advanced approaches which have provoked waves of revolution in travel demand modeling, it is still important to note that classical travel demand models remain the mainstream in transportation planning [10]. Most of the advanced technologies were developed not to construct a new paradigm, but to modify and reinforce the traditional framework by enhancing its capability and reliability for certain types of applications [8]. Therefore, reviewing traditional travel demand models and understanding their respective procedures, capability, and weakness, are essential to help enhance the existing models by incorporating new data and techniques.

2.1.1 Four-step model

The history of travel demand models has been dominated by the four-step model (FSM) [1]. A brief introduction of the four steps is as follows [45, 46]. First, in the **trip generation**, a regression model or cross-classification analysis is used to predict the number of trips produced from (and attracted to) all TAZs as a function of land use, socioeconomics, and demographics, yielding Production-Attraction (PA) tables. Second, in **trip distribution**, trip productions are distributed to match trip attractions based on well-established theories, such as the gravity model and intervening-opportunities model, to reflect underlying travel impedance, yielding the Origin-Destination (OD) matrices. Third, in **mode choice**, the OD matrices are split into multiple OD tables to reflect relative proportions of trips by modes. Utility functions

that show the mode choice preferences are estimated under the discrete choice theory using data collected via state-preference surveys. Finally, in **traffic assignment**, multimodal OD matrices are assigned to mode-specific networks, yielding link-level traffic flow, passenger flow, or freight volume.

Limitations of FSM have been widely discussed in previous studies [5, 8, 9, 47, 48]. The first fundamental criticism of FSM is its strong aggregate nature, both in time and space. Origins and destinations of all trips per OD pair are treated as single points; daily rhythms of travel demand are simplified as peak and off-peak patterns; households in the same TAZ are treated as identical or simply divided into a few segments. The aggregation induces the possible ecological fallacy since aggregated patterns of travel do not explicitly indicate individual behaviors. The second limitation of FSM is its weak dynamics. Zone and household characteristics that dictate travel behaviors are assumed to be stable over the entire analysis timeframe. Meanwhile, FSM has difficulties capturing weekly and seasonal patterns, as well as perturbations triggered by weather, holidays, or other events. Last, most criticisms are related to the lack of behavioral realism in FSM. FSM does not contain enough individual-level behavioral constructs and choice mechanisms reflecting the nature of human decision-making. This further leads to the neglect of individual interactions and behaviors, resulting in poor performance and low sensitivity to increasingly complex policies.

2.1.2 Activity-based model

The change from large-scale infrastructure construction to fine-grained travel demand management necessitates more disaggregated and powerful travel demand models

[10], among which an activity-based model has been promoted as the most successful alternative [9]. An activity-based model recognizes that the travel demand is derived from the desire to participate in activities, rather than the desire to travel. Moreover, in contrast to FSM using the trip as the basic analyzed unit, the analyzed unit of activity-based models is defined by chaining related trips termed “tours” [48]. A summary of the implementation of an activity-based model is as follows [8]. First, a synthetic population is created, followed by a long-term structure to extrapolate it to future scenarios. Next, a set of generation models is applied to predict the number of activities by purpose, as well as how these activities are formed into tours. A set of tour-level models are then applied to each tour to assign the destination, the departure and arrival time, and the travel mode. Next, the list of tours is converted into trips, and for each trip, the trip-level models are applied. Ultimately, trips are aggregated into OD matrices and passed through the traffic assignment, as in the FSM.

Although activity-based models have successfully avoided some inherent limitations of FSM, one remaining issue is the heavy reliance on survey data [10]. Traditional travel diary surveys involve costly and labor-intensive processes for collecting and processing data. As a result, surveys are often limited by infrequent updates, small sample sizes, short survey durations, under-reporting, and survey fatigue [12]. Compared with FSM, the activity-based model is more disaggregated, requiring more individual/household-level travel details. Thus, greater data collection efforts are needed for activity-based models to achieve reliable performance [11]. Such a high data requirement, however, increases the perceived complexity of activity-based models and becomes a significant barrier to preventing their rapid

prevalence [8]. Highly relying on survey data also indicates that activity-based models cannot avoid the inherent weakness of the survey itself, for example, the poor dynamics [7]. Changes in travel demand caused by abnormal interventions cannot be timely captured by activity-based models since traditional surveys are mainly carried out on normal days to capture typical travel patterns. Also, medium and long-term dynamics in travel demand remain difficult to estimate due to the lack of sufficient longitudinal survey data. In sum, the shift to disaggregated modeling consequentially requires more individual travel data with finer spatiotemporal resolutions [24], pushing researchers to seek new options for acquiring and assessing travel demand.

2.1.3 Travel demand and underlying factors

Either FSM or activity-based models recognize that travel demand is significantly associated with travelers' socioeconomic and demographic factors, as well as the built environment near trip origins and destinations [37, 38]. Several studies provide economic and behavioral explanations of why these factors might influence travel demand [49-51]. For example, a well-known “3Ds” framework [21], including density, diversity, and design, which was later expanded to “5Ds” [37], followed by destination accessibility and distance to transit, was proposed to summarize the effects of the built environment on travel behaviors. Via the meta-analyses, Ewing and Cervero [37, 38] noted that trip frequencies were primarily a function of socioeconomics and secondarily a function of the built environment. They also found that population and job density lose significance once other factors were controlled.

Most of the studies before 2010 examined the relations between travel demand and its underlying factors using statistical models such as (generalized) linear

regression, discrete choice models, and structural equation models, based on data collected from travel surveys [39, 50-52]. Several main changes can be observed in recent studies. First, examining the demand for emerging options such as shared mobility [53-56] and autonomous vehicles [57, 58], has attracted considerable interest. Second, methods have shifted from classical linear-based models to more complex models with the capability of capturing nonlinear effects, such as generalized additive (mixed) models [53, 54, 59] and machine learning methods [42, 55, 60]. Third, data used to assess travel demand have also been shifted from surveys to big data sources including mobility service transactions [61, 62], transit smart cards [62, 63], location-based service data [64], and traffic detectors [65].

2.2 Mobile device location data (MDLD) in travel demand modeling

Nowadays, advanced wireless communication, unobtrusive and ubiquitous sensing and positioning technologies, and large-scale computing infrastructures have produced vast amounts of travel data [4, 66], changing how data are used in assessing travel demand [22] and challenging the extent to which new data sources can complement travel diaries surveys [67]. For simplicity, this section particularly focuses on MDLD as well as how it can be used in travel demand modeling.

2.2.1 Mobile device location data (MDLD)

The 21st century witnessed the wide spread of mobile devices, as well as the pervasive usage of positioning technologies [3, 23]. Data from various positioning sensors, including GPS satellites, cellular tower pings, Bluetooth, short-range positioning beacons, and Wi-Fi networks [6, 23], are integrated to reflect high-

resolution mobile device locations and thus provide massive spatiotemporal information describing individual-level mobility patterns, enabling researchers to quantitatively analyze travel behaviors and estimate travel demand [20, 22, 24].

Among the most widespread MDLD are the mobile phone Call Detail Records (CDRs), which consist of time-stamped tower locations with caller IDs when the user places a phone call or sends a message [4, 7, 20]. In recent years, another similar but more accurate mobile phone dataset called “sightings data” has received wide attention. Sightings data are generated each time a phone is positioned and are likely to have a higher level of spatiotemporal resolution [7, 68]. Typically, a sightings record includes an anonymized device ID, timestamp, and triangulated latitude and longitude [7, 68]. Other popular MDLD sources include GPS trajectories collected by navigation applications or dedicated applications designed by researchers to collect travel data, geotagging information created on social media, and transaction data collected by commercial companies [4, 6, 22]. Unlike surveys or traditional traffic sensors, MDLD allow for the characterization of human mobility at an unprecedented spatiotemporal resolution with population-representative penetration rate, large spatial coverage, longitudinal and continuous temporal span, and near-real-time update frequency [12]. Also, the passive nature of the MDLD removes the reliance on human responses, reducing the labor and time cost, facilitating study over longer periods and wider coverage, and aiding data accuracy [4].

Due to these unique and fascinating advantages, MDLD have gained substantial research attention over recent years. Studies have demonstrated the strong capability of MDLD in capturing travel patterns [25], identifying activities [69],

detecting transport modes [70], inferring trip purposes [17], and formulating origin-destination matrices [17, 71]. During the COVID-19 pandemic, numerous studies have also used MDLD to monitor human mobility, and further explored the relations between human mobility and COVID-19 health outcomes [16, 64, 72].

Although MDLD overcome many shortcomings of traditional data acquisition approaches and inevitably come into mainstream use, whether they can entirely supplant travel surveys remains controversial [12, 23]. The higher level of passivity in collecting MDLD presents significant challenges when it comes to acquiring contextual individual information [12]. In addition, MDLD do not explicitly provide the data that transportation engineers needed, such as trip, mode, and activity. Hence, considerable efforts are required to process raw data, infer useful information, and validate outcomes, which are computationally intense and need support from surveys [13, 25]. Moreover, sampling bias still exists in MDLD despite their high penetration rate. In sum, inputs from surveys are still necessary to make up for the lack in MDLD of contextual information related to individuals and to validate outputs [3, 7].

2.2.2 Travel demand estimation using MDLD

Using MDLD to estimate travel demand has received substantial attention in recent years [73]. MDLD alone cannot directly provide detailed information needed for a complete travel demand model. Therefore, significant efforts are devoted to distilling useful information such as trips, modes, activity types, and OD matrices from MDLD that can be incorporated into travel demand models, including both FSM [18, 71, 74] and activity-based models [25]. A normative pipeline of parsing MDLD to extract travel information can be synthesized in four steps: trip identification, activity

inference, mode imputation, and OD matrices estimation [25]. Once the multimodal OD matrices are formulated, traffic assignment can be conducted to further generate the mode-specific link-level measures [17, 19, 71].

Trip identification: To extract a trip from a sequence of successive traces, the first step is to distinguish the stays (i.e., locations where the person is engaging in some activities) from the moving pass-bys. A stay is characterized by "spending some time in one place" [75]. Therefore, most studies identified the stays by bounding the sequence of traces with a set of pre-defined temporal and spatial constraints [13]. The spatial constraint is the roaming distance when a user is staying at a location, which is typically set as 200-300 m [25, 76], while the temporal constraint is the minimum duration spent at a location that can be viewed as engaging in an activity, which is set as 10-20 minutes [71, 76]. After detecting the stays, the trip roster can be generated using stays as origins and destinations. Other trajectory points are labeled as pass-bys and sometimes can be further used to infer travel paths [25, 71]. Some studies also conducted a clustering process after the identification of stays. The reason is that positioning noises result in multiple candidates that are in the same place being estimated at slightly different coordinates. To account for this, clustering is used to consolidate candidates into a single point [13, 17].

Activity inference: Linking land use and POI information of a diverse region that an individual visits with the explicit activity that the individual engages in is challenging since MDLD are passively collected, lacking the ground truth of activity types [77]. However, some activities such as home and work are relatively easy to infer due to their high regularity in people's daily itineraries. Rule-based methods are

widely used to identify home/work locations, taking into account visit frequencies and temporal patterns. For example, a person's home location is defined as the stay point where he/she visits most frequently between 08:00 PM and 07:00 AM (+1) on weeknights, while a person's work location is defined as the stay point other than home that he/she visits the most between 07:00 AM and 08:00 PM on weekdays [17, 71]. Some other rules are further applied to distinguish telecommuting or unemployed persons. For example, the work location is considered blank if the stay point is visited less than twice per week or if it is less than 500 m from the home location [71].

Most studies only consider three activity types, i.e., home, work, and others, and very few of them [69] expand to other types such as shopping, leisure, and education. Although surrounding land use can provide some information about a user's activity, the mixed-use land development easily distracts the model and results in problematic matching precision. Future studies may need to consider data collected from other sensors equipped in smartphones, such as the accelerometer and magnetometer, to enhance the power of activity inference [12].

Mode imputation: Taking advantage of GPS tracking technologies, imputing travel modes using GPS data has attracted wide research interest [78]. MDLD cannot be directly used to train the mode imputation due to the lack of ground-truth mode types. Therefore, GPS-based surveys provide the main data sources [79] for model training, and the trained model can be applied to trip rosters extracted from MDLD to impute trip-level travel modes. Numerous classifiers have been introduced for mode imputation, including tree-based models, hidden Markov model (HMM), rule-based models, Bayesian networks, Support Vector Machine (SVM), deep learning methods,

and hybrid methods [70, 80-83]. Although techniques vary, the underlying assumption is common that travel modes are differentiated by speed, duration, distance, orientation, and acceleration, supplemented with personal historical travel patterns, socioeconomics, demographics, and mode-specific transport networks [79]. Such assumptions guide the selection of features for imputing modes. For example, frequently-used features include the statistics of speed, time, orientation, and acceleration, bus stop, and rail line trajectory closeness. [79, 84].

With the increase in data and the progress in classifiers, most studies have reached a high accuracy of over 85% [84]. Common travel modes such as stationary, non-motorized, and motorized, are easy to detect [80]. A recent research trend is to expand modes to a wider range, using more features and more accurate positioning and motion data. For example, Nitsche et al. distinguish nine modes, including walk, bicycle, motorcycle, car, bus, electric tramway, metro, train, and stationary [14], and report an accuracy ranging from 65% (train, subway) to 95% (bicycle).

OD matrices estimation: After extracting trips, activities, and modes, a series of data processing tasks, including data cleaning, tour completion, and population weighting, are required before aggregating trip rosters to OD matrices. Meanwhile, a range of validation processes is needed after the aggregation by comparing data-driven OD with survey-based OD or with traffic flow recorded by sensors [17, 18].

Data cleaning is carried out to filter out irregular trips, such as trips with extremely high speeds or short durations, and abnormal devices, such as devices with overly frequent travels or too few observations [17]. Tour completion is based on the assumption that a user always starts and ends at home within a day, where a day is

defined as a 24 h period beginning and ending at 3 AM. If not, the home location is added to the stay point sequence, and the departure time is assigned based on a conditional probability, which forms a new sequence to form a complete tour [25, 71]. Population weighting is in expanding the observed trips to represent all individuals in the study area. The simplest method is to divide the actual population by the number of devices whose home locations are located in that area to obtain the expansion factor. A variety of complex weighting algorithms are proposed to address different sample biases. For example, Toole et al. first scaled trips based on phone usage frequency and then expanded trips to account for market penetration rates [71]. Iqbal et al. determined weighting factors by minimizing the gap between the simulated traffic volume based on sampled OD matrices and field observations [18].

Most studies compare data-driven OD matrices with survey-based OD matrices [73] to demonstrate their reliability. One consensus is that high precision can only be obtained when the aggregation unit is coarse, while fine aggregation units could lead to noisy and unbalanced OD representations [17, 71]. For example, Alexander et al. found that the correlation between MDLD-based and survey-based tract-to-tract and town-to-town flows is 0.45 and 0.99, respectively [17]. Some studies also passed their OD matrices through a traffic assignment tool and compare the road traffic volume and speed with ground truth and found a fairly high correlation even via a simple assignment algorithm [17, 19].

2.3 Spatiotemporal travel demand modeling

The proliferation of big data, combined with the sheer power of AI techniques, has prompted a revolution in travel demand modeling using AI-driven methods [85].

Researchers can either validate existing knowledge using new data or introduce AI methods to capture nonlinearities and spatiotemporal dependencies in travel demand. Spatiotemporal travel demand modeling is one of the most thriving fields in recent years [35], with numerous AI models such as the convolutional neural network (CNN), the recurrent neural network (RNN), the graph neural network (GNN), and their hybrids being introduced [86, 87]. This section gives a comprehensive review of related studies from cross-section to spatiotemporal modeling of travel demand.

2.3.1 Cross-sectional modeling and nonlinearity

Cross-sectional modeling refers to analyzing the associations between average travel demand and underlying factors. One subject that has received substantial attention in recent years is nonlinearity [88]. Due to the amalgam of processes related to aggregated socialization, gaming, tolerance, contagion, and diminishing returns, relations between travel demand and its determinants may be nonlinear or piecewise [41]. Early studies relaxed the linear assumption by pre-specifying a nonlinear function such as the exponential function [89, 90], which, however, is restrictive and may not well fit the complex nonlinearity. New methods are thus introduced to capture the nonlinearity in a nonparametric setting. A representative statistical method is the generalized additive model (GAM) [91], which is a semi-parametric model with linear predictors involving a series of additive non-parametric smoothers of covariates to capture nonlinear effects. Via the GAM, studies have proved that nonlinear effects are prevalent in relations between external environments and active travel, shared mobility, bike-and-ride, and highway volume [53, 54, 62, 92, 93].

Explainable Machine learning (EML) is another popular non-parametric technique in learning and explaining nonlinearities, among which tree-based models are the mainstream [94, 95]. In the field of transportation, via EML and related interpretation techniques, numerous studies have demonstrated the salient nonlinearity between external factors and travel-related features such as driving distance [96, 97], mode choice [98], shared mobility usage [55], car ownership [42], crash frequency [99], transit ridership [100], and active travel [101, 102].

2.3.2 Explainable machine learning (EML)

Although studies on ML models mainly focus on their predictive accuracy, the interpretability of ML has also gained wide attention [103]. The need for interpretability arises from an "incompleteness in problem formalization" [104], which means that, for certain problems, it is necessary not only to predict but also to understand how such a prediction is obtained. Interpretation techniques developed for ML models help increase model transparency, which is critical to detecting model biases, increasing social acceptance, and uncovering underlying mechanisms[104].

The proposed interpretation techniques can be divided into two types: model-specific and model-agnostic [105]. The former needs the model itself to be explainable while the latter separates the explanation from the model. The impurity feature importance of random forest (RF) is one of the most notable model-specific interpretation milestones [95], which, however, is only suitable for tree-based models. Due to the restriction in models and sometimes the loss in predictive performance, more efforts have shifted from model-specific to model-agnostic methods [103]. A model-agnostic method is more flexible since it can apply to any ML model. For

example, permutation feature importance was proposed to replace impurity importance and has been widely used in different ML models [106]. Partial dependence plots (PDPs) [94] have been applied to a range of tree ensembles and neural networks to delineate learned relations.

Another noteworthy change in ML interpretation techniques is the shift from global to local interpretation. Local interpretation techniques display learned relations at an individual level, which can uncover more local heterogeneity that may be obfuscated by global averaging. For example, individual conditional expectation (ICE) curves are the individual-level building blocks for PDPs [107]. Local interpretable model-agnostic explanations (LIME) focus on training local surrogate models to explain individual predictions [108]. SHapley Additive exPlanations (SHAP) connect the global interpretations with the local interpretations based on the additivity attribute of Shapley values [109].

2.3.3 Temporal modeling (Time-series analysis)

With the deluge of time series data in the transportation domain, temporal modeling, in particular time-series forecasting, has become increasingly active [110, 111]. Main forecasting targets include short-term traffic state (volume, speed, travel time), travel demand (regional population flow, shared mobility demand, transit ridership), and individual movement [86, 112]. Generally, travel demand time series are more challenging to forecast because they involve high-dimensional data structures, intertwined spatiotemporal dependencies, zone-specific dynamics, and complex nonlinearities triggered by external factors such as weather, holidays, and other big events [86]. To address these challenges, a vast number of models are produced and

can be classified into statistical models, traditional machine learning models, and deep learning methods [35, 85].

Previously, statistical models were mainly used, such as AutoRegressive Moving Average (ARMA) [113], Autoregressive Integrated Moving Average (ARIMA) [114], and the variant Stationary and Seasonal ARIMA (SARIMA) [115]. In general, statistical methods have difficulties in handling spatial dependencies, capturing long-term temporal patterns, fitting complex nonlinearity, and incorporating external effects, thus making them ineffective for travel demand forecasting [110].

Since the early 2000s, many researchers have moved from the statistical perspective to machine learning methods [111], among which SVM [116], K-nearest neighbors [117], and tree-based models [40] are the most popular. Machine learning can model nonlinearities and learn complex patterns in time series; however, they require considerable effort in feature engineering, a time-consuming process that requires prior knowledge of the domain and may neglect accountable features [35]. Moreover, traditional machine learning also has difficulties capturing long-term temporal patterns and spatiotemporal dependencies due to its shallow structure, lessening its popularity in traffic time series prediction [85].

As theoretical and computing advances emerge, deep learning is now the most prominent due to its sheer prediction power [118]. Compared with machine learning, feature engineering is performed automatically in deep learning. Meanwhile, the hybrid, deep, and complex structures offer unprecedented potential to capture nonlinearities and spatiotemporal dependencies in traffic time series. The recurrent neural network (RNN) is one of the most famous deep learning structures for

temporal modeling [119]. RNN allows previous outputs to be used as inputs along a temporal sequence while having learnable hidden states to capture temporal dynamics. However, traditional RNNs encounter vanishing or exploding gradient problems in the presence of long sequences. To deal with that, Gated Recurrent Unit (GRU) [120] and Long Short-Term Memory (LSTM) Unit [121] are designed. Early studies directly employed LSTM and GRU for travel demand prediction [122-124], while later studies mainly used the RNN as a component of their hybrid models to handle temporal dynamics [30, 125]. Meanwhile, advanced techniques to augment RNNs' capacity are involved, such as the attention mechanism [126, 127], the gating mechanism [128], and the residual mechanism [129], further strengthening the power of RNN in capturing temporal dynamics.

Due to its recursive structure, RNN suffers from time-consuming iterative propagation and gradient vanishing/explosion problems. As an alternative solution, CNN demonstrates its superiority by tackling temporal sequences in a nonrecursive manner with the advantages of simple structure, parallel computing, and stable gradients [130]. However, traditional 1D-CNN is less effective than RNN due to its failure in storing long-term memory [128]. To address this, the novel convolution operation WaveNet is proposed, integrating causal convolution and dilated convolution [131], which outperforms RNN in text-to-speech tasks. Bai et al. further generalized it for time-series prediction and renamed it the temporal convolution network (TCN) [132]. TCN is now increasingly popular in modeling temporal dynamics due to its superior performance and efficiency [133-135].

2.3.4 Spatial modeling

The most distinguishable feature of the travel demand time series is its spatial dependency. A region's population outflow may affect the inflows of nearby regions [29], and the downstream traffic flow may affect the upstream traffic flow [136]. Early studies directly applied CNN to learn the spatial relation by treating the map of travel demand as an image. For example, most previous studies partitioned a city into squared tiles and applied a bidimensional convolutional kernel to detect spatial dependencies in travel demand matrices [27, 29, 125, 137]. However, a major limitation of CNN is that it is not compatible with spatial structures that do not conform to Euclidean space. For example, road segments that are close to each other may have long routing distances constrained by the topology of the road network [136, 138]. Also, when the spatial tessellation is constructed based on irregular administrative boundaries, such as census blocks or TAZs [26, 30], the adjacency of two pixels in the convolutional filter cannot reflect the real closeness due to the different sizes and shapes of tessellations. The graph neural network (GNN), on the other hand, is more appropriate for these data compared to CNN because of its ability to capture non-Euclidean spatial relations. Due to this advantage, GNN has rapidly become the frontier of travel forecasting in recent years [86]. The graph can well describe the spatial structure of a travel demand map or road network by considering sensors or TAZ centroids as nodes, and dependencies among nodes as edges [112].

This section is focused on convolutional GNN since it is currently one of the most popular methods used in traffic forecasting research. The convolutional GNN can be further divided into spectral-based approaches and spatial-based approaches

[139]. Spectral-based approaches define graph convolutions by introducing filters from graph signal processing, where the convolutional operation is used to smooth graph signals [140]. Spectral-based approaches are differentiated by filters, such as the Spectral network [140, 141], which uses a learnable diagonal matrix as the filter, the Chebyshev spectral network (ChebNet) [142], which approximates the filter by a k^{th} order Chebyshev polynomial, and the Graph convolutional network (GCN) [143], which simplifies the ChebNet by using its first-order approximation. Another popular type of GNN, the spatial-based approach, defines graph convolutions by message propagation to convolve the central node's representation with its neighbors' representations. Representatives include the Diffusion CNN (DCNN) [144], which considers the graph convolution as a diffusion process, the Message-passing neural network (MPNN) [145], which treats the graph convolution as a message-passing process, and the GraphSage [146], which samples neighbors to obtain a fixed number of neighbors for each node. Meanwhile, techniques to augment GNNs' capacity are widely involved, such as the Graph attention network (GAT) [147], which adopts the attention mechanism to learn the node pairwise weights, and the GeniePath [148], which proposes a gating mechanism to control message flow.

One key component of GNN is the adjacency matrix, which is used to describe heterogeneous pairwise relations between nodes. Different traffic problems may have different assumptions regarding node pairwise relations, resulting in various forms of adjacency matrices including fixed matrices, adaptive matrices, and multiple matrices [86]. Fixed matrices assume that spatial relations are fixed and constant over time. Therefore, adjacency matrices are pre-defined based on the

knowledge of the spatial structure from different perspectives, such as POI-based regional function similarity [149], transportation network distance [133, 138], and temporal similarity [150]. The pre-defined fixed matrix, however, may not precisely reflect the real spatial structure due to defective prior knowledge or dynamic evolution [151]. Therefore, the adaptive matrix is proposed to set the adjacency matrix as learnable parameters. Experiments have proved that the adaptive mechanism helps discover hidden spatial structures and enhance the model performance [134, 135]. Recently, due to the increasingly complex spatial structure, studies [26, 30, 125, 149, 152] increasingly adopt multiple graphs to describe underlying spatial dependencies, i.e., the multi-graph GNN. Outputs of the multi-graph convolutions will be fused via weighted summing, averaging, or a fully connected neural network (FNN).

2.3.5 Spatiotemporal modeling

To jointly capture spatiotemporal dependencies, a hybrid framework that fuses different neural networks into an entity is needed. In a hybrid setting, the CNN/GNN is used to capture spatial relations, and the CNN/RNN is used to learn temporal dynamics. Several mechanisms are proposed for fusing these networks [85]. The simplest way is to feed the outputs of all neural networks into a fusion layer to generate final outputs [153, 154]. Another way is to connect neural networks successively. The output of the previous network is fed into the subsequent one [155-157]. The most complex way is to modify the internal network structure to embed one neural network into another. For example, several studies modified the RNN unit to include a graph convolution operation [138, 156].

Besides CNN, RNN, and GNN, auxiliary components are incorporated to strengthen the hybrid framework. One important component is the fully connected neural network (FNN), whose main purpose is to perform tasks such as aggregating outputs, transforming dimensions, and incorporating external effects. For example, a two-layer FNN is widely used to handle external effects, of which the first extracts important information and the second transforms dimensions [26, 29, 152]. Another widely employed technique is the skip connection operation [129, 135, 158], which gathers information from historical representations to mitigate overfitting. Moreover, when graphs are large, node sampling and graph partition techniques are usually needed to divide a large graph into several small components [159, 160].

2.3.6 Population flow forecasting and Next-location forecasting

This last review section is focused on two travel demand forecasting tasks, population flow forecasting, and individual trip itinerary (Next-location) forecasting, to describe some main practices in travel demand forecasting that jointly considers both spatial and temporal dependencies using deep learning methods. These two tasks are also in line with the main scopes of this dissertation.

Most of the population flow forecasting is done to predict the regional inflow/outflow. The seminal work, ST-ResNet [29], consists of three modules that rely on residual CNN [129] to capture spatiotemporal closeness, period, and trend of grid-based citywide population flow. The three modules are dynamically aggregated and further combined with a two-layer FNN that deals with external factors. Another notable work proposes a Deep Multi-View Spatial-Temporal Network (DMVST-Net) to predict taxi demand, which consists of three views: temporal view to model

temporal dynamics via LSTM, spatial view to model local spatial correlation via CNN, and semantic view to model correlations among regions sharing similar temporal patterns via GNN [125]. A similar multi-view structure is proposed by Sun et al. [26] to predict citywide population flows in irregular regions by using GCN. Another popular direction is the multi-graph network. For example, Geng et al. [149] build three graphs considering the neighborhood, function similarity, and connectivity among different regions, to capture complex spatial dependency in ridesharing demand, while Chai et al. [152] also constructed three graphs, including distance, interaction, and correlation, to predict station-level bikesharing flow. Another similar framework, the Spatiotemporal Encoder-Decoder Residual Multi-Graph Convolutional network (ST-ED-RMGC), designs a residual multi-graph convolutional network to encode contextual-aware spatial dependencies and an LSTM to encode temporal dynamics [30].

Population flow forecasting belongs to aggregated travel demand forecasting, while at an individual level, next-location forecasting is the most popular [35]. Next-location forecasting predicts where (always POI-based) individuals will visit given their historical tracks. The problem has previously been explored using probabilistic or pattern-based approaches, such as hidden Markov Chain [161, 162] and tensor factorization [163, 164]. With the prevalence of deep learning, various new methods are proposed, with stronger capabilities of capturing spatiotemporal and semantic dimensions. Liu et al. [33] proposed a Spatial-Temporal RNN (ST-RNN) to model local temporal and spatial contexts with time and distance-specific transition matrices. Feng et al. [165] proposed DeepMove, an attention-based recurrent neural

network [126] for mobility prediction from lengthy and sparse trajectories. Some studies predict not only where but also when the next visit will occur. Du et al. [166] proposed a Recurrent Marked Temporal Point Process (RMTPP) to simultaneously predict visiting time and location. Chen et al. [167] proposed a context-aware model called DeepJMT for jointly predicting mobility and time, which consists of a hierarchical RNN to capture temporal patterns and a spatial context extractor to extract location semantics. Another strand of research focuses on predicting the next location using semantic trajectory [35, 36, 168], which is a sequence of locations labeled with activities being carried out [169]. For example, Ying et al. [170, 171] proposed a Geographic-Temporal-Activity-based Location Prediction (GTS-LP), which takes into account a user's geographic, temporal, and activity-triggered intentions to predict the next location. Yao et al. [34] proposed a Semantics-Enriched Recurrent Model (SERM), which relies on an embedding layer to represent the time, location, and activity coupled with an LSTM to predict the next location.

2.4 Discussion

Travel demand has been studied for decades, involving numerous data, techniques, and theories, fostering a vast scientific production from various aspects. Through a comprehensive literature review, several research gaps emerge as follows:

1) *Cross-sectional modeling*: Although considerable effort has been devoted to extracting travel demand from MDLD, little effort has been made to examine relations among the national MDLD-driven travel demand and its determinants. Although related knowledge has accumulated using survey data [37, 38], it is still important to validate the findings using big data since big data measures a population-

representative travel demand, which may help uncover more patterns and provide more convincing evidence compared with existing knowledge drawn from small samples [12]. Moreover, introducing advanced methods to revisit the documented relations may further reveal new patterns which are previously ignored due to simplified model assumptions [40]. Another noticeable gap is the lack of extensive comparisons of different EML models. Most previous studies trained a single model and explained it via one of two interpretation techniques; however, whether these interpretation outcomes remain consistent under different model architectures, interpretation techniques, and model parameters is unclear. Lastly, few studies have critically considered the effects of outliers, feature dependency, and local heterogeneity on model interpretation outcomes. Although such effects have been shown to negligibly affect the accuracy of most ML models [95, 172], their impacts on interpretation are controversial [105].

2) *Population flow forecasting*: Population flow forecasting has attracted great interest in recent years to address challenges such as diverse temporal dynamics, multi-view graph structures, and heterogeneous external effects. Although deep learning methods have achieved promising results in population flow forecasting, previous studies separately focus on addressing parts of the challenges, while a dearth of studies comprehensively integrates these advances into a holistic entity to test its performance. In addition, external effects are always neglected in previous studies or simply integrated by FNN without considering the diversity in data dimensions. A well-designed module that can handle different types of external variables is absent. Another main limitation of current spatiotemporal models is that their parameters are

globally shareable. The locality is yet to be carefully addressed, which, however, is important for population flow forecasting considering their highly over-dispersion and zone-specific patterns. Last, most studies forecast the population flow at a regional level or even smaller, a citywide population flow forecasting is limited. Large-scale population flow graphs require a model with higher efficiency and lower memory consumption, which is not considered by many complex hybrid frameworks.

3) *Individual trip itinerary forecasting:* Most of the current individual-level human mobility forecasting focuses on the next-location prediction, while a few studies have jointly predicted the location and departure time, although both are important for constructing a complete trip itinerary. In addition, if devices have no mobility history, it is difficult to predict their future locations, which is known as the cold start problem [44]. Such a problem should be addressed in particular in travel demand models since there exists a large number of synthesized populations when extrapolating from the observed samples to the whole population. These synthesized populations do not have historical travel records but also need the model to generate their future itineraries. Third, most studies predict the next locations of devices solely based on their historical visits, which limits the model's capability to predict the visits to new locations. This may not be realistic, particularly in behavior realism. Last, imbalanced distribution widely exists in individual trip itineraries. Some locations are much more popular than others, resulting in the imbalanced distribution of location labels. Meanwhile, some activities such as stationary state, home, and work, also occur more frequently than others, resulting in an imbalanced classification issue that should be addressed by sampling or modifying loss functions.

3 Chapter 3: Extracting travel demand from MDLD

This section briefly introduces methods for parsing raw MDLD to distill useful travel information. The pipeline broadly follows home&work identification, trip identification, mode imputation, population weighting, and result validation (**Figure *3-1***). A real-world application is introduced after the pipeline, demonstrating the methodology's feasibility in quantifying large-scale human travel patterns in near-real-time frequency. It is noteworthy that this data processing module is not the main focus of this dissertation. Instead, the major emphasis is on forecasting the processed travel demand. Hence, this section is abbreviated and more details can be found on the project website [173, 174] and other related publications [64, 70, 175].

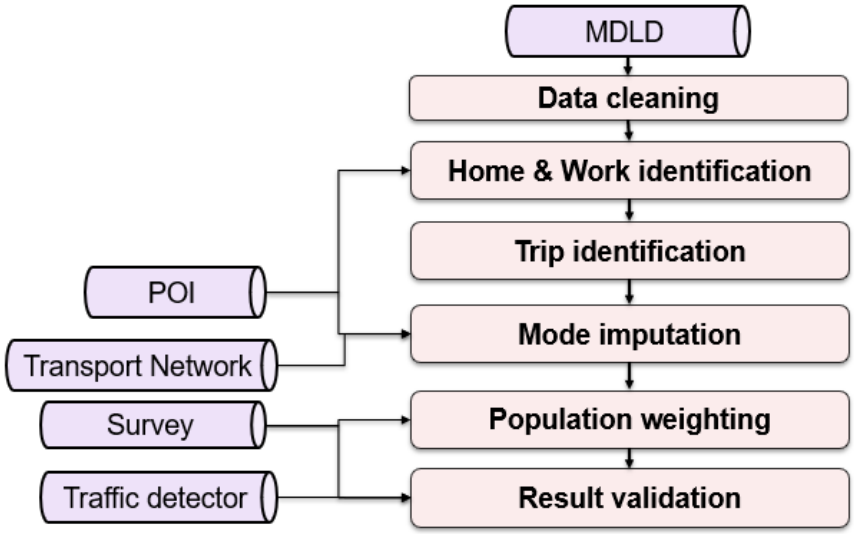


Figure 3-1 Framework of extracting travel demand from MDLD

3.1 Raw data cleaning & Data statistics

The MDLD used in this study is the nationwide sightings data [7, 68], encompassing triangulated coordinates, timestamps, and device ID each time a mobile device is

positioned. Some data cleaning filters are applied to the raw dataset to handle irregular observations, including removing points with invalid data entries (e.g., negative values for coordinates), removing completely duplicate points, deduplicating points with the same timestamp but different locations for the same device, and removing points with abnormal oscillations. After the data cleaning, some statistics regarding quality metrics of the national dataset in 2020 are reported as follows:

Table 3-1 Quality metrics statistics of national MDLD

Metrics	Description	Value
Monthly active users (MAU)	The number of devices with at least one record in a month	~270 million
Regularly active users (RAU)	The number of devices with at least seven days in a month having more than ten records	~68 million
Temporal consistency	The average number of observed days for RAUs in a month	~24 days
Data frequency	The average daily number of records for RAUs	~230
Geographical representativeness	The Gini coefficient of population coverage (by devices) among different counties.	0.4

Note: The Gini coefficient is a value between 0 and 1, with 0 indicating an equal sampling rate in all zones and 1 indicating that all RAUs are from one zone.

Overall, the large number of MAUs and RAUs indicates the mobile devices covered in this dissertation have a reliable representation of the entire population. The temporal consistency and the data frequency imply the location data are updated at a high frequency, and the geographical representativeness indicates that the dataset is evenly distributed across the counties.

3.2 Home&Work identification

Similarly to previous studies [17, 71], this study uses a rule-based method to detect home and work locations. The underlying assumption is that people spend most of their nighttime hours at home and some regular daytime hours at work. One

difference between this study and previous work lies in the data type. Previous studies mostly rely on the CDRs which use the cell tower as the location anchor, while this study uses the sightings data which only provides continuous coordinates. To efficiently process the tremendous amount of continuous locations, this study utilizes geohash [176] to aggregate latitudes and longitudes into location anchors.

Specifically, the home location is identified as level-7 geohash zones (error: ±0.076 km) with the longest dwell time and most frequent visits from 09:00 PM to 06:00 AM per day in a month. The work location is identified as the non-home level-7 geohash zones with the longest dwell time and most frequent visits during common work schedules, i.e., from 06:00 AM to 09:00 PM per weekday in a month. Only those with geohash zones being visited at least three workdays or half of the total observed workdays are labeled as workers, where a visit is defined as a stay with at least a 2-hour duration. The work locations of others are left blank, which may be due to unemployment, telework, or working with no fixed workplaces. The identification algorithm is run monthly. Parameters are determined based on previous studies [17, 64, 71, 175] and a set of surveys including the American Time Use Survey (ATUS), American Community Survey (ACS), and Longitudinal Employer-Household Dynamics (LEHD) Origin-Destination Employment Statistics (LODES).

3.3 Trip identification

This study first defines the tour as the trajectory between two successive at-home observations. For each trajectory point within the same tour, a rule-based recursive algorithm is applied to identify whether it is stationary, pass-by (i.e., belonging to the same trip as its previous point), trip start, or trip end. Unlike previous studies that

only consider time and distance thresholds, this study further involves a speed threshold. The time and distance thresholds are used to identify trip ends. A trip ends only when the device stays longer than 5 minutes and roams within 954 ft (i.e., 300 meters). The speed threshold is used to identify whether the device is stationary. For each point, two types of speed are calculated, namely the backward speed and forward speed (**Figure *3-2***). A device is stationary only when its forward speed ≤ 3 mph. The detailed recursive algorithm is reported in **Table *3-2***. Note that the algorithm may identify a local movement as a trip if the device moves within a stay. Hence, all trips that are shorter than 984 ft are removed.

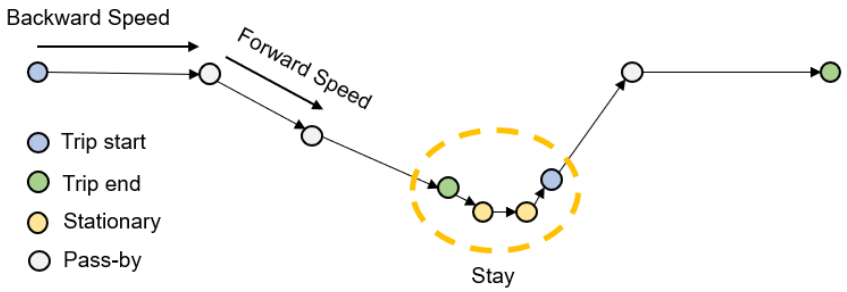


Figure 3-2 Illustration of trip identification

Table 3-2 A rule-based recursive algorithm to identify trips

If P_{t-1} *is stationary*:
If *forward speed > 3 mph*: P_t is a trip start.
Else: P_t is stationary.
Else:
If *backward speed > 3 mph*: P_t is a pass-by.
Else If *backward speed ≤ 3 mph* **and** *backward distance ≤ 984 ft*:
Compute the cumulative duration for all eligible points.
If *cumulative duration < 5 minutes*: P_t is a pass-by.
Else: (A stay is detected)
If *forward speed ≤ 3 mph*: P_t is a trip end.
Else: P_t is a trip start.

3.4 Mode imputation

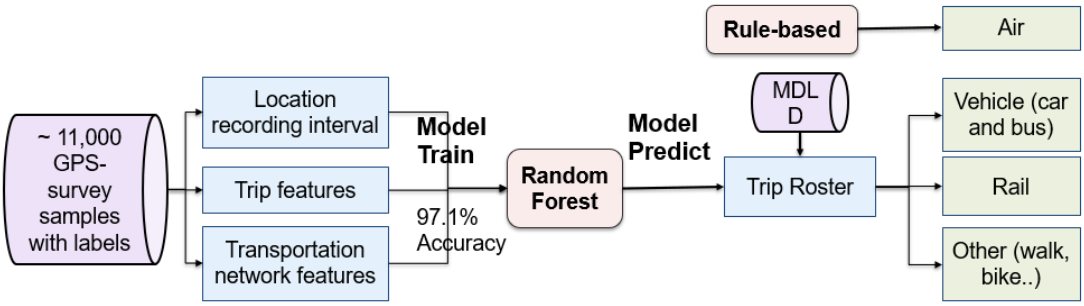


Figure 3-3 Methodological framework for mode imputation

This study proposes a hierarchical mode imputation algorithm (**Figure *3-3***). The air travel mode is first imputed based on a heuristic rule calibrated based on the Airline Origin and Destination (DB1B) Survey. Then, a random forest (RF) [95] is trained to impute ground travel modes taking into account information from both the MDLD itself and the multimodal transportation networks [70]. The air trips are identified using the following rules: the origin-destination air distance should exceed 50 miles, the travel time should exceed 30 minutes, the average travel speed should exceed 75 mph, and the origin and destination distances to the nearest airport should be under two miles. After air trips are imputed, an RF is trained to impute the ground travel modes for non-air trips, including by vehicle (car and bus), rail, and other (walk, bike, and other modes). Features include:

1) Location recording interval feature, represented by the average number of points per minute, which indicates the location service usage during a trip.

2) Trip features, including the origin-destination great-circle distance, network distance, travel time, average travel speed, and different percentiles of travel speed.

3) Transportation network features, including the quantile distance from each point to its nearest rail and bus lines, the distance from the trip origin/destination to its nearest rail and bus stops, and the % points within 165 ft of all rail or bus stops.

An RF is trained using over 11,000 sample data with labeled travel mode information collected by GPS-based surveys [70]. The Synthetic Minority Over-Sampling Technique (SMOTE) is applied to address the imbalanced sample problem by synthesizing the minority class from the existing samples [65]. The randomized search approach is used to fine-tune the model with 10-fold cross-validation. Results show that the RF can achieve 97.1% accuracy for ground travel mode imputation.

3.5 Population weighting

Sampled mobile devices can neither cover the entire population nor all human movements. To address these biases, a two-stage weighting procedure is designed to extrapolate trip rosters to national population-representative estimates. First, device-level weighting is applied to assign a weight to each device whose home location is identified. The device weight, i.e., the county-level sampling rate, is computed by dividing the county population from the ACS by the number of devices residing in that county. The overall sampling rate at the national level is 16.1%. Second, trip-level weighting is proposed to address inherited biases in estimated trips across different travel modes, times of day, and travel distance bands (0-10, 10-25, 25-50, 50-75, 75-100, 100-150, 150-300, >300, in miles). Previous studies have documented that MDLD yield higher trip rates, shorter trip distances, more driving trips, and fewer non-motorized trips than travel surveys [68, 177]. Meanwhile, different levels

of mobile device usage during times of the day may also induce temporal biases. Specifically, trip weights are calculated following the rules:

1) Air travel: The T-100 Domestic Market Data serves as the ground truth of air travel volume. For each month, trip weights by origin state and distance band are developed based on the ratio of air travel volume estimated from MDLD and T-100.

2) Vehicle travel: The 2017 NHTS and the Traffic Volume Trends (TVT) reports are used to assess the ground truth of vehicle travel. The 2017 NHTS provides the vehicle travel volume in 2017; however, for years after 2017, the data are not available. Hence, TVT reports are used in calculating inflation factors to extrapolate the 2017 NHTS vehicle travel volume to other years. Then, for each month, trip weights by census division, time of day, and distance band are calculated based on the ratio of vehicle travel volume estimated from MDLD and inflated 2017 NHTS.

3) Rail travel: The weighting process of rail travel is analogous to vehicle travel except using the 2017 NHTS and the National Transit Database (NTD) to assess the ground truth of rail travel volume.

3.6 Result validation

After completing the previous steps, the national trip roster, including coordinates and timestamps of trip origin and destination, trip distance, travel time, travel mode, device weight, trip weight, and devices' home and work locations, are obtained. The PA and OD matrices can then be generated by aggregating the trip roster into different geographical zones [174]. To ensure product quality, rigorous quality assurance and quality control (QAQC) are developed by comparing data-driven travel

measures with a set of surveys including NHTS, TVT, NTD, DB1B, and T-100. Using the year 2020 as a case, the main validation results are summarized as follows:

1) *Daily trip rate per person*: Figure 3-4 compares the 2020 data-driven trip rates with the 2017 NHTS trip rates at the census division level. While the data-driven trip rates are smaller in 2020 compared to 2017, which can be explained by the COVID-19 pandemic, the overall spatial distribution at the division level is similar. Figure 3-5 shows the comparison of hour-of-day and day-of-year trip rate patterns. The hour-of-day patterns also show high similarity between survey and data-driven products. In addition, a steep fall at the beginning of the COVID-19 pandemic can be observed in day-of-year patterns derived from MDLD.

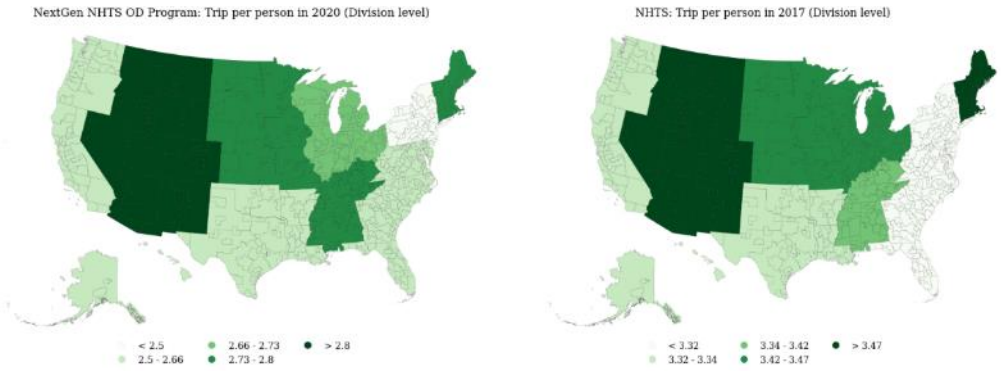


Figure 3-4 Comparison of trip rates: MDLD vs. NHTS 2017

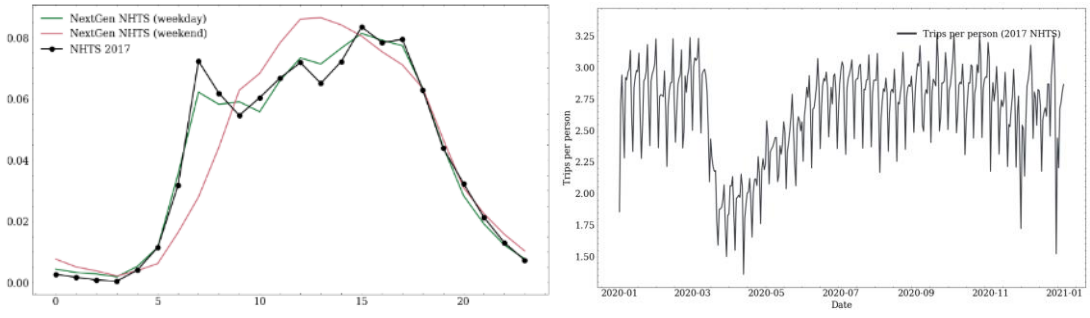


Figure 3-5 Comparison of trip temporal patterns: MDLD vs. NHTS 2017

2) *Daily vehicle mile travel (VMT) per person*: **Figure *3-6*** summarizes the comparison of daily VMT per person using TVT 2020 data at the division level. Overall, the two datasets match well. The data-driven daily VMT per person is 23.1 miles, while the TVT is 23.8 miles, with an average absolute deviation of 3.2%.

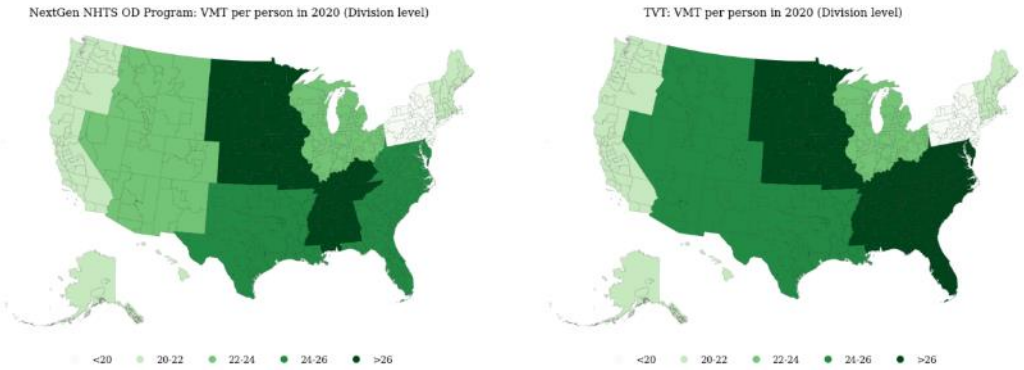


Figure 3-6 Comparison of VMT per person: MDLD vs. NHTS 2017

3) *Air trips*: The data-driven air travel volume shows high consistency with T100 data for the four quarters in 2020. Q1-Q4 percentage errors are 4.88%, 4.47%, 1.54%, and 2.10%, respectively. The overall average absolute error is 2.09%.

4) *Rail trips*: The data-driven rail trip volume is closely in line with NTD rail trip totals. The absolute percentage difference between the two data is 8.16%.

3.7 A real-world application

During the unprecedented coronavirus disease 2019 (COVID-19) challenge, non-pharmaceutical interventions became a widely adopted strategy to limit physical movements and interactions to mitigate virus transmissions. For situational awareness and decision support, quickly available yet accurate big-data analytics about human mobility and social distancing is invaluable to agencies and decision-makers. Following the aforementioned pipeline, this section presents a real-world

implementation that ingests terabytes of MDLD on a daily basis and quantitatively assesses the human mobility trend during COVID-19. Using MDLD of over 150 million monthly active samples in the US, this implementation successfully measures human mobility with three main metrics: daily average number of trips per person, daily average person-miles traveled, and daily percentage of residents staying home.

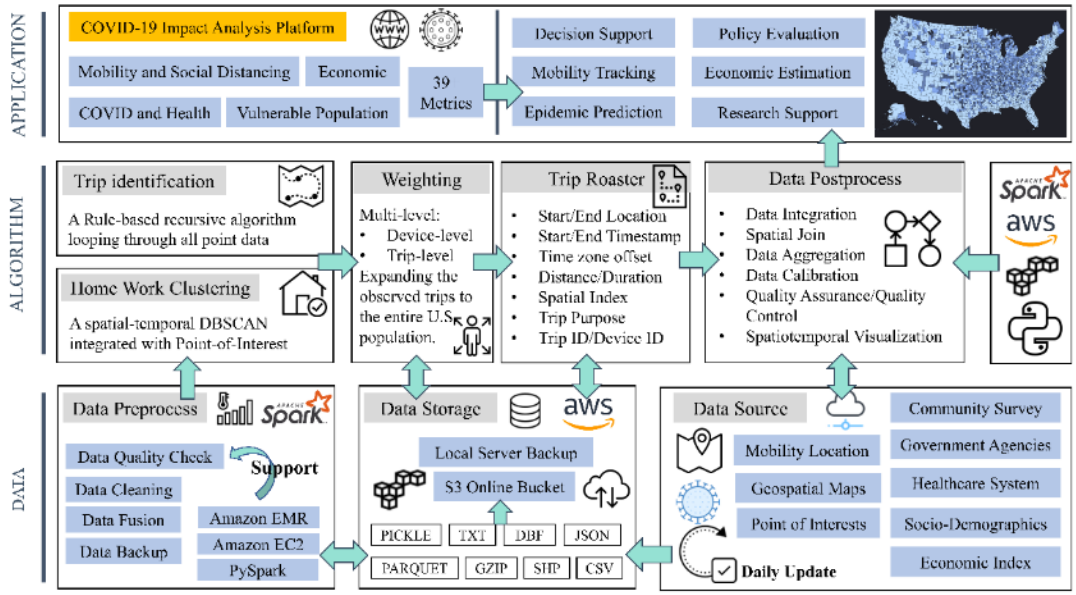


Figure 3-7 Analytical framework for modeling travel demand during COVID-19

The proposed big-data-driven analytical framework is illustrated in **Figure *3-7***. A data panel of emerging MDLD representing person movements for the entire US is developed, incorporating over 20 million anonymous individuals daily (over 150 million monthly) active mobile devices. The execution of the analytics on all the MDLD is conducted via cloud computing and service solutions. 1.2–3.4 billion data records are generated daily, which is impractical to process with a regular computing setup. Hence, a cloud-based distributed cluster-computing framework (Spark) built on EMR (Amazon Elastic MapReduce) is employed to address the computation problem. Using a cluster with a configuration of one c5.18xlarge master node and ten

c5.12xlarge cores nodes, the average computation time of deriving the daily weighted trip rosters is reduced from 36 h on a local server (192 GB Memory, 28 cores, Intel® Xeon® CPU E5-2697 v3 @ 2.60 GHz) to 1 h on the EMR cluster.

Results and computational algorithms have been validated with a variety of independent datasets such as NHTS and ACS, and peer-reviewed by an external expert panel in a US Department of Transportation Federal Highway Administration's Exploratory Advanced Research Program project. Moreover, this study compared national mobility trends during COVID-19 calculated by the proposed method and by other companies including Apple, Google, and SafeGraph (**Figure 3-8**), and found high consistency. Validation results suggest the data sources and algorithms used in this study are convinced to represent the population-level mobility trend in the US.

The applications of the framework are profound. Ingesting over 60 TB of data and utilizing over 75,000 CPU hours of computation, this framework provides timely ground-truth information on how people in the US move during the COVID-19 outbreak. The mobility informatics are analyzed daily at the national, state, and county levels in the US and made available to the general public via the COVID-19 impact analysis platform (https://data.covid.umd.edu/). The outputs of the models and analytics can help agencies monitor and improve their policy effectiveness, as well as enable cross-disciplinary research and collaborations.

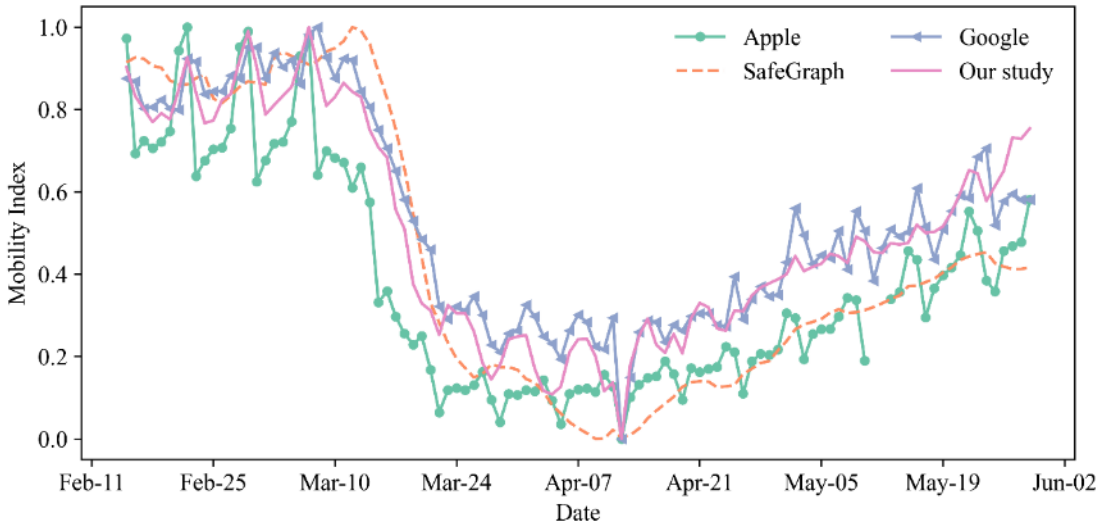


Figure 3-8 National mobility trends during the pandemic (multi-source)

Note: Apple features daily changes in mobility by transport modes including driving, transit, and walking. Google features daily changes in mobility by categories of places such as retail and recreation, groceries and pharmacies, parks, transit stations, workplaces, and residential. Similarly, SafeGraph features visitors by categories of places in more detail and provides free access to the national footprint dataset. These mobility indexes are computed based on different algorithms and thus are with different magnitudes. For the convenience of comparison, this study averages each data source into one curve and then normalizes each curve into the range [0, 1].

4 Chapter 4: Revisiting travel demand and underlying factors

MDLD contain population-representative, fine-grained travel demand information, facilitating opportunities to validate well-established relations between travel demand and underlying factors from a big data perspective. This chapter extensively compares a variety of EML models and interpretation techniques to comprehensively understand the relations between big-data-driven travel demand and related factors. Models from linear regressions, tree-based estimators, and neural networks are compared. The census block group (CBG)-level population inflow extracted from MDLD across the contiguous US is used as the proxy of travel demand. Various exogenous factors are considered, including socioeconomics, demographics, land use, and CBG attributes. Specifically, three research questions are explored:

1) How do EML models perform in estimating MDLD-based travel demand?

2) How should EML models be interpreted and what are the main findings?

3) Do interpretation outcomes hold consistently across different models, hyperparameter configurations, and interpretation techniques?

This study is important as it provides critical evidence on relations between travel demand and underlying factors, which is a rudimentary question in the transportation domain. The fine-tuned model can serve as an upstream component of customary travel demand models, with the capacity to complement or replace statistical models in estimating trip production or attraction. Moreover, aided by advanced interpretation techniques, this study provides versatile functions for visualizing and understanding complex nonlinearities and interactions between travel

demand and external environments, offering new perspectives for developing hand-in tools for urban transportation modelers.

4.1 Variable and data description

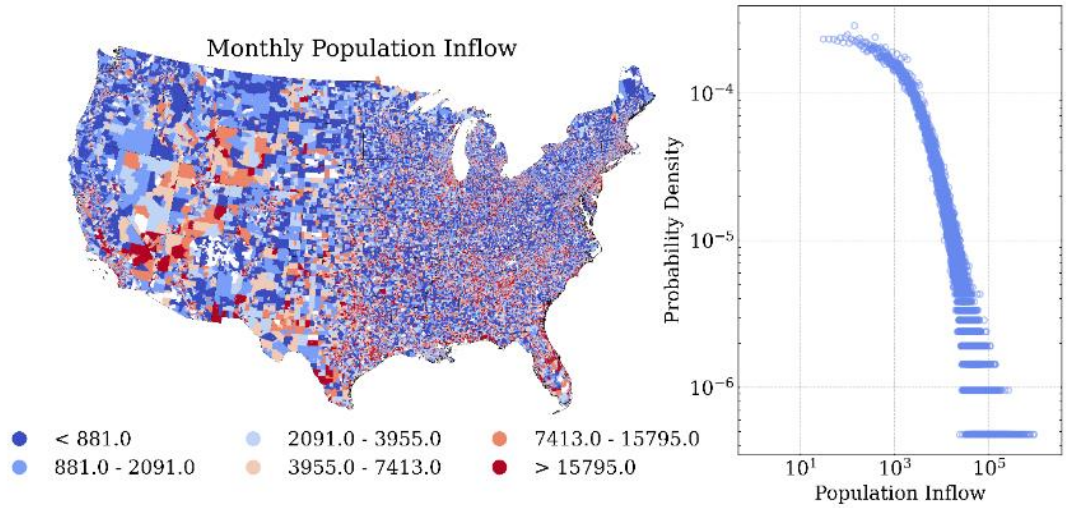
4.1.1 Prediction target

In this section, the national CBG-level number of trip attractions identified from MDLD was employed as the prediction target. Here it is termed population inflow. It is noteworthy that the MDLD-based population inflow is similar to but not the same as the trip attraction in conventional four-step travel demand models since MDLD cannot record all movements of the whole population. A particular weighting process is required to extrapolate samples to the population [71]. Indeed, the MDLD-based population inflow is more similar to the "urban vitality" [178] in the urban planning domain. However, since urban vitality is a broader concept that can be measured by diverse data sources [178] such as transactions, nighttime light, and social media check-ins, this study did not name it as urban vitality for the sake of clarity.

Regarding temporal coverage, this study used the monthly total population inflow during September 2021 as the prediction target. The reasons for choosing September 2021 are twofold: First, human mobility has almost recovered to the pre-pandemic level by then, eliminating the irregular mobility changes triggered by the pandemic [179]. Second, the average temperature for September 2021 in the contiguous US was 67.8°F, which was suitable for travel. Hence, abnormal travel patterns caused by inclement winter weather were excluded. For spatial coverage, this study considered CBGs located in the contiguous US, except those with an average

daily inflow below 1. Eventually, 210,324 CBGs were included, among which 189,292 CBGs were used as the training/validation set (~90%) with 5-fold cross-validation, while 21,032 were set as the testing set (~10%) for model evaluation.

The spatial distribution of the population inflow is exhibited in **Figure *4-1***. A pronounced spatial clustering can be found, which is plausible considering population spillover and urban agglomeration. A pronounced spatial clustering can be found, which is plausible considering population spillover and urban agglomeration. To address the spatial dependence, CBGs' coordinates and state dummy variables were included in the factors. Another visible issue is the over-dispersion distribution of CBG-level population inflow (**Figure *4-1*** (b) and **Figure *4-2***). The statistics of population inflow in **Table *4-2*** also suggest a great gap between the mean and median, as well as a high standard deviation.



(a) (b)

Figure 4-1 Spatial distribution of CBG-level population inflow (a) and its log-transformed distribution plot (b)

The over-dispersion pattern necessitates a data transformation for population inflow and other related features before model building. Specifically, a Box-Cox

transformation is applied to transform the population inflow and a z-score normalization is used to normalize features:

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \quad (4\text{-}1)$$

$$y'_i = \begin{cases} \frac{y_i^{\gamma} - 1}{\gamma}, \text{if } \gamma \neq 0; \\ \log(y), \text{if } \gamma = 0. \end{cases} \quad (4\text{-}2)$$

where $x_{i,j}$ is the j^{th} feature of the i^{th} CBG; $x'_{i,j}$ is its normalized value; μ_j is the mean of the j^{th} feature and σ_j is the standard deviation of the j^{th} feature; y_i is the population inflow of the i^{th} CBG; y'_i is its transformed value; γ is the parameter that results in the best approximation of a normal distribution; μ_j, σ_j, γ are all estimated only using training data to avoid data leakage.

Figure *4-2* shows the transformed population inflow, which now presents a clear bell-shaped pattern. Note that the parameters used in z-score normalization and Box-Cox transformation did not include data from testing sets to avoid information leaks. In addition, transformed data were only used when training the model, whose purpose was to enhance model accuracy and efficiency, as the original over-dispersed data could easily lead to an unsmooth and slow learning process. The transformed data were rolled back to the original value for better interpretation.

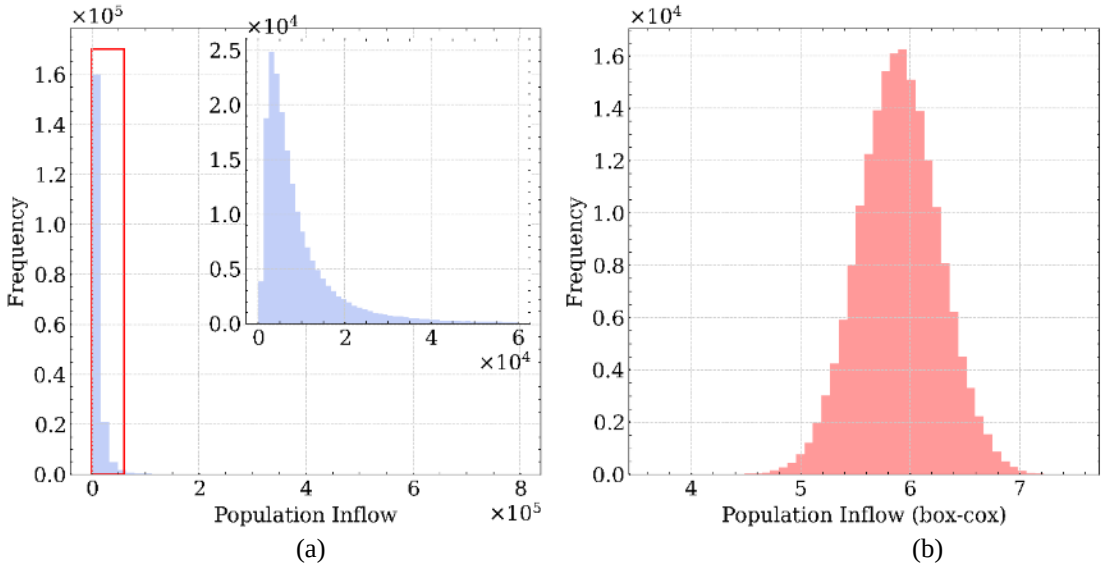


Figure 4-2 Distribution of CBG-level population inflow before (a) and after (b) Box-Cox transformation

4.1.2 Feature engineering

This section followed the traditional travel demand theory to select features, covering CBG-level socioeconomics, demographics, built environment, and spatial factors (**Table *4-2***). Socioeconomics and demographics were from 5-year (2015–2019) ACS. Partisanship was from the 2020 presidential election result provided by the MIT election lab [180]. Since there was no nationwide unified dataset with detailed and up-to-date land use information, this study used POIs as proxies for land use. POI information was from SafeGraph [181], a company that provides POI-level foot traffic across ~4.4 million POIs in the US. Each POI is labeled with a North American Industry Classification System (NAICS) code to describe its land-use type and a foot traffic volume to describe the number of visits (**Table *4-1***). To simplify the features, this section ranked POI types by their monthly foot traffic and selected the top 20 types as the final POI features, which covered over 95% of the total visits. The

summary and statistics of the features used in the model building can be found in **Table *4-2***. It is noteworthy that feature dependency does exist in this study, especially among the total population and the number of different types of POIs. Although ML models can handle such a problem when making predictions, they should not be neglected when interpreting models as many interpretation techniques assume independencies among features [94, 105, 182]. This study extensively compares interpretation techniques to examine whether the feature dependency in this dataset could distort the outcomes.

Table 4-1 Summary of national POI information

NAICS Code	Description	Monthly foot traffic (Million)	%	POI Count (Million)
72	Accommodation and Food Services	2812.293	30.036	1.055
44	Retail Trade	1882.336	20.104	1.267
45	Retail Trade	1188.949	12.698	0.434
71	Arts, Entertainment, and Recreation	912.664	9.748	0.373
62	Health Care and Social Assistance	754.258	8.056	1.074
61	Educational Services	669.485	7.150	0.197
81	Other Services (except Public Administration)	497.159	5.310	1.353
52	Finance and Insurance	122.473	1.308	0.480
48	Transportation and Warehousing	92.516	0.988	0.040
51	Information	91.208	0.974	0.093
92	Public Administration	65.957	0.704	0.072
53	Real Estate and Rental and Leasing	45.466	0.486	0.187
42	Wholesale Trade	39.654	0.424	0.057
31	Manufacturing	36.284	0.388	0.046
49	Transportation and Warehousing	35.846	0.383	0.059
54	Professional, Scientific, and Technical Services	31.214	0.333	0.111
0	Unknown	24.732	0.264	0.095
56	Administrative and Support and Waste Management and Remediation Services	21.098	0.225	0.060
33	Manufacturing	10.439	0.111	0.015
23	Construction	9.475	0.101	0.020
55	Management of Companies and Enterprises	7.254	0.077	0.009
22	Utilities	7.240	0.077	0.009
32	Manufacturing	4.387	0.047	0.006
11	Agriculture, Forestry, Fishing, and Hunting	0.402	0.004	0.001
21	Mining, Quarrying, and Oil and Gas Extraction	0.200	0.002	0.000

Table 4-2 Summary of CBG-level target and features for EML models

	Description	Mean	St.d.	Med.
Prediction Target				
Population Inflow	Monthly total number of trips with destinations to a specific CBG	10057.85	12681.42	6426
Features				
Demographics				
Total Population	Total population, in 10^4 persons	0.151	0.103	0.128
Population Density	Population density, in 10^4 persons/sq. mile	0.611	1.461	0.255
Urbanized Population	% urbanized areas population	68.398	45.222	100.000
White	% Non-Hispanic Whites	62.851	31.238	72.279
Hispanic	% Hispanics/Latinos	16.236	22.529	6.472
African American	% African Americans	13.459	22.838	2.956
Asian	% Asians	4.564	9.485	0.713
Age 18-44	% residents aged between 18 and 44	34.825	12.068	33.548
Age 45-64	% residents aged between 45 and 64	26.602	8.232	26.574
Age >65	% residents 65 and over	16.991	10.175	15.493
Male	% male	49.068	6.313	49.123
Socioeconomics				
High Educated	% residents with education attainment equal to/higher than Bachelor	30.121	20.665	24.940
Unemployed Rate	% the total labor force that is unemployed	5.756	6.169	4.167
Median Income	Median household income (in 2019 Inflation-Adjusted Dollars), in $\$10^4$/household	6.863	3.672	6.125
Rent to Income	% household income a tenant will need for the monthly median gross rent	30.860	9.261	30.873
Central	1: Central (85.43%); 2: Outlying (6.81%); 3: Rural (7.76%)	-	-	-
Democrat	% Democrats in 2020 presidential candidate vote totals	48.118	17.750	48.441
Poverty	% households below the national poverty level	13.987	12.814	10.448
Work at home	% work-from-home commuters among workers 16 years and over	4.995	5.667	3.540
Built environment				
POI Count	# POIs (from SafeGraph)	31.397	47.847	18.000
Administration	(NAICS 56) # POIs of Administration	0.167	0.529	0.000
Manufacture	(NAICS 31-33) # POIs of Manufacture	0.488	1.363	0.000
Wholesale Trade	(NAICS 42) # POIs of Wholesale Trade	0.368	1.081	0.000
Real Estate	(NAICS 53) # POIs of Real Estate	0.832	2.262	0.000
Public	(NAICS 92) # POIs of Public	0.348	0.970	0.000
Information	(NAICS 51) # POIs of Information	0.399	1.121	0.000
Transportation	(NAICS 48-49) # POIs of Transportation	0.492	1.117	0.000
Finance	(NAICS 52) # POIs of Finance	2.145	4.515	1.000
Education	(NAICS 61) # POIs of Education	0.992	1.541	1.000
Health Care	(NAICS 62) # POIs of Health Care	4.281	10.248	1.000
Recreation	(NAICS 71) # POIs of Recreation	1.534	2.592	1.000
Retail Trade	(NAICS 44-45) # POIs of Transportation	7.132	13.848	3.000
Accommodation &Food	(NAICS 72) # POIs of Accommodation & Food	4.753	9.811	2.000

Spatial factors				
Area	The area of the CBG, in sq. mile	13.816	99.897	0.525
Longitude	The county longitude	-91.044	15.308	-86.870
Latitude	The county latitude	37.920	4.898	38.981
CTFIPS	The county FIPS code	-	-	-
STFIPS	The state FIPS code	-	-	-

4.2 Explainable machine learning (EML) models

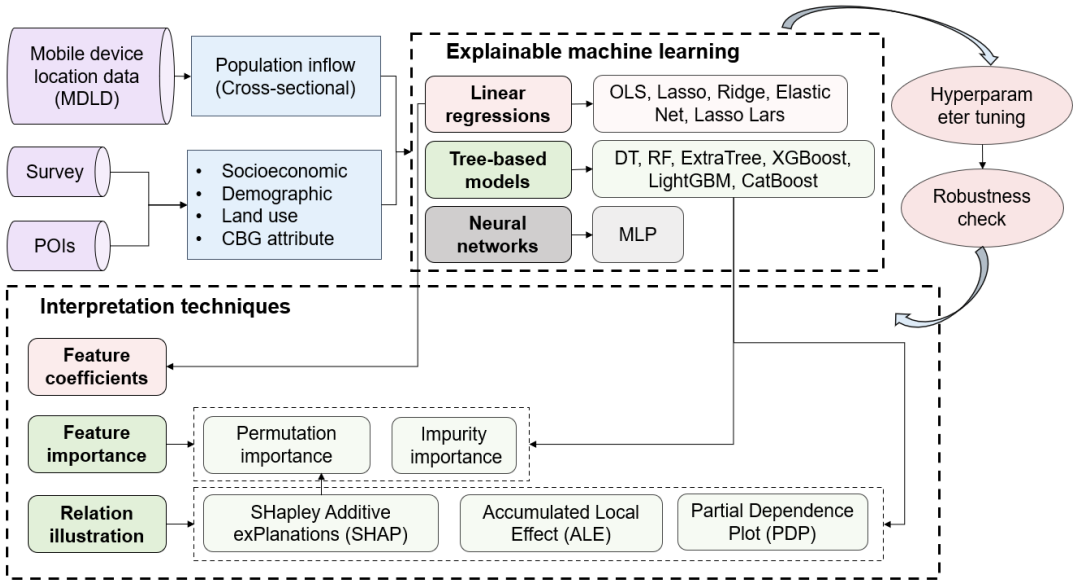


Figure 4-3 Analytical framework for EML models comparison

The main scope of this section is to extensively compare popular EML models for travel demand estimation. The methodology framework is shown in **Figure *4-3***. First, the target, which is the cross-sectional population inflow extracted from MDLD, in conjunction with underlying factors, was fed into a range of EML algorithms. These EML models include linear regressions, tree-based models, and neural networks. Second, the hyperparameters of these models were tuned to achieve the best performance. The fine-tuned models were then interpreted via a set of interpretation techniques. For linear regressions, feature coefficients were extracted. For tree-based models, feature importance was computed to represent the feature contribution to the model prediction. Meanwhile, relation illustration techniques were employed to

delineate learned relations among population inflow and underlying factors. Last, interpretation outcomes were compared among different interpretation techniques to choose the best option and compared among models with different hyperparameter settings to check the interpretation robustness.

A variety of EML models were compared. The summary of models is outlined in **Table *4-3***. In the following section, a brief introduction to EML models and interpretation techniques is presented. Note that since all these methods are well-established and widespread, this chapter does not meticulously introduce details but only provides a rough description. In addition, only one simple neural network is included since this chapter emphasizes EML, among which linear regressions and tree-based models are the mainstream.

Table 4-3 EML models for cross-sectional population inflow estimation

Model	Description
Linear regression	
Linear	Ordinary least squares regression.
Lasso	Linear regression trained with the L1-norm as the regularizer.
Ridge	Linear regression trained with the L2-norm as the regularizer.
Elastic Net	Linear regression trained with combined L1 and L2-norm regularizer.
Lasso Lars	Lasso model fitted with least angle regression.
Neural network	
MLP	Multi-layer perceptron regression, aka feedforward FNN.
Tree-based models	
Decision Tree	Single decision tree.
RF [95]	Random Forest, a popular framework for bagging trees.
Extra Trees	Similar to RF but has extremely randomized trees.
XGBoost [172]	Regularizing gradient boosting trees.
LightGBM [183]	Fast gradient boosting trees with advanced techniques for computational enhancement.
CatBoost [184]	Unbiased gradient boosting trees that attempt to solve for categorical features.

4.2.1 Linear regression

OLS regression aims to find a linear approximation to minimize the residual sum of squares between the target and the prediction. Other variants of linear regressions are

built based on OLS but include some regularization techniques to avoid overfitting [185]. Specifically, Lasso regression [186] adds the absolute value of coefficients (L1-norm) as the penalty term in the objective function (Eq. (*4-3*)). Ridge regression [185] adds the squared value of coefficients (L2-norm) as the penalty term (Eq. (*4-4*))). Elastic Net regression [187] adds both L1 and L2-norm regularization (Eq. (*4-5*))). Lasso Lars regression is a Lasso regression implemented using the least angle regression (LARS) algorithm, which is more computationally efficient for big data [188]. The objective functions of these variants are shown as follows:

$$O_{Lasso}(\boldsymbol{\beta}) = \|\boldsymbol{X\beta} - \boldsymbol{Y}\|_2^2 + \alpha\|\boldsymbol{\beta}\|_2^2 \quad (4\text{-}3)$$

$$O_{Ridge}(\boldsymbol{\beta}) = \frac{1}{2N}\|\boldsymbol{X\beta} - \boldsymbol{Y}\|_2^2 + \alpha\|\boldsymbol{\beta}\|_1 \quad (4\text{-}4)$$

$$O_{Elastic\ Net}(\boldsymbol{\beta}) = \frac{1}{2N}\|\boldsymbol{X\beta} - \boldsymbol{Y}\|_2^2 + \alpha\delta\|\boldsymbol{\beta}\|_1 + \frac{\alpha(1-\delta)}{2}\|\boldsymbol{\beta}\|_2^2 \quad (4\text{-}5)$$

where $\boldsymbol{X}$ is the matrix of features; $\boldsymbol{Y}$ is the vector of targets; $\boldsymbol{\beta}$ is the vector of coefficients; α is the weight of the regularizer; N is the number of samples; δ is the weight of two regularizers.

4.2.2 Single decision tree

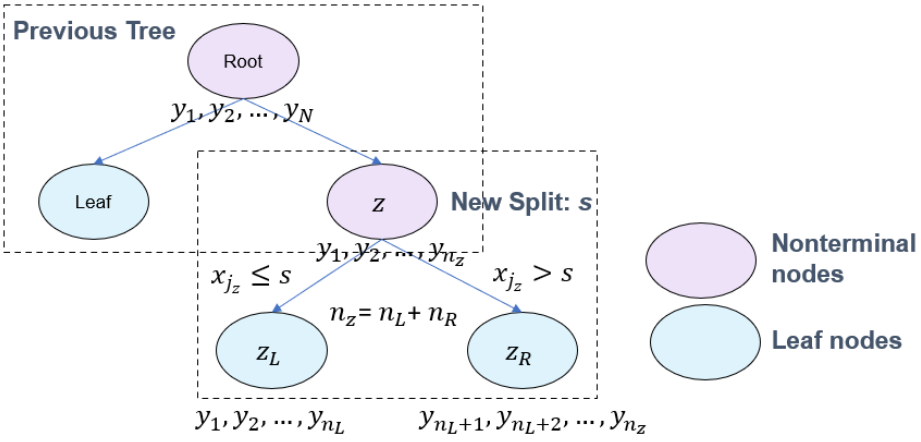


Figure 4-4 An illustration of a new split in a single decision tree

A single decision tree is a non-parametric supervised learning method that predicts the target by learning decision rules inferred from features. Different algorithms have been well developed [189] for identifying the optimal split point for each feature (i.e., the decision rules) that will yield the largest information gain. This study uses the CART algorithm to build a single decision tree. The decrease of impurity is used to define the information gain, which can be measured by various criteria such as the Gini index for classification, and squared error for regression. Assume a node *z* is split into two child nodes at point *s*. Then, the decrease of squared error impurity $\Delta G(z, s)$ after the split *s* is computed as [94]:

$$\Delta SE(z, s) = \frac{n_L n_R}{n_L + n_R}(\bar{y}_L - \bar{y}_R)^2 \tag{4-6}$$

where $\bar{y}_L$ and $\bar{y}_R$ are the target means in the left and right child nodes; n_L and n_R are the number of samples sent to the left and right child nodes; $\Delta SE(z, s)$ is the decrease in squared error as a result of the split *s* at node *z* (see **Figure *4-4*** for an illustration), which is the difference in sample variances before and after the split.

Proof: The squared error impurity in Eq. (*4-6*) is the same as the difference in sample variances before and after the split. Let $N = n_L + n_R$:

$$\begin{aligned} &\Delta SE(z, s) = \textstyle\sum_{i=1}^{N}(y_i - \bar{y})^2 - \sum_{i=1}^{n_L}(y_i - \bar{y}_L)^2 - \sum_{i=n_L+1}^{N}(y_i - \bar{y}_R)^2 = \\ &(\bar{y}_L - \bar{y})\textstyle\sum_{i=1}^{n_L}(2y_i - \bar{y}_L - \bar{y}) + (\bar{y}_R - \bar{y})\sum_{i=n_L+1}^{N}(2y_i - \bar{y}_R - \bar{y}) = n_L(\bar{y}_L - \bar{y})^2 + \\ &n_R(\bar{y}_R - \bar{y})^2 = n_L\frac{n_R^2}{N^2}(\bar{y}_L - \bar{y}_R)^2 + n_R\frac{n_L^2}{N^2}(\bar{y}_L - \bar{y}_R)^2 = \frac{n_L n_R}{n_L + n_R}(\bar{y}_L - \bar{y}_R)^2 \end{aligned} \tag{4-7}$$

Note: Some models used $n_L(\bar{y}_L - \bar{y})^2 + n_R(\bar{y}_R - \bar{y})^2$ as impurity (e.g., CatBoost [184]). The proof documents that they are indeed the same as Eq. (*4-6*).

4.2.3 Tree ensembles: bagging and boosting

The single tree may suffer from limitations such as overfitting and unstable tree structure, which can be well addressed by the ensemble method. The ensemble method combines predictions of a set of single trees to improve holistic generalizability and robustness. The ensemble method can be divided into bagging and boosting [190]. In bagging methods, individual trees are built independently. Prediction outcomes of all single trees are averaged to obtain the final output. In boosting methods, single trees are built sequentially. The new tree is built upon previous trees to optimally reduce the current bias. Assuming $\boldsymbol{X}$ is the set of features, the formulation of a tree ensemble can be expressed as [94]:

$$F_K(\boldsymbol{X}) = \sum_{k=1}^{K} h_k(\boldsymbol{X}), h_k \in \varkappa \tag{4-8}$$

where $F_K(\boldsymbol{X})$ is the function of the ensemble estimator combing K single trees; $h_k(.)$ is the additive function of the k^{th} single tree; $\varkappa$ is the functional space.

Bagging trees can be further distinguished by their ways of involving randomness. RF is one of the most popular bagging trees [95]. In RF, two types of randomness are injected. First, each tree is built from a bootstrap sample drawn from the training set (sampling by rows). Second, the best split at each node is found from a random subset of features (sampling by columns). In ExtraTree, randomness goes one step further in the way splits are computed [191]. Instead of following Eq. (*4-6*) to find the best split, splits in ExtraTree are generated randomly and the best of these random splits is selected as the final splitting rule.

Boosting trees share the same structure as bagging trees. The difference arises from how to update every single tree. Unlike the bagging method which learns all trees simultaneously, the boosting method modifies the tree according to the last iteration to place higher weights on samples that are more difficult to predict. For each iteration, a new tree $h_t(\boldsymbol{X})$ is added to minimize the objective function $L(.)$ given the previous ensemble $F_{t-1}(\boldsymbol{X})$. After the t^{th} iteration, the ensemble estimator is expressed by Eq. (*4-9*). The new tree, as well as its first-order Taylor approximation, is expressed by Eq. (*4-10*):

$$F_t(\boldsymbol{X}) = F_{t-1}(\boldsymbol{X}) + \zeta h_t(\boldsymbol{X}), \zeta \in (0,1) \tag{4-9}$$

$$h_t(\boldsymbol{X}) = \underset{h}{\operatorname{argmin}}\, L\big(\boldsymbol{Y}, F_t(\boldsymbol{X})\big) \cong \underset{h}{\operatorname{argmin}}\, h(\boldsymbol{X})\left[\frac{\partial L\big(\boldsymbol{Y}, F(\boldsymbol{X})\big)}{\partial F(\boldsymbol{X})}\right]_{F=F_{t-1}} \tag{4-10}$$

where $\boldsymbol{X}$ is the matrix of features; $\boldsymbol{Y}$ is the vector of targets; $F_t(.)$ is the function of the ensemble containing *t* single trees; ζ is the learning rate to prevent overfitting. Since the last term in Eq. (*4-10*) is the partial derivative of the objective function, the update of a new tree at each iteration can be viewed as the gradient descent in the functional space. Hence, boosting ensemble decision trees shown in Eq. (*4-10*) is also termed gradient boosting decision trees (GBDT).

4.2.4 Advanced boosting trees

The boosting methods are believed to have higher accuracy since the model automatically adjusts new trees based on previous residuals [172]. However, due to the sequence learning structure, the training process of boosting trees is difficult to parallelize. Meanwhile, the overfitting issue still exists due to the nature of decision trees. Hence, a range of variants has been proposed based on the boosting tree to

enhance the model efficiency and generalizability, among which XGBoost, LightGBM, and CatBoost are the most prevalent.

XGBoost [172]: XGBoost is a regularizing gradient boosting framework for ensemble trees. One significant advantage of XGBoost is that it includes a regularizer in the objective function to constrain the tree complexity and prevent overfitting [172]. Other salient techniques in XGBoost include parallelized tree building, hardware optimization, tree pruning using a level-wise (depth-first) approach, and efficient optimal split finding via a histogram-based algorithm. Specifically, a major reason for the lower efficiency of traditional GBDT lies in its time-consuming process of enumerating all the possible splits for continuous features to find the optimal split. In XGBoost, a histogram-based algorithm, named weighted quantile sketch, is proposed by mapping continuous features into buckets.

LightGBM [183]: A similar histogram-based algorithm is used by LightGBM for optimal split searching. In addition, LightGBM further enhances computational efficiency by utilizing two novel techniques called gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB) [183]. With GOSS, LightGBM excludes a large fraction of samples with small gradients and only uses the rest to train the tree. With EFB, LightGBM bundles mutually exclusive features to reduce the feature size. Meanwhile, sufficient efforts have been made by LightGBM in distributed learning to speed up the model learning process. To improve model accuracy, LightGBM grows a tree leaf-wise (best-first) instead of level-wise. Since the leaf-wise strategy chooses the best split based on its contribution to the global loss

instead of the local loss along a particular branch, it often learns lower-error trees faster than the level-wise strategy, particularly in large datasets [183].

CatBoost [184]: Traditional GBDT represents categorical features with one-hot encoding, which is infeasible for high-cardinality features. CatBoost is hence proposed with an emphasis on handling categorical features [184]. In CatBoost, an ordered target statistic strategy is introduced to convert categories to their target statistics, which is defined as the expected target value conditioned by the category. In addition, CatBoost improves the model accuracy by introducing an ordered boosting algorithm to address the prediction shift caused by target leakage present in the current GBDT [184]. Note that to be compatible with different interpretation techniques, this study does not apply the strategy in CatBoost to handle categorical features. Instead, a categorical feature is still represented with one-hot encoding, which is acceptable since there is no high-cardinality feature in this study.

4.3 Interpretation approaches

In this study, linear regressions are simply explained by their coefficients. Tree-based models are interpreted by feature importance and relation illustration. Each of them includes a set of different techniques. The main attributes of these adopted interpretation techniques are summarized in **Table 4-4**, with detailed descriptions reported in the following sections.

Table 4-4 Summary of interpretation techniques

Interpretation approaches	Global/Local	Model-agnostic/ model-specific	Allow feature dependency?	Computational load
Feature importance				
Impurity feature importance	Global	Model-specific	Y	Low

Permutation (shuffle) feature importance	Global	Model-agnostic	N	Median
Permutation (SHAP) feature importance	Global	Model-agnostic	N	High
Relation illustration				
PDP	Global	Model-agnostic	N	Median
ALE	Global	Model-agnostic	Y	Median
SHAP	Global&Local	Model-agnostic	N	High
TreeSHAP	Global&Local	Model-specific	Y	Median (Boosting trees); High (others)

4.3.1 Feature importance

Feature importance measures how much importance the model attaches to a specific feature when predicting the target. Two main methods have been developed to measure the feature importance in tree-based models, including impurity importance and permutation importance. Impurity importance is defined as the impurity decrease that a feature brings to the node where the feature serves as the split point, weighted by the probability of samples reaching that node, and averaged over all trees of the ensemble. In an ensemble containing *K* single trees, the impurity importance of feature x_j is defined as [94, 192]:

$$\mathrm{Imp}(x_j) = \frac{1}{K}\sum_{k=1}^{K}\sum_{z\in\varphi_k} 1(j_z = j)[\frac{n_z}{N}\Delta i(z,s)] \qquad (4\text{-}11)$$

where *z* denotes the z^{th} nonterminal node of the tree φ_k; j_z is the identifier of the feature used for splitting node *z*; $1(.)$ is the indicator function; n_z is the number of samples reaching node *z*; *N* is the total number of samples; $\Delta i(z,s)$ represents the impurity decrease after splitting at the z^{th} node by the split *s*, which is measured by Eq. (*4-11*) (see **Figure *4-4*** for an illustration).

Since the impurity decrease is the same as the information gain, the importance calculated by Eq. (*4-11*) is also named “Gain importance” [172, 183,

193]. Some other methods ignore the gain in Eq. (*4-11*) by directly defining feature importance as the number of times a feature is used to split, such as the "Weight importance" in XGBoost and the "Split importance" in LightGBM. This study mainly focused on the Gain importance, as it is more intuitive in reflecting the feature's contribution to the model.

Unlike the built-in impurity importance, permutation importance is a post hoc model-agnostic interpretation technique that measures the importance as the decrease in model accuracy or, equivalently, the increase in model loss, when the link between the feature and the target is broken. There are many ways to break the link, such as randomly shuffling the feature, entirely removing the feature [184], and treating feature combinations as coalitions in game theory (i.e., SHAP) [193]. Using the shuffling method as an example, the permutation importance of the feature x_j is:

$$\mathrm{Per}(x_j) = \frac{1}{R}\sum_{r=1}^{R}[L\left(\boldsymbol{Y}, F\left(\tilde{\boldsymbol{X}}^{(r,j)}\right)\right) - L(\boldsymbol{Y}, F(\boldsymbol{X}))] \qquad (4\text{-}12)$$

where R is the number of times the feature x_j is shuffled (10 in this study); $F(.)$ is the ensemble function trained by the original feature set $\boldsymbol{X}$; $\tilde{\boldsymbol{X}}^{(r,j)}$ is the corrupted version of the feature set where the feature x_j is shuffled r^{th} times; $L(.)$ is the loss function.

Both permutation importance and impurity importance suffer from some flaws. For example, impurity importance favors high-cardinality categorical features, which, fortunately, is not a concern in this study since there is no high-cardinality feature included. As for permutation importance, it is computationally expensive for large datasets because there are several repeats of shuffling and predicting for each feature (see **Table *4-6*** for computation time). In addition, permutation importance

can result in misleading conclusions when features are highly correlated [192]. This study is intended to compare both types of importance in order to explore their differences in interpreting the same model.

4.3.2 Partial dependence plot (PDP) and Accumulated Local Effect (ALE)

PDP is a post hoc model-agnostic interpretation technique that shows the dependence between the target and the feature, marginalizing the effects of other features. For a training set including N samples, the partial dependence of the j^{th} feature is [94]:

$$\psi_j(\boldsymbol{z}_j) = E_{\boldsymbol{X}_{\setminus\{j\}}}\left[F(\boldsymbol{z}_j, \boldsymbol{X}_{\setminus\{j\}})\right] \cong \frac{1}{N}\sum_{i=1}^{N} F(\boldsymbol{z}_j, \boldsymbol{X}_{i,\setminus\{j\}}) \tag{4-13}$$

where $F(.)$ is the approximation function of the estimator; $\boldsymbol{z}_j$ is the vector of the grid value of the j^{th} feature (i.e., $\boldsymbol{z}_j = \{\min(\boldsymbol{x}_j), \min(\boldsymbol{x}_j) + \tau, \ldots, \min(\boldsymbol{x}_j) + k\tau \ldots, \max(\boldsymbol{x}_j)\}$, where τ is the step size); $\boldsymbol{X}_{\setminus\{j\}}$ is the complement subset of $\boldsymbol{x}_j$.

PDP assumes that the feature $\boldsymbol{x}_j$ is independent of all features in $\boldsymbol{X}_{\setminus\{j\}}$, which would lead to unrealistic combinations between $\boldsymbol{x}_j$ and $\boldsymbol{X}_{\setminus\{j\}}$ if they were dependent. One solution is to calculate ALE based on conditional expectation [182]:

$$\tilde{\tilde{\psi}}_j\left(z_j^{(k)}\right) = \sum_{s=1}^{k}\frac{1}{n_j^{(s)}}\sum_{i=1}^{n_j^{(s)}}\left[F\left(z_j^{(s)}, \boldsymbol{X}_{i,\setminus\{j\}}\right) - F\left(z_j^{(s-1)}, \boldsymbol{X}_{i,\setminus\{j\}}\right)\right] \tag{4-14}$$

$$\tilde{\psi}_j\left(z_j^{(k)}\right) = \tilde{\tilde{\psi}}_j\left(z_j^{(k)}\right) - \sum_{k=1}^{K}\tilde{\tilde{\psi}}_j\left(z_j^{(K)}\right) \tag{4-15}$$

where $\boldsymbol{z}_j$ is the vector of the grid value of the j^{th} feature and $z_j^{(k)}$ is its k^{th} value; K is the size of the vector $\boldsymbol{z}_j$; $n_j^{(s)}$ is the number of samples located between $z_j^{(s-1)}$ and

$z_j^{(s)}$; $\boldsymbol{X}_{\setminus\{j\}}$ is the complement subset of $\boldsymbol{x}_j$; $\tilde{\tilde{\psi}}_j\left(z_j^{(k)}\right)$ is the uncentered ALE of the j^{th} feature and $\tilde{\psi}_j\left(z_j^{(k)}\right)$ is the corresponding centered ALE. The value of the $\tilde{\psi}_j\left(z_j^{(k)}\right)$ can be interpreted as the effect of the feature at a certain value on the prediction compared to the corresponding average prediction.

4.3.3 SHapley Additive exPlanations (SHAP)

Both PDP and ALE belong to global interpretation techniques. One main limitation of global techniques is that they may hide heterogeneous effects since only average effects are computed. Also, outliers may significantly twist the curve shape and create unintentionally misleading interpretations [105]. One possible solution is to use local interpretation methods such as the SHAP. SHAP is a local model-agnostic interpretation method that connects local interpretations with Shapley values – a value from coalitional game theory – under a solid theoretical foundation [109]. With SHAP, global and local interpretations are unified since local Shapley values are the "atomic unit" of global interpretations [194]. Meanwhile, SHAP contributes to a new permutation-based way to measure feature importance, i.e., SHAP importance [193].

The Shapley value of the j^{th} feature is its contribution to the prediction compared to the average, weighted and summed across all feature combinations:

$$\phi_j = \frac{1}{|\boldsymbol{X}|} \sum_{\boldsymbol{S} \subseteq \boldsymbol{X}_{\setminus\{j\}}} \binom{|\boldsymbol{X}| - 1}{|\boldsymbol{S}|}^{-1} \left[v\left(\boldsymbol{S} \cup \{\boldsymbol{x}_j\}\right) - v(\boldsymbol{S})\right] \tag{4-16}$$

$$v(\boldsymbol{S}) = \int F(\boldsymbol{X})\, d\mathbb{P}_{x \notin \boldsymbol{S}} - E_{\boldsymbol{X}}(F(\boldsymbol{X})) \tag{4-17}$$

where $\boldsymbol{X}$ is the whole set of features and $\boldsymbol{S}$ is one of its subsets without $\boldsymbol{x}_j$ (i.e, the coalition); $((|\boldsymbol{X}| - 1)¦(|\boldsymbol{S}|))$ denotes the number of combinations for choosing $|\boldsymbol{S}|$

features from the feature set without $\boldsymbol{x}_j$; $v(\boldsymbol{S})$ is the prediction based on the feature subset $\boldsymbol{S}$ marginalizing over features that are not included in set $\boldsymbol{S}$; $v(\boldsymbol{S} \cup \{j\}) - v(\boldsymbol{S})$ denotes how much the j^{th} feature contributes to the prediction after joining the coalition $\boldsymbol{S}$; $F(.)$ is the approximation function of the estimator.

Example: If $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}\}, j = 3$, then $\boldsymbol{S} \subseteq \boldsymbol{X}_{\setminus\{j\}} = \{\emptyset, \{\boldsymbol{x_1}\}, \{\boldsymbol{x_2}\}, \{\boldsymbol{x_1}, \boldsymbol{x_2}\}\}$, $\phi_3 = \frac{1}{3}[(v(\boldsymbol{x}_3) - v(\emptyset)) + \frac{1}{2}(v(\boldsymbol{x_1}, \boldsymbol{x_3}) - v(\boldsymbol{x_1})) + \frac{1}{2}(v(\boldsymbol{x_2}, \boldsymbol{x_3}) - v(\boldsymbol{x_2})) + (v(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}) - v(\boldsymbol{x_1}, \boldsymbol{x_2}))]$.

Shapley value is a sample-wise measure, which means for each CBG, there exists a set of Shapley value $\phi_{1,2,\ldots,J}^{(n)}$ describing how much each feature contributes to its prediction compared with the average. After computing all individual Shapley values, the global SHAP importance of the j^{th} feature is obtained by averaging the absolute Shapley values across all samples (Eq. (*4-18*)).

$$\text{SHAP}(x_j) = \frac{1}{N}\sum_{n=1}^{N} |\phi_j^{(n)}| \qquad (4\text{-}18)$$

Another pronounced advantage of SHAP lies in its well-designed algorithm to specify interaction effects. The interaction effect of two features $\boldsymbol{x}_j$ and $\boldsymbol{x}_r$ $(j \neq r)$ can be expressed by Eq. (*4-19*). Meanwhile, the SHAP interaction plot of the j^{th} feature can be easily achieved by plotting the set of points $\{(x_j^{(n)}, \phi_{j,r}^{(n)})\}_{n=1}^{N}$.

$$\phi_{j,r} = \frac{\sum_{\boldsymbol{S} \subseteq \boldsymbol{X}_{\setminus\{j,r\}}} \binom{|\boldsymbol{X}|-2}{|\boldsymbol{S}|}^{-1} [v(\boldsymbol{S} \cup \{j,r\}) - v(\boldsymbol{S} \cup \{j\}) - v(\boldsymbol{S} \cup \{r\}) + v(\boldsymbol{S})]}{2(|\boldsymbol{X}| - 1)} \qquad (4\text{-}19)$$

where the notations are the same as those in Eqs. (*4-16*) and (*4-17*).

Although SHAP has become a prevalent interpretation technique in recent years, one of its main limitations is the high computational cost. To this end,

TreeSHAP, a fast, tree-specific variant of SHAP, was proposed [193]. TreeSHAP can reduce the computational complexity from $O(K \cdot L \cdot 2^J)$ to $O(K \cdot L \cdot D^2)$, where K is # trees, L is the maximal # leaves, J is # features, and D is the maximal depth of trees. Moreover, TreeSHAP defines the $v(.)$ in Eq. (*4-17*) using conditional instead of marginal expectation, which helps address the feature dependency issue in permutation-based methods. In this study, most of the boosting trees were interpreted via TreeSHAP. However, since bagging trees tend to be very deep, the computational load of interpreting bagging trees was still heavy, even using TreeSHAP. Thus, only boosting trees were interpreted via the SHAP method in this study.

4.4 Experiment settings

Since ML models are sensitive to their hyperparameters, this study tunes all ML models within the same budgets (i.e., 50 trials) to find their best configurations. It would be extremely time-consuming if the grid search method is employed. To enhance the tuning efficiency, hyperparameter optimization is conducted via random search using the successive halving algorithm (SHA) [195]. SHA is a multi-armed bandit algorithm to identify the best one among multiple trials and perform principled early stopping on those less-promising ones. The key advantage of SHA is that it does not need to evaluate a trial until it completes all epochs. The less promising trials can be identified early and stopped after partially running. Then more computational resources can be allocated to the promising trials. An example of a hyperparameter tuning process is illustrated in **Figure *4-5*** and the best parameters are listed in **Table *4-5***. In general, SHA begins with all trials in the base rung and loops as follows:

1) Allocate a budget r_i (e.g., one epoch) to a set of trials in a given rung *i*.

2) Evaluate the performance of all trials using the validation set after using up allocated budgets. Promote the top $\lfloor n_t \eta^{-i} \rfloor / \eta$ of trials to the next rung, where η is the reduction rate and n_t is the number of trials.

3) Increase the budget per test to $r_i \eta^{i+s_e}$ for the next rung and repeat until only one trial remains, where s_e is the minimum early-stopping rate.

The mean absolute percentage error (MAPE) is used as the loss function. Other widely used metrics including mean absolute error (MAE), root mean square error (RMSE), and R^2 are used to evaluate model performance on the testing dataset. All models are tuned with 5-fold cross-validation. Several callbacks are applied to monitor and adjust the model during the training procedure. The learning rate is reduced by a factor of 10 when the validation loss does not improve for specific iterations. Early stopping is used to determine the optimal number of iterations.

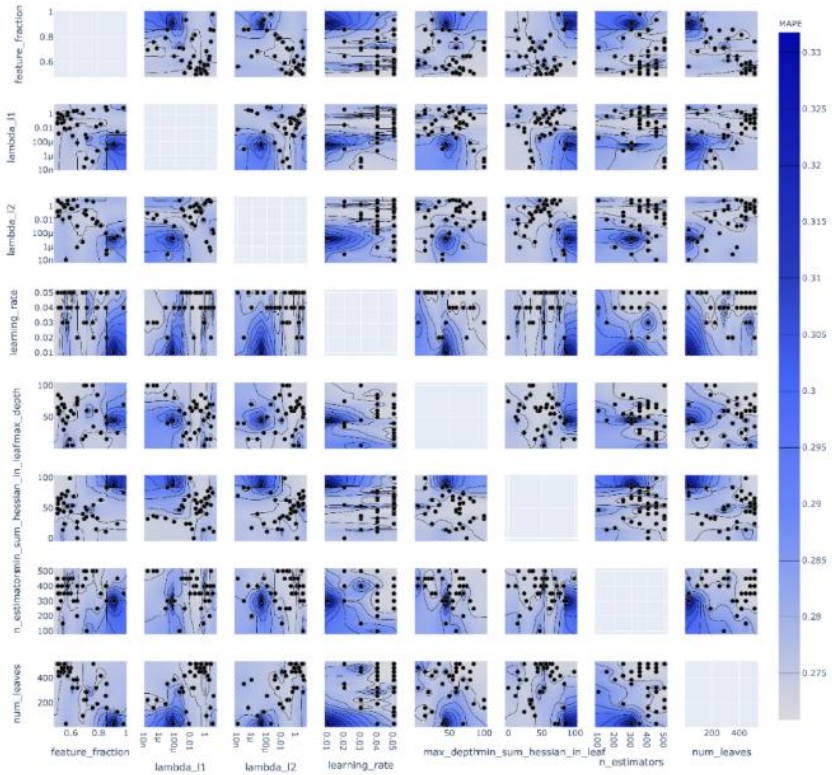


Figure 4-5 Contour plot of hyperparameter tuning for LightGBM

Table 4-5 Hyperparameters tuning and best configurations for EML models

Model	Hyperparameters	Description	Range	Best
LightGBM	n_estimators	# boosting iterations (trees)	100 – 500, step: 50	400
	learning_rate	Shrinkage rate	0.01 – 0.1, step: 0.01	0.05
	lambda_l1	L1 regularization	10^{-7} – 10, step: log	1.23
	lambda_l2	L2 regularization	10^{-7} – 10, step: log	7.79
	max_depth	The maximum depth of a tree	5 – 100, step: 5	65
	num_leaves	The maximum # leaves in a tree	32 – 512, step: 32	512
	min_sum_hessian_in_leaf	The minimal sum hessian in one leaf	0 – 100	57
	feature_fraction	% features to use on each iteration (without resampling)	0 – 1	0.50
	subsample	% samples to use on each iteration (without resampling)	0 – 1	0.95
XGBoost	n_estimators	# boosting iterations (trees)	100 – 500, step: 50	450
	lambda	L2 regularization term on weights	10^{-7} – 10, step: log	$9.16*10^{-5}$
	alpha	L1 regularization term on weights	10^{-7} – 10, step: log	$4.36*10^{-4}$
	max_depth	The maximum depth of a tree	1 – 15	11
	learning_rate	Shrinkage rate	0.01 – 0.1, step: 0.01	0.05
	subsample	% samples to use on an iteration	0 – 1	0.87
	min_child_weight	The minimum sum hessian in a child	0 – 10	3
	colsample_by*	% features to use for each tree, each level, and each node	0 – 1	0.70
CatBoost	n_estimators	# boosting iterations (trees)	100 – 2000, step: 100	1200
	max_depth	The maximum depth of a tree	4 – 12	11
	colsample_bylevel	% features to use at each split selection	0 – 1	0.10
	learning_rate	Shrinkage rate	0.01 – 0.1, step: 0.01	0.07
RF	n_estimators	# trees	50 – 300, step: 50	300
	max_depth	The maximum depth of a tree	10 – 300, step: 10	25
	min_samples_leaf	The minimum # samples required to be at a leaf node	1 – 20, step: 3	7
	min_samples_split	The minimum # samples required to split an internal node	2 – 10	3
ExtraTree	n_estimators	# trees	50 – 300, step: 50	250
	max_depth	The maximum depth of a tree	10 – 300, step: 10	150
	min_samples_leaf	The minimum # samples required to be at a leaf node	1 – 20, step: 3	4
	min_samples_split	The minimum # samples required to split an internal node	2 – 10	3
DT	max_depth	Maximum depth of a tree	1 – 30, step: 2	17
	min_samples_leaf	The minimum # samples required to be at a leaf node	1 – 20, step: 2	19
	min_samples_split	The minimum # samples required to split an internal node	1 – 20, step: 2	10
MLP	max_iter	# iterations	50 – 1000, step: 50	450

	alpha	L1 regularization	10^{-7} – 10^{-3}, step: log	$7.18*10^{-5}$
	hidden_layer_sizes	The # neurons in each hidden layer	(200,), (150,), (100,), (50,), (200, 150), (150, 100), (100, 50), (150, 100, 50)	(200,)
Lasso	alpha	L1 regularization	10^{-7} – 10, step: log	0.13
Ridge	alpha	L1 regularization	10^{-7} – 10, step: log	0.004
Elastic Net	alpha	L1 regularization	10^{-7} – 10, step: log	0.24
Lasso Lars	alpha	L1 regularization	10^{-7} – 10, step: log	$3.30*10^{-4}$

Models are implemented in the Python environment and are run in an AWS server (CPU: Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz; Processors: 16; Memory: 128 GB; GPU acceleration: unsupported). Linear regressions, the single tree, bagging trees, and the neural network are trained using packages including *scikit-learn* [196] and *PyCaret* [197]. Boosting trees are trained using packages including *lightgbm*, *catboost*, and *xgboost*. Parameters are tuned via the package *optuna* [198]. For interpretation, PDPs are generated via the package *scikit-learn*. SHAP is computed via *shap* [109]. ALE is plotted via *Alibi* [199].

Table 4-6 Computational efficiencies of interpretation techniques

Model	**Interpretation techniques**	**Time (s)**
CatBoost	SHAP	18.951
XGBoost		54.563
LightGBM		54.174
ExtraTree		Failed (>1 day)
RF		Failed (>1 day)
DT		Failed (>1 day)
CatBoost	Permutation importance (shuffling)	36.501
XGBoost		79.638
LightGBM		173.014
ExtraTree		916.272
RF		795.895
DT		16.386
CatBoost	Impurity importance	0.100
XGBoost		0.142
LightGBM		0.023
ExtraTree		0.472
RF		0.676
DT		0.027

CatBoost	PDP	236.280
XGBoost		101.335
LightGBM		212.365
ExtraTree		1348.892
RF		177.928
DT		14.853
CatBoost	ALE	823.686
XGBoost		1661.271
LightGBM		797.541
ExtraTree		1265.179
RF		1223.451
DT		786.296

4.5 Predictive performance

4.5.1 Performance across models

Table 4-7 shows the performance of all vanilla models (i.e., non-tuned models). Under the default setting, CatBoost exhibited the best performance among all models, yielding a 2.220% to 4.750% improvement in the four metrics compared with the next best model, i.e., XGBoost, and a 27.040% to 49.640% improvement compared with the OLS regression. Distinct tiers can be noticed in the performance of 12 models. Three boosting trees belonged to the first tier (MAPE: 33.710% - 36.190%), followed by bagging trees (MAPE: 38.100% -38.470%). The single tree belonged to the third tier but its performance sharply plummeted compared with the first two tiers (MAPE: 52.190%). The neural network performed worse under the default setting (MAPE: 52.330%), while linear regressions had the poorest performance (MAPE: 57.690% - 74.820%). Note that under default settings, some advanced linear regressions performed even worse than OLS regressions, e.g., the Elastic Net regression. The poor performance reveals a high sensitivity of machine learning models in hyperparameters. Last, Table 4-7 also reports the training time of each model. Linear regressions were executed fastest, followed by boosting trees, bagging

trees, and the neural network. One notable model was LightGBM, which exhibited a training speed close to linear regression but maintained a much higher accuracy.

Table 4-7 EML model performance comparison (vanilla)

Model	MAPE (%)	MAE	RMSE	R2	Time (s)
CatBoost	33.710	2839.907	6051.856	0.778	0.100
XGBoost	35.240 (4.340%)	2981.669 (4.750%)	6278.167 (3.600%)	0.761 (-2.220%)	0.142
LightGBM	36.190 (6.850%)	2990.022 (5.020%)	6345.502 (4.630%)	0.756 (-2.910%)	0.023
Extra Trees	38.100 (11.520%)	3083.360 (7.900%)	6601.220 (8.320%)	0.736 (-5.720%)	0.472
RF	38.470 (12.370%)	3145.416 (9.710%)	6711.353 (9.830%)	0.727 (-7.030%)	0.676
Decision Tree	52.190 (35.410%)	4548.452 (37.560%)	9868.702 (38.680%)	0.400 (-94.650%)	0.027
MLP	52.330 (35.580%)	3687.320 (22.980%)	7015.183 (13.730%)	0.699 (-11.290%)	5.447
Lasso Lars	57.690 (41.570%)	4176.725 (32.010%)	8245.457 (26.600%)	0.587 (-32.520%)	0.002
Lasso	63.850 (47.200%)	4241.997 (33.050%)	7884.684 (23.250%)	0.622 (-25.060%)	0.050
Ridge	64.100 (47.410%)	4248.953 (33.160%)	7884.151 (23.240%)	0.622 (-25.040%)	0.010
Linear	66.940 (49.640%)	4462.324 (36.360%)	8295.291 (27.040%)	0.581 (-33.840%)	0.013
Elastic Net	74.820 (54.950%)	4991.004 (43.100%)	9423.714 (35.780%)	0.459 (-69.610%)	0.029

Note: Percentages in brackets are the increase in metrics versus the best model (the model in the first row). Rows are sorted in ascending order based on MAPE. All metrics are computed on the testing set.

Table 4-8 shows the performance of fine-tuned models, and **Figure 4-6** shows the prediction versus observation plot. Compared with vanilla models, a significant improvement in the performance of fine-tuned models can be observed. LightGBM now outperformed all models but with a close performance to CatBoost. It outperformed the neural network by 3.380% - 8.140%, bagging trees by 8.240% - 16.150%, the single tree by 22.430% - 26.100%, and linear regressions by 24.770% - 59.340%. The ranking among different models also varied. For example, the performance of the neural network was greatly enhanced after hyperparameter tuning

and became the second tier, yielding a MAPE of 29.632%. The training time, now including the time of model tuning, documented that the neural network required the longest time to train, while linear regressions were executed the fastest. LightGBM, again, performed saliently in both efficiency and accuracy.

Table 4-8 EML model performance comparison (fine-tuned)

Model	MAPE (%)	MAE	RMSE	R2	Time (s)
LightGBM	27.220	2618.580	6036.795	0.775	52.196
CatBoost	27.237 (0.060%)	2612.934 (-0.220%)	5993.186 (-0.730%)	0.779 (0.480%)	126.569
XGBoost	27.418 (0.720%)	2632.713 (0.540%)	6092.767 (0.920%)	0.772 (-0.480%)	655.694
MLP	29.632 (8.140%)	2984.265 (12.250%)	6375.118 (5.310%)	0.750 (-3.380%)	4550.786
Extra Trees	29.666 (8.240%)	2905.229 (9.870%)	7199.347 (16.150%)	0.681 (-13.830%)	113.562
RF	30.362 (10.350%)	2931.704 (10.680%)	6997.812 (13.730%)	0.699 (-10.960%)	208.988
Decision Tree	36.835 (26.100%)	3486.975 (24.900%)	7782.284 (22.430%)	0.627 (-23.590%)	18.558
Lasso Lars	57.638 (52.770%)	4220.682 (37.960%)	8483.213 (28.840%)	0.557 (-39.160%)	1.036
Elastic Net	61.810 (55.960%)	4174.502 (37.270%)	8153.295 (25.960%)	0.591 (-31.200%)	142.855
Lasso	62.610 (56.520%)	4201.375 (37.670%)	8051.468 (25.020%)	0.601 (-28.980%)	87.324
Ridge	63.508 (57.140%)	4220.267 (37.950%)	8024.762 (24.770%)	0.604 (-28.420%)	0.863
Linear	66.940 (59.340%)	4462.324 (41.320%)	8295.291 (27.230%)	0.581 (-33.360%)	0.013

Note: Percentages in brackets are the increase in metrics versus the best model (the model in the first row). Rows are sorted in ascending order based on MAPE. All metrics are calculated on the testing set.

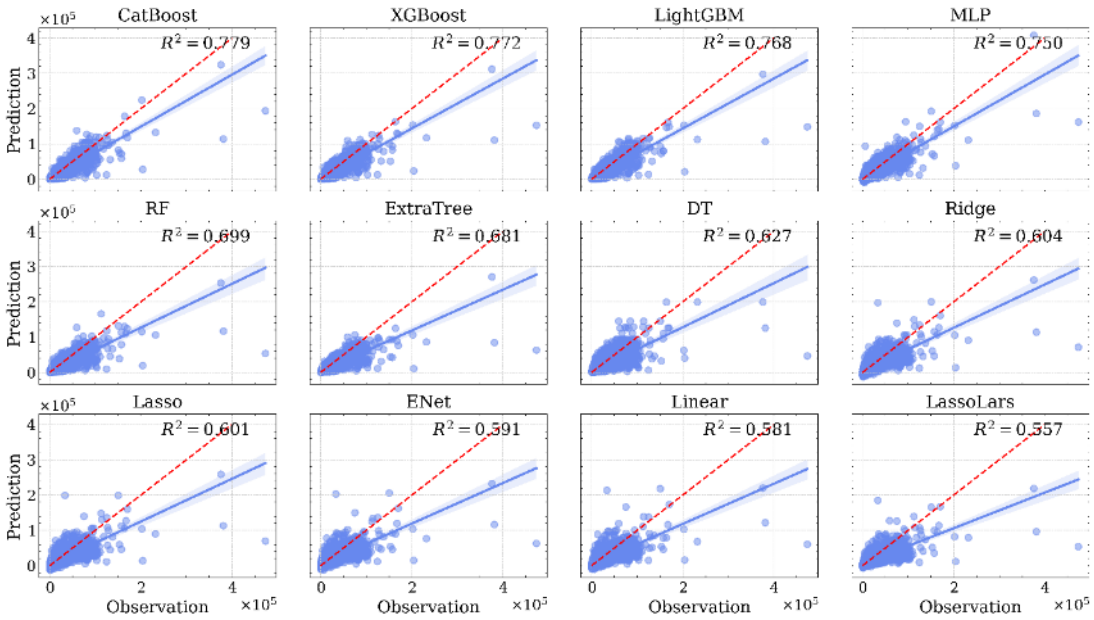


Figure 4-6 Prediction vs. Observation Plot across EML models

Note: Each spot represents one CBG in the testing set. The red dashed line has a slope of 1. The blue line is the linear fit of prediction vs. observation, with 95% CI showing in buffers.

4.5.2 Performance across MDLD sampling rate

It is noteworthy that the prediction target in this study is not the ground truth, but a proxy measured by the MDLD collected from observed samples. Hence, the gap between the ground truth and the proxy may vary across CBGs with different sampling rates (i.e., the number of observed mobile devices divided by the CBG total population). A much lower sampling rate indicates the proxy is not population-representative, while a much higher sampling rate is more likely due to the tiny population size of the CBG or some irregular observations. Checking how model performance varies across sampling rates provides insight into the spatial fairness of MDLD, as well as whether the residual should be sourced to the model or the data.

Figure 4-7 (a) shows how the performance of the best model, i.e., the fine-tuned LightGBM, varies across sampling rates. As shown, population inflow of CBGs with much lower or higher sampling rates (i.e., Quantiles 1 and 15) were harder to predict in terms of MAPE, which is plausible since the travel demand of these CBGs was more likely to be imprecisely measured due to biased sampling. MAE, on the other hand, presented a monotonically increasing relation with the sampling rate. It is also explainable since MAE is an absolute performance metric. CBGs with higher sampling rates were more likely to have larger populations, corresponding to a larger prediction target. Such an explanation can be further affirmed by **Figure *4-1*** (b). Another visible pattern in **Figure 4-7** (b) is the model fairness issue [200]. The model tended to underestimate CBGs with lower sampling rates (blue lines) and overestimate CBGs with higher sampling rates (red lines). These estimation errors, however, may be partially attributed to the measurement biases, since MDLD also tended to underestimate the travel demand of CBGs with lower sampling rates and overestimate others with higher sampling rates.

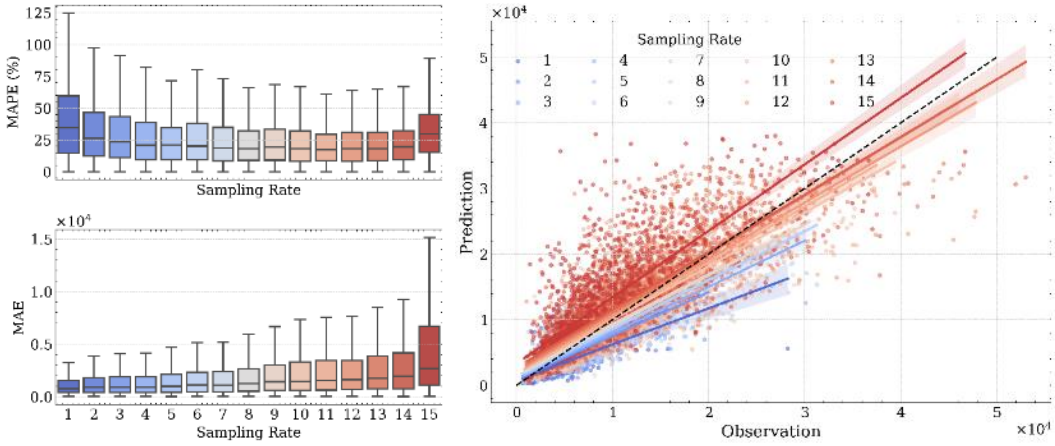


Figure 4-7 Model performance (a) and Prediction vs. Observation Plot (b) across different sampling rates

Note: Sampling rates are split into 15 bins by quantiles. Outliers, which are defined as those exceeding the 1.5*inter-quartile range (IQR), are removed. The black dashed line in panel (b) has a slope of 1, while the other lines are the linear fits of prediction vs. observation, buffered by 95% CIs.

4.6 Feature importance

4.6.1 Feature importance of tree-based models

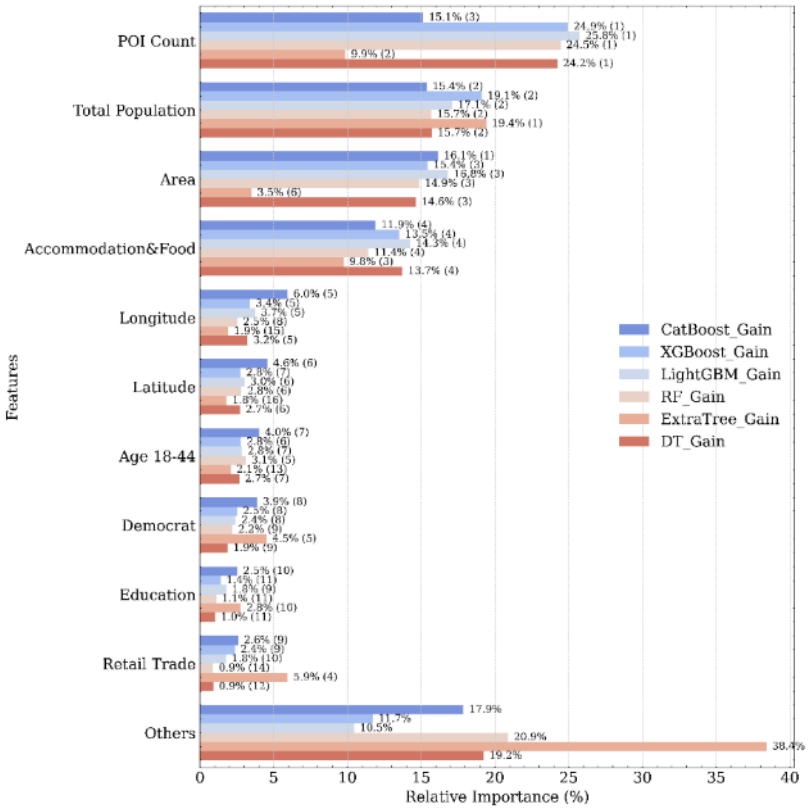


Figure 4-8 Impurity importance of tree-based models

Note: Importance is normalized to 0-100%. Only the top 10 features (ranked by LightGBM Gain Importance) are plotted. Other features are grouped as “Others”. The number in parentheses on the right side of each bar denotes the rank.

The impurity importance, which measures the importance by the total information gain brought by the feature, is presented in **Figure 4-8**. As shown, the most important features broadly stayed consistent across different tree-based models. POI count, total

population, CBG area, and # accommodations and food stores were the four most important features in most models. However, the ranks of the less important features varied significantly across models. For example, the longitude of CBG ranked 5th in CatBoost, XGBoost, LightGBM, and DT, but ranked 15th in ExtraTree and ranked 8th in RF. Meanwhile, the feature importance of some models was more similar to each other compared to others. Specifically, XGBoost, LightGBM, RF, and DT presented a highly similar distribution regarding their feature importance, while nuances emerged when comparing them with CatBoost and ExtraTree. For example, the CBG area instead of the POI count ranked first in CatBoost, while its rank changed to 6th in ExtraTree. Another takeaway message is that the distribution of feature importance was highly uneven. The top 10 features among the total 84 features (36 continuous features + 48 dummy variables) accounted for most of the total importance (89.5% in LightGBM, 88.3% in XGBoost, 82.1% in CatBoost, 80.8% in DT, 79.1% in RF, and 61.6% in ExtraTree), indicating a large fraction of features failed to provide sufficient knowledge for predicting the target.

The permutation importance is measured by the decrease in model accuracy when the link between the feature and the target is broken. Two types of permutation importance, including the shuffling method and the SHAP method, were reported (**Figure 4-9**). By comparing the two methods, one finding was that SHAP importance was more consistent across models and more evenly distributed across features. For example, compared with the shuffling method, the weight of inconsequential features (i.e., "Others" in **Figure 4-9**) significantly increased when using the SHAP method (20.1% → 27.6% for XGBoost and 10.3% → 17.9% for LightGBM).

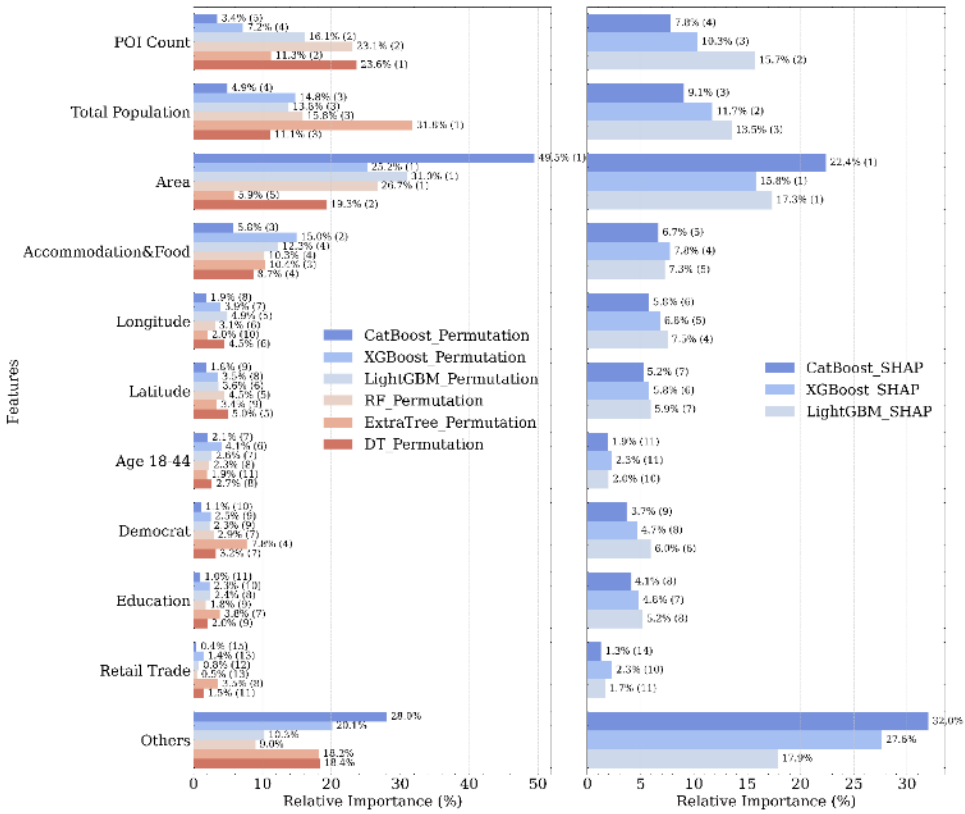


Figure 4-9 Permutation importance of tree-based models (Shuffling vs. SHAP)

Note: Due to the high computational cost of estimating SHAP importance for non-boosting trees, the SHAP importance is only computed for three boosting trees.

Permutation importance was analogous to impurity importance (**Figure 4-8**) in many ways. For example, the permutation importance was highly similar for XGBoost, LightGBM, RF, and DT but relatively different from CatBoost and ExtraTree; the distribution of permutation importance was highly uneven; the top four important features were similar between permutation and impurity importance. However, several differences can also be observed. The most distinguishable difference was in the most important features estimated by the two methods: POI count was ranked first in impurity importance while the first changed to CBG area in permutation importance. The main reason may be that the permutation importance was sensitive to feature dependency. In this study, POI count, total population, and #

accommodations and food stores were highly correlated. Such a dependency indicated that losing one of them would not sharply decrease the model accuracy since other correlated features can serve as a substitute. Hence, the CBG area, which was not highly correlated with other features, became the most important feature.

4.6.2 Comparison with regression coefficients

Figure 4-10 reports standardized coefficients of linear regressions. Coefficients of different linear models were similar in both magnitudes and signs. Most of the top 10 features were positively related to population inflow, except for the % Democrats and the latitude of CBG. Compared with the feature importance of tree-based models (**Figure 4-8** and **Figure 4-9**), some differences can be found. First, total population and # accommodations and food stores showed the strongest relation with population inflow in linear regressions, while the POI count, which was ranked as most important by impurity importance in trees, became less important. One explanation is that the positive correlations among total population, # accommodations and food stores, and POI count, induced the multicollinearity issue in linear regressions, which obscured the effects of POI count on population inflow. Second, some features such as CBG area and longitude of CBG, which were ranked fairly high in trees, were even regressed out of the top 10 in linear regressions. The reason may be that their relations with population inflow were essentially nonlinear (**Figure 4-13**). Hence, only models that can well capture nonlinear relations would assign high importance to features like CBG area and coordinate.

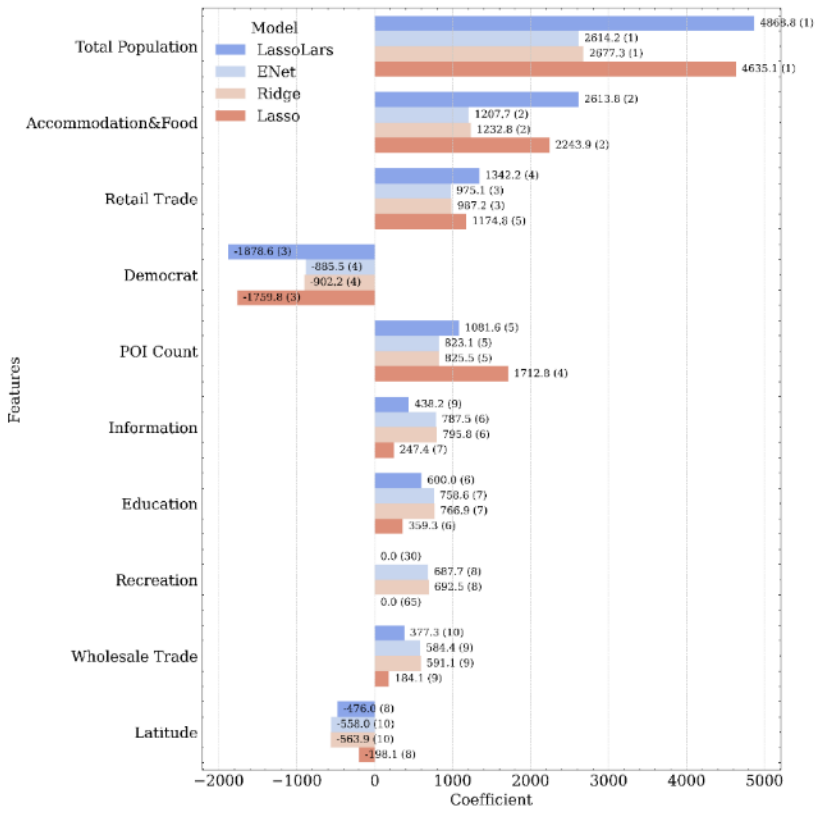


Figure 4-10 Standardized coefficients of linear regressions

Note: Standardized coefficients mean features used to fit the linear regressions are z-score normalized to eliminate the effects of scales.

4.6.3 Robustness check of feature importance

The robustness check was achieved by exploring whether feature importance varied across the same model with different parameter configurations. For brevity, only the best model, i.e., the LightGBM, was checked. This study first examined the relation between feature importance and model performance. To this end, the feature importance of all trials of LightGBM during the hyperparameter tuning process was extracted and plotted using the MAPE as the x-axis (**Figure 4-11**). As shown, the tuning process quickly decreased the MAPE to a low level within 10 trials, and most of the other trials (**Figure 4-11** (b)) brought little improvement to model performance.

Interestingly, even a model with the poorest performance, for example, with a MAPE of ~ 50%, can still accurately identify the most important features such as POI count, total population, area, and # accommodations and food stores. In addition, **Figure 4-11** illustrates the convergence in feature importance as the MAPE approached the optimum, indicating high robustness of feature importance in fine-tuned trees.

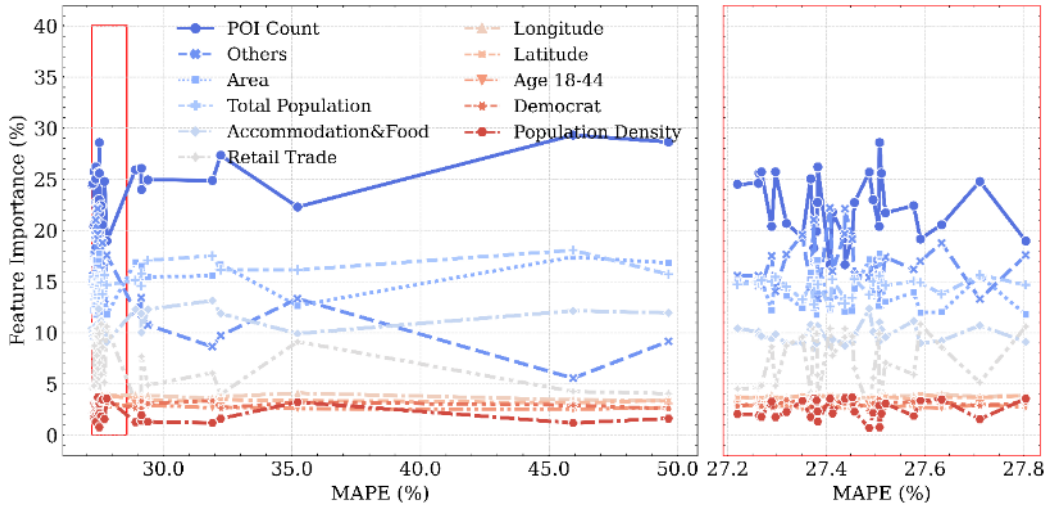


Figure 4-11 Evolution of impurity importance varying across different MAPE

Note: Importance is normalized to 0-100%. Only the top 10 important features in LightGBM are plotted. Other features are grouped as "Others". Panel (b) is the zoom-in view of panel (a) covering trials with MAPE from 27.2% to 27.9%.

Although **Figure 4-11** provided some evidence of the overall robustness of feature importance, it remained unclear to what extent each hyperparameter could influence the feature importance. Hence, this study further conducted sensitivity analyses on the relation between feature importance and each hyperparameter. For each panel in **Figure 4-12**, the hyperparameter in the x-axis was changed in grids, while the remaining hyperparameters were held constant as their best values (see the detailed description of all hyperparameters in **Table *4-5***). The six most important hyperparameters were tested, including # trees, the feature sampling rate, the

minimum leaf weight, the maximum tree depth, the learning rate, and the maximum # leaves. Learning curves were also depicted to prevent drawing findings from overfitting models, such as models with learning rates exceeding 0.25.

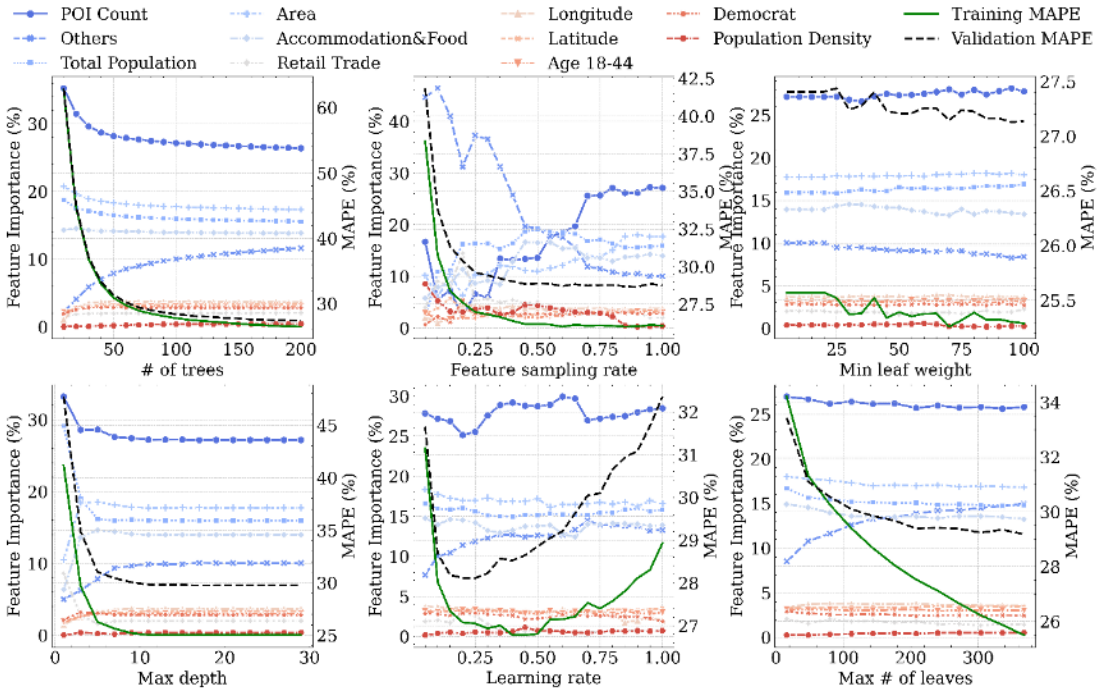


Figure 4-12 Sensitivity analysis of impurity importance

Note: The left y-axis shows the relative feature importance. The right y-axis shows the model MAPE. Only the top 10 features in LightGBM are plotted. Other features are packed as "Others".

Two takeaway messages can be concluded from **Figure 4-12**. First, feature importance tended to distribute more evenly as the tree ensemble grew more complex. For example, the importance of features out of the top 10 (i.e., "Others") steadily increased with the increase in # trees, the depth of trees, and # leaves in trees, while the importance of the most important features overall decreased. The model's focus transitioning from top features to others indicates that the model was learning deeper patterns and more comprehensive relations by utilizing more features. Second, most of the hyperparameters did not show pronounced effects on feature importance,

except for the feature sampling rate, which is % features to use on each iteration. A low feature sampling rate may lead to an unstable distribution of feature importance. Hence, a high feature sampling rate is suggested when interpretation is required.

4.7 Nonlinear relations

4.7.1 Global nonlinear relations: PDP and ALE

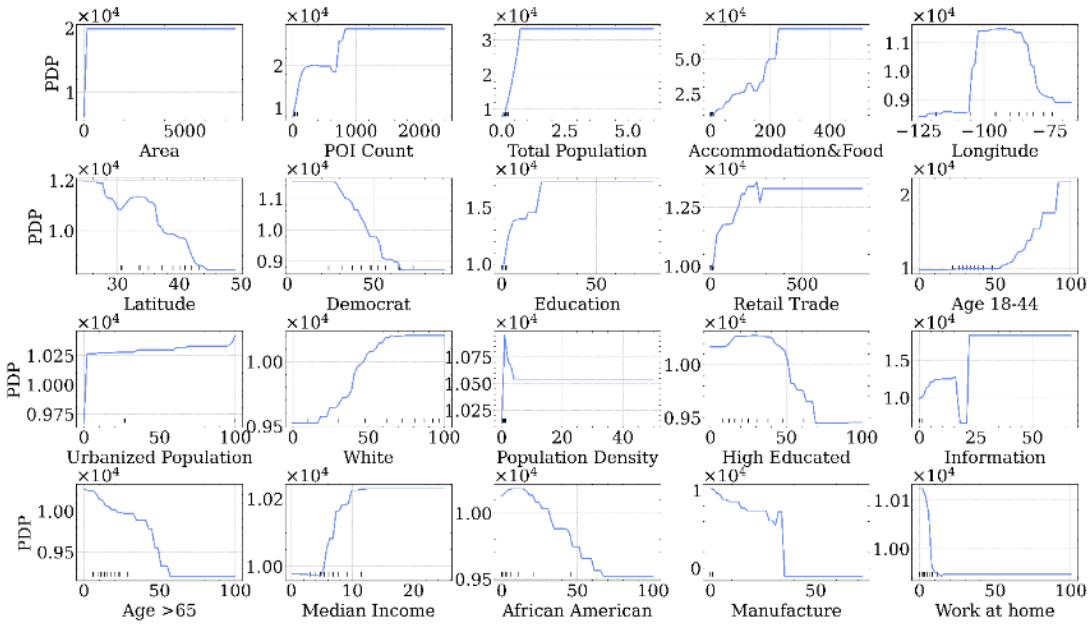


Figure 4-13 PDPs of the top 20 important features

Feature importance only measures the overall influence of a feature but cannot illustrate detailed relations. PDP, on the other hand, can visualize the global marginal effect a feature has on the prediction. **Figure 4-13** shows the PDPs of the top 20 important features (measured by impurity importance in **Figure 4-8**) extracted from the fine-tuned LightGBM. For brevity, only the best model is reported here. The main findings are summarized as follows:

1) CBG area, POI count, total population, # accommodations and food stores, # education services, # retail trades, % residents aged 18-44, % urbanized populations, % Whites, and median household income all presented positive relations with population inflow, although most of their trends were not strictly monotonic but thresholded. Specifically, most of them presented positive relations during certain intervals but remained stable in other ranges. For example, % residents aged 18-44 only presented a positive relation when it exceeded 50%; % Whites only presented a positive relation between 15% and 70%; the median household income only presented a positive relation between 4 and 10 ($\$10^4$/household). Another interesting pattern was that POI count, # accommodations and food stores, # education services, and # retail trades all exhibited some intermediate stagnations or jumps before reaching the plateau. These odd perturbations were very likely caused by feature dependency since similar patterns were also documented in previous studies when using PDP to delineate correlated features [105].

2) Latitude, % Democrats, % highly-educated residents, % elderly, % African Americans, # manufacturing POIs, and % telecommuters all presented negative relations with population inflow. Similarly, such relations were not strictly monotonic. For example, % elderly only presented a negative relation when it was below 50%; % Democrats only presented a negative relation between 25% and 65%; % highly-educated residents only presented a negative relation between 40% and 70%; % African Americans only presented a negative relation between 15% and 70%; % telecommuters only presented a negative relation below 10%.

3) Other variables presented more complex nonlinear relations with population inflow. For example, the longitude of CBG presented a downward parabolic pattern when it was located between 75 to 105° W. Another noticeable pattern was that there existed several irregular drops or spikes in some relations, such as the population density, % urbanized populations, # information POIs, and # manufacturing POIs. These irregularities should be carefully interpreted since they were more likely to be sourced to the effects of outliers.

PDP may not be reliable in some cases since it cannot handle the intertwined effects of correlated features and overfitting biases caused by outliers [105]. **Figure 4-13** has already indicated some potential failures, such as irregular perturbations, steep drops or spikes, and long leading stagnation and tailing plateau. Hence, ALE plots were introduced and shown in **Figure 4-14**. By comparing **Figure 4-13** with **Figure 4-14**, high consistency can be observed regarding their overall trends. However, several new findings can be documented as follows:

1) Some of the long tailing plateaus in PDPs were clearly illustrated by ALE plots as outlier effects, such as POI count, total population, # accommodations and food stores, # education services, # retail trades, # information POIs, and # manufacturing POIs. One possible solution is to limit the range of features used to create the PDP, for example, trimming PDPs by 5 and 95 percentiles of the feature. However, it is difficult to determine the best trimming point for each feature. The ALE, on the other hand, computed local effects by dividing the feature value into intervals, which constrained the effects of outliers into a local interval and thus alleviated their effects on the holistic relations.

2) Most of the intermediate stagnations and jumps in PDPs of POI count, # accommodations and food stores, # education services, and # retail trades were eliminated in ALE plots, implying that the ALE can successfully address the feature dependency issue. Meanwhile, the steep drops or spikes in PDPs of # information POIs, % urbanized populations, and # manufacturing POIs were also clearly illustrated by ALE plots as outlier effects.

3) If removing the outliers, threshold patterns still existed in relations between population inflow and features such as the CBG area, the latitude of CBG, % Democrats, % residents aged 18-44, % Whites, % highly-educated residents, % the elderly, median household income, % African Americans, and % telecommuters. Threshold effects in these features were alleviated by more samples and thus can be viewed as more reliable.

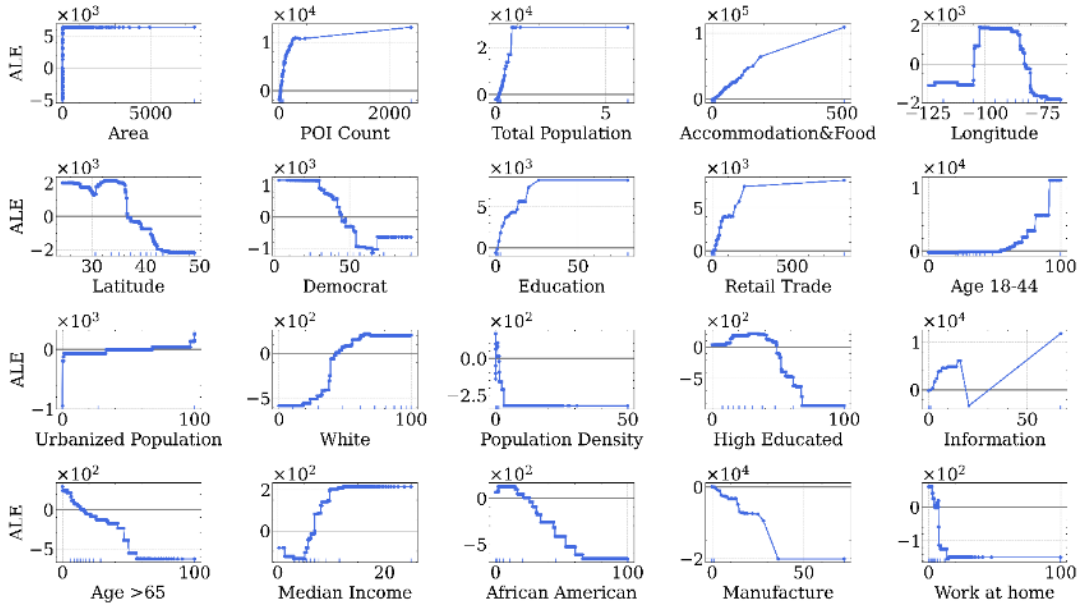


Figure 4-14 ALE plots of the top 20 important features

Note: For each subplot, the x-axis denotes the feature value, while the y-axis denotes the feature effect on population inflow relative to the average effect across the dataset.

4.7.2 Local nonlinear interaction: SHAP

One main limitation of PDP and ALE is that local heterogeneous effects might be hidden since they only show average effects. One solution is to use local interpretation methods such as SHAP. **Figure 4-15** shows the SHAP interaction plot of the top 20 important features in fine-tuned LightGBM. Comparing it with PDPs and ALE plots, the overall relations remained consistent, but with more details uncovered at an individual level or from the interaction aspect:

1) Consistently with PDPs and ALE plots, POI count, total population, # accommodations and food stores, # education services, # retail trades, and # information POIs, all presented positive relations with population inflow, while # manufacturing POIs presented a negative relation. Similar to ALE plots, SHAP plots also showed that tailing plateaus in PDPs of these features were caused by outliers.

2) In line with PDPs and ALE plots, % Democrats, % highly-educated residents, % the elderly, and % African Americans all presented negative relations with population inflow in some specific ranges but exhibited steeper slopes (i.e., more negative) in CBGs with more POIs. Conversely, % White persons and median household income, all presented positive relations in some ranges but exhibited steeper slopes (i.e., more positive) in CBGs with more POIs. % residents aged between 18-44 presented an exponentially positive relation when it was over 50% but exhibited steeper slopes in CBGs with lower population density.

3) Interaction effects were also found in spatial features. For example, regarding the relation between the longitude of CBG and population inflow, CBGs with more POIs presented a downward parabola of higher kurtosis. Meanwhile, the

latitude presented a negative relation with population inflow after 36° N but exhibited steeper slopes in CBGs with more POIs. These patterns were consistent with the spatial distribution of population inflow shown in **Figure *4-1***.

4) Several relations in SHAP plots were different from PDPs and ALE plots. For example, a positive relation between CBG area and population inflow can only be observed in CBGs with fewer POIs, and such a positive relation rapidly flattened as the area grew. On the other hand, for CBGs with more POIs, the CBG area showed a limited impact on population inflow. In addition, % telecommuters, % urbanized populations, and population density all failed to present informative relations with population inflow at an individual level.

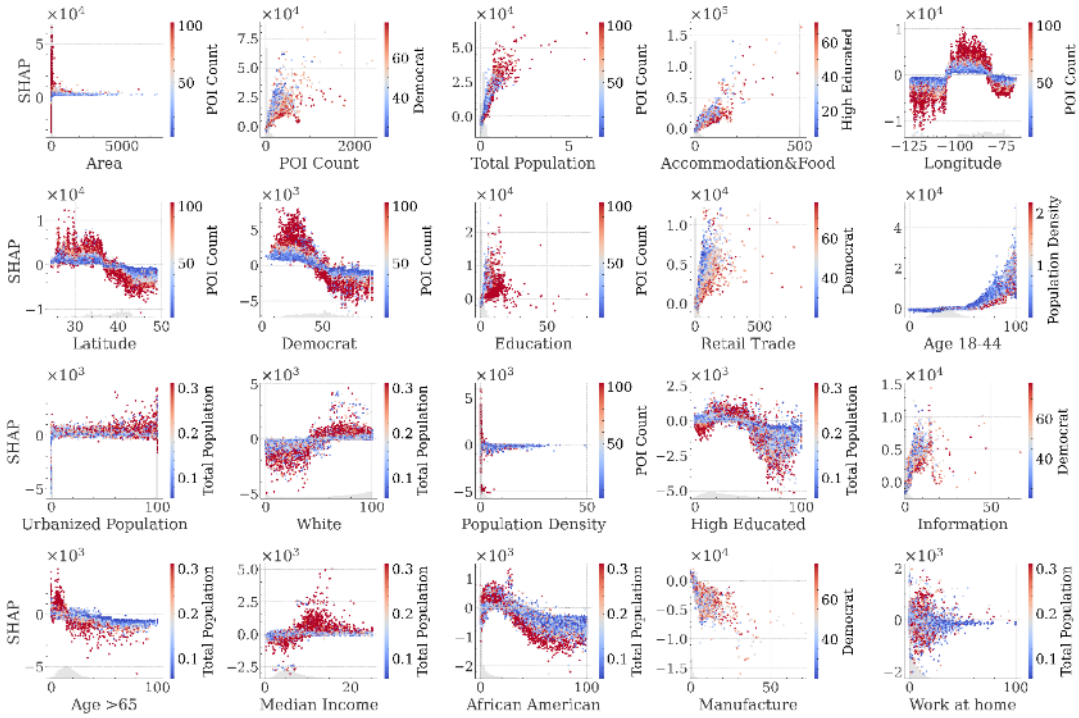


Figure 4-15 SHAP interaction plots of the top 20 important features

Note: In each subplot, the x-axis is the feature, the y-axis is the contribution of the feature to the difference between the actual and the mean prediction, the right color bar is the interaction feature, and

each point represents a single CBG with its color changing by the interaction feature. The interaction feature is selected from all other features that show the greatest interaction effects.

4.8 Discussion

This study is among the first to comprehensively compare a series of EML models and interpretation techniques in travel demand estimation using a nationwide MDLD-based travel demand dataset. Various nonlinearities, threshold effects, and interaction effects are uncovered in relations between travel demand and external factors. Moreover, the extensive comparison across models and interpretations provides empirical evidence of their pros and cons, as well as their sensitivity to hyperparameters and data attributes. Retracing back to the three research questions at the beginning, the answers can now be documented as follows.

How do different ML models perform in estimating travel demand? ML models exhibit fairly high accuracy in estimating MDLD-based travel demand. Among fine-tuned models, boosting trees present the best accuracy, followed by neural networks, bagging trees, single tree, and linear regressions. LightGBM outperforms all models in this study and executes as fast as linear regressions. Another noteworthy finding is the model fairness issue across regions with different mobile device sampling rates. Models present higher MAPE in CBGs with much lower or higher sampling rates. This may be explained by the sampling biases, but may also be sourced to the different travel patterns between urban and rural regions.

How should ML models be interpreted and what are the main findings? This study introduces six different interpretation techniques to illustrate the knowledge learned by ML models. Among feature importance, the impurity importance is more

appropriate for this study since it allows feature dependency and it is computationally efficient. Measured by impurity importance, POI count, total population, CBG area, and # accommodations and food stores, are the top 4 important features and account for 74% of the total importance in the fine-tuned LightGBM. In addition, the feature importance of tree-based models is not explicitly comparable with coefficients of linear regressions, which may be due to the effects of multicollinearity and nonlinearity. Among relation visualization methods, PDP suffers from irregular perturbations and long leading/tailing plateaus due to its assumption of feature independence and its sensitivity to outliers. ALE plots help to address these issues. The SHAP interaction plot further enhances the interpretation reliability and informativeness by focusing on heterogeneous interaction effects.

Do interpretation outcomes hold robustly? Feature importance of the top-tier features broadly holds consistently among different models. The most important features are captured well by all models, even by a vanilla single tree. However, those less important features may vary across models and show less robustness. The importance generated by different techniques, including the impurity-based and permutation-based measures, is also broadly consistent. However, permutation-based measures are affected by feature dependency, presenting underestimated importance of those correlated features. For hyperparameters, the feature sampling rate shows the greatest effect on feature importance. A low feature sampling rate may lead to an unstable distribution of feature importance, even with relatively high prediction accuracy. Meanwhile, feature importance tends to shift from the most important features to those inconsequential features as the tree ensemble grows more complex.

5 Chapter 5: Population flow time series forecasting

The previous chapter describes how MDLD can be parsed to distill the travel information, as well as how to interpret EML models to understand nonlinear relations between the processed travel demand and underlying factors. However, these analyses are presented from a cross-sectional aspect, while the longitudinal travel demand time series remains unexplored. Accurate forecasting of the inflow can largely benefit the whole life cycle of travel demand modeling by providing insights in a time-varying and continuous manner instead of traditional snapshots. Using historical time series as the training set, this chapter introduces a graph-based temporal neural network to forecast future citywide population inflow time series.

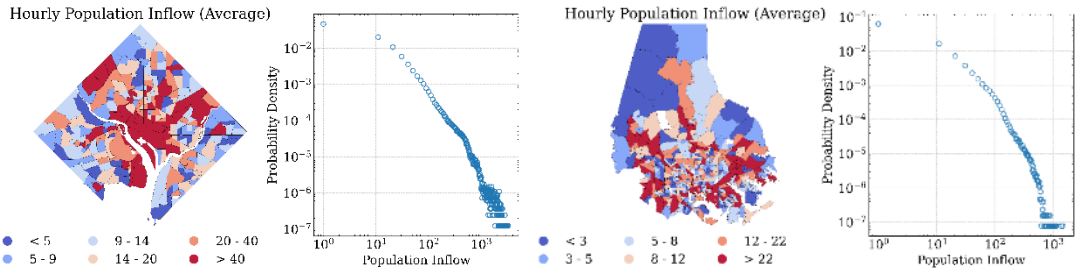
Generally, the problem of population inflow forecasting is challenging as it involves complex spatiotemporal dependency, diverse temporal dynamics, and high nonlinearity triggered by external factors [86]. Compared to traditional statistical and machine learning models that have limited expressiveness and flexibility, deep learning methods have become prevalent because of their strong capability in handling nonlinear relationships, unstructured data, and knowledge fusion [85]. However, there are several challenges to be addressed:

1) **Diverse temporal dynamics**. Temporal patterns of hourly population flow time series are highly diverse since they are a mixture of different seasonality (e.g., daily and weekly), trends (e.g., short-term and long-term), and white noise. Meanwhile, they also exhibit high locality since travel behaviors are a function of numerous local factors. Another visible issue is the over-dispersion nature of population inflow across different zones, and such a dispersion is further intensified

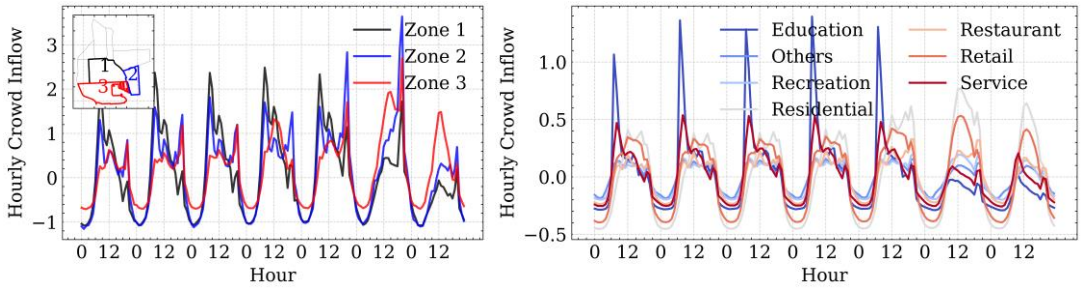
by irregular zone systems (see the near power law pattern in **Figure 5-1** (a-b)). The high diversity in population inflow time series requires the model can learn both global and local knowledge from heterogeneous zone-specific patterns [201].

2) **Multi-view graph structures**. Unlike microscopic traffic flow mostly constrained by road connectivity, spatial dependency in population inflow is not only constrained by distance but also highly correlated with zonal functionality, accessibility, mobility connectedness, and other unobserved factors. As shown in **Figure 5-1** (a-c), the spatial distribution of population inflow does not strictly follow a distance decay rule. Nearby census tracts may also present different population inflow temporal patterns. Hence, a single predefined adjacency matrix cannot well describe the real structure of population inflow graphs.

3) **Effects of external factors**. Population inflow time series is associated with a variety of external factors with diverse dimensions. Time-varying features like weather and holidays would trigger abnormal fluctuations in population inflow. In addition, temporal cycle patterns of the population inflow are conditional on the zonal static features such as socioeconomics, demographics, and land use (**Figure 5-1** (d)). Such a diverse set of external information should be carefully handled to enable the model to learn useful knowledge from multi-dimensional variables.



(a) Spatial plot and log-log plot in D.C. (b) Spatial plot and log-log plot in Baltimore



(c) Time series of nearby zones (d) Time series across land use types

Figure 5-1 Spatiotemporal patterns of population inflow

To address these challenges, this chapter introduces a Multi-graph Multi-head Adaptive Temporal Graph Convolutional Network (Multi-ATGCN) for multivariable population inflow forecasting. Specifically, the Multi-ATGCN contains several main modules to address the challenges accordingly: 1) A Multi-head temporal fusion module to fuse multiple temporal patterns including closeness, period, and trend. 2) A Multi-view adaptive graph construction module to learn an adaptive graph structure given prior knowledge from different adjacency matrices measured by distance closeness, OD volume, and functional similarity. 3) An integration of Recurrent neural network (RNN) and Zone-specific mix-hop GCN (ZMGCN) for jointly handling complex spatiotemporal dependency. 4) An Auxiliary variable enrichment decorator scattering across the framework to handle external static variables and temporal variables via parameter initialization and sequence concatenation. The proposed model is evaluated on two real-world citywide datasets and exhibits steady performance improvement and comparable efficiency over extensive state-of-the-art baselines. Such an improvement is even more salient in data-sparse zones and long-horizon scenarios that are more difficult to predict.

5.1 Problem statement

This study is intended to forecast the future population inflow of each geographic zone across a city. The population inflow is defined as the hourly number of people entering a specific zone. Instead of splitting the city into grids, this study adopts the irregular administrative zone, i.e., the census tract, as the analytical unit. All census tracts across the city are viewed as a directed graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{A})$, where $\boldsymbol{V}$ is the set of $|\boldsymbol{V}| = N$ zones (i.e., census tracts), $\boldsymbol{E}$ is the set of edges indicating the connectivity between zones, and $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. Note that one graph may have multiple adjacency matrices to describe multi-view connectivity. Hence, a generalization of $\boldsymbol{A}$ is $\widehat{\boldsymbol{A}} \in \mathbb{R}^{M \times N \times N}$, where M is # adjacency matrices of the graph $\mathcal{G}$.

In population inflow forecasting, the graph is assumed to be static, while variables attached to each zone may be either static or time-varying [202]. The whole time series is split into multiple time fragments by a rolling window with a d_E-step historical window and a d_D-step prediction window. Then, for a time fragment whose current time is t_0, each zone i is associated with a set of features including:

1) Time-varying auxiliary features $\boldsymbol{Z}_{i,(t_0-d_E):t_0} = \left[\boldsymbol{z}_{i,t_0-d_E}, \boldsymbol{z}_{i,t_0-d_E+1}, \dots, \boldsymbol{z}_{i,t_0-1}\right]^T \in \mathbb{R}^{d_E \times d_F}$, such as holidays, weekends, hour index, and weather conditions, where d_F is the number of time-varying auxiliary features.

2) Static features $\boldsymbol{S}_i \in \mathbb{R}^{1 \times d_S}$, such as socioeconomics, demographics, and land use, where d_S is the number of static features.

3) Historical population inflow $\boldsymbol{Y}_{i,(t_0-d_E):t_0} = \left[y_{i,t_0-d_E}, y_{i,t_0-d_E+1}, \dots y_{i,t_0-1}\right]^T \in \mathbb{R}^{d_E \times 1}$.

Let $\boldsymbol{X}_{:,(t_0-d_E):t_0} = \left[\boldsymbol{Y}_{:,(t_0-d_E):t_0}, \boldsymbol{Z}_{:,(t_0-d_E):t_0}\right] \in \mathbb{R}^{N\times d_E\times(d_F+1)}$, the goal of this study is to learn a nonlinear function $\mathcal{F}(.)$ between features and future population inflow patterns across all census tracts, given the graph structure $\mathcal{G}$:

$$\widehat{\boldsymbol{Y}}_{:,t_0:(t_0+d_D)} = \mathcal{F}\left(\boldsymbol{X}_{:,(t_0-d_E):t_0}, \boldsymbol{S}; \mathcal{G}\right), \forall t_0 \in [d_E, d_E+1, \dots, T-d_D] \quad (5\text{-}1)$$

where T is the duration of the time series.

5.2 Proposed approach: Multi-ATGCN

This study introduces a Multi-ATGCN for multivariable population inflow forecasting. **Figure 5-2** shows the high-level architecture, with each module described in detail in the following section.

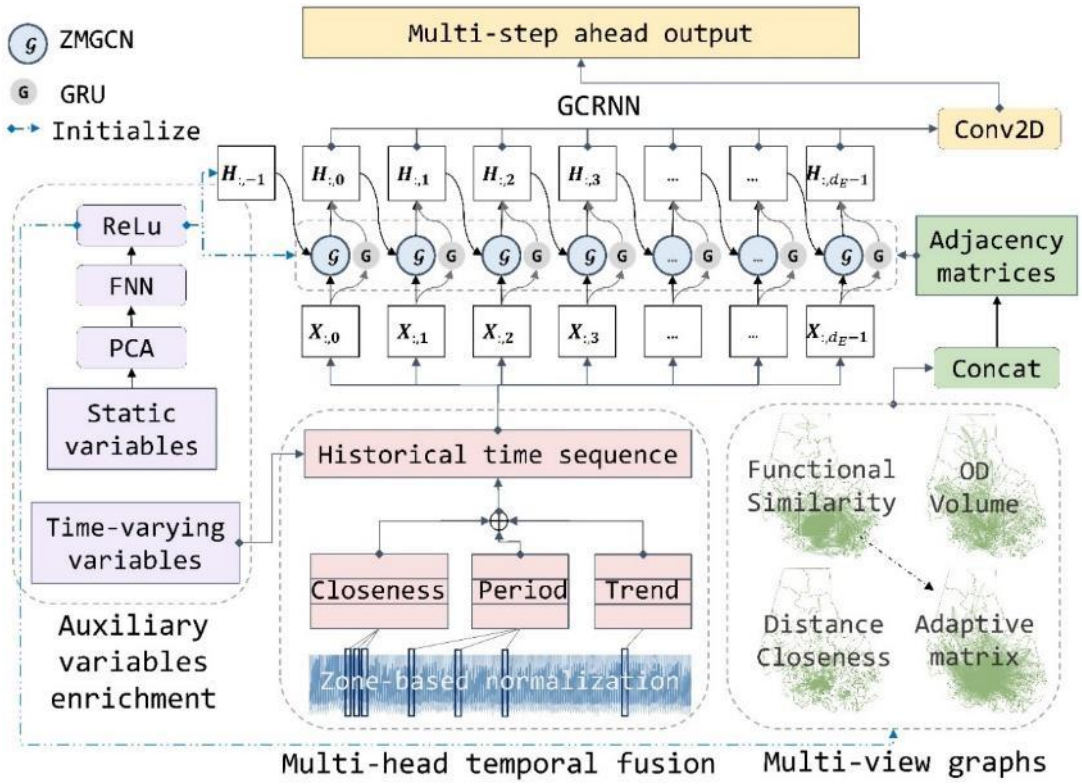


Figure 5-2 The Multi-ATGCN architecture

5.2.1 Multi-head temporal fusion

As mentioned before, the distribution of population inflow across different census tracts is strongly skewed. Some zones, such as those located downtown, have a much larger population inflow compared with those located suburban. This may cause the model to focus on more attractive census tracts while ignoring the less vibrant zones. Hence, a census tract-based normalization is applied to each time series, followed by a global normalization, to assign a zone-specific bias to each census tract [203]:

$$y'_{i,t} = (\frac{y_{i,t} - \mu_i}{\sigma_i} - \mu')/\sigma' \quad (5\text{-}2)$$

where μ_i and σ_i are the mean and standard deviance (st.d.) of the population inflow in zone *i* across the training set; $y_{i,t}$ is the population inflow of zone *i* at time t and $y'_{i,t}$ is its normalization; μ' and σ' are the global mean and st.d. of the normalized population inflow across all census tracts.

After normalization, three temporal heads are extracted from the raw time series to represent multi-dimensional temporal patterns [29, 204], including the closeness (daily patterns), the period (weekly patterns), and the trend (monthly patterns). Let t_0 be the current time, the set of closeness, period, and trend heads are:

$$\boldsymbol{Y}_C^{(k)} = \boldsymbol{Y}'_{:,(t_0 - d_C * k - d_E):(t_0 - d_C * k)} \in \mathbb{R}^{N \times d_E}, k = [0,1,2,\dots,n_C - 1] \quad (5\text{-}3)$$

$$\boldsymbol{Y}_P^{(k)} = \boldsymbol{Y}'_{:,(t_0 - d_P * k - d_E):(t_0 - d_P * k)} \in \mathbb{R}^{N \times d_E}, k = [1,2,\dots,n_P] \quad (5\text{-}4)$$

$$\boldsymbol{Y}_T^{(k)} = \boldsymbol{Y}'_{:,(t_0 - d_T * k - d_E):(t_0 - d_T * k)} \in \mathbb{R}^{N \times d_E}, k = [1,2,\dots,n_T] \quad (5\text{-}5)$$

where $\boldsymbol{Y}_C^{(k)} \in \mathbb{R}^{N \times d_E}$, $\boldsymbol{Y}_P^{(k)} \in \mathbb{R}^{N \times d_E}$, and $\boldsymbol{Y}_T^{(k)} \in \mathbb{R}^{N \times d_E}$ are the set of the kth closeness heads, period heads, and trend heads, respectively; d_C, d_P, and d_T are the interval between each two closeness heads, period heads, and trend heads,

respectively, which are typically set as the corresponding cycle length (in this study, $d_C = 24h$, $d_P = 7 * 24h$, $d_T = 28 * 24h$); n_C, n_P, and n_T are the number of closeness heads, period heads, and trend heads, respectively; $\boldsymbol{Y}'$ is the normalized population inflow from Eq. (*5-2*).

Since temporal patterns may vary across different census tracts, a parametric-matrix-based weighting function is designed to fuse multiple temporal heads, which allows the model to adaptively adjust the weight of each temporal head for each zone. In addition, instead of separately feeding the three temporal heads into the network and fusing their outputs in the last layer, this study fuses them before passing through the whole network (i.e., pre-model fusion), which can substantially mitigate memory load and accelerate the training process while retaining comparable accuracy. The output of the multi-head temporal fusion is:

$$\widetilde{\boldsymbol{Y}}_{:,(t_0-d_E):t_0} = \sum_{k=1}^{n_C} \beta_C^{(k)} \boldsymbol{Y}_C^{(k)} \odot \boldsymbol{W}_C^{(k)} + \sum_{k=1}^{n_P} \beta_P^{(k)} \boldsymbol{Y}_P^{(k)} \odot \boldsymbol{W}_P^{(k)} + \sum_{k=1}^{n_T} \beta_T^{(k)} \boldsymbol{Y}_T^{(k)} \odot \boldsymbol{W}_T^{(k)} \quad (5\text{-}6)$$

where $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{N \times d_E}$ is the fused population inflow; $\boldsymbol{W}_C^{(k)} \in \mathbb{R}^{N \times d_E}$, $\boldsymbol{W}_P^{(k)} \in \mathbb{R}^{N \times d_E}$, and $\boldsymbol{W}_T^{(k)} \in \mathbb{R}^{N \times d_E}$ are zone-specific learnable weights reflecting the importance of different temporal heads for each zone; $\odot$ is the Hadamard product; $\beta_C^{(k)}, \beta_P^{(k)}, \beta_T^{(k)}$ are the Softmax-transformed global weights for each temporal head (i.e., $\boldsymbol{\beta} = \text{softmax}([\boldsymbol{\beta}_C, \boldsymbol{\beta}_P, \boldsymbol{\beta}_T]) \in \mathbb{R}^{n_C+n_P+n_T}$).

Note that the multi-head temporal fusion is only applied to historical population inflow since other auxiliary features do not influence the future population

inflow in such a multi-cycle manner. Alternatively, the new set of features can be rewritten as $\boldsymbol{X}_{:,(t_0-d_E):t_0} = \left[\widetilde{\boldsymbol{Y}}_{:,(t_0-d_E):t_0}, \boldsymbol{Z}_{:,(t_0-d_E):t_0}\right]$.

5.2.2 Multi-view adaptive graph learning

Human travel behavior is a function of numerous factors. Hence, the spatial connection of population flow cannot be simply described by a single adjacency matrix [125]. In this study, a multi-view graph is proposed to incorporate different types of zonal connectivity into the adjacency matrix. Specifically, three pre-defined adjacency matrices are first computed based on different measures. Then, a self-adaptive adjacency matrix is designed, initialized by pre-defined knowledge, and learned end-to-end through stochastic gradient descent. Last, all matrices are concentrated in a mix-hop manner as the final set of adjacency matrices to jointly describe the spatial dependency of population inflow.

Distance closeness ($\boldsymbol{A_D}$): The distance closeness is measured by the pairwise great circle distance between two census tracts. The thresholded Gaussian kernel [205] is employed to transfer the distance to the distance-based adjacency matrix:

$$\boldsymbol{A_D}_{i,j} = \begin{cases} \exp\left(-\frac{\mathrm{dist}(i,j)^2}{\sigma^2}\right), \exp\left(-\frac{\mathrm{dist}(i,j)^2}{\sigma^2}\right) \geq \varepsilon \\ 0, \text{otherwise} \end{cases} \qquad (5\text{-}7)$$

where $\boldsymbol{A_D}_{i,j}$ is the distance-based edge weight between two census tracts i and j; $\mathrm{dist}(.)$ is the great circle distance function; σ is the st.d. of distances; ε is the threshold (set as 0.1 here).

Functional (semantic) similarity ($\boldsymbol{A_F}$): The underlying assumption of functional similarity is that regions with similar functionality are more likely to

present similar travel patterns. To measure the functional similarity, a vector (i.e., static variables $\boldsymbol{S}_i$) is defined for each census tract, including regional demographics, socioeconomics, and POIs (**Table *5-1***). Z-score normalization is applied to each factor across all census tracts. Then, a reciprocal Euclidean distance function is employed to compute the functional similarity adjacency matrix:

$$\boldsymbol{A}_{\boldsymbol{F}_{i,j}} = \begin{cases} \dfrac{1}{\sqrt{\sum_{r=1}^{d_S}\left(S'^{(r)}_i - S'^{(r)}_j\right)}}, i \neq j \\ 1, \text{otherwise} \end{cases} \quad (5\text{-}8)$$

where $\boldsymbol{A}_{\boldsymbol{F}_{i,j}}$ represents the functional similarity between census tracts i and j; $S'^{(r)}_i$ and $S'^{(r)}_j$ denote the r[th] normalized functionality factor of two census tracts i and j.

Origin-destination (OD) volume ($\boldsymbol{A}_{\boldsymbol{OD}}$): OD volume directly measures the travel connectivity between two census tracts. However, limited studies have used it to construct the adjacency matrix perhaps due to data inaccessibility. This study defines the OD-based edge weight as the ratio of average OD volume to the self-loop volume truncated by a maximum of 1:

$$\boldsymbol{A}_{\boldsymbol{OD}_{i,j}} = \min\left(\frac{OD_{i,j}}{OD_{j,j}}, 1\right) \quad (5\text{-}9)$$

where $\boldsymbol{A}_{\boldsymbol{OD}_{i,j}}$ is the OD-based edge weight between two census tracts i and j, and $OD_{i,j}$ is the average OD volume between census tracts i and j in the training set.

Self-adaptive adjacency matrix ($\boldsymbol{A}_{\tilde{\boldsymbol{A}}}$): The concept of the self-adaptive adjacency matrix is borrowed from [134], who defined it as the multiplication of two learnable node embedding matrices $\overrightarrow{\boldsymbol{E}}_1 \in \mathbb{R}^{N \times d_{EB}}$ and $\overleftarrow{\boldsymbol{E}}_2 \in \mathbb{R}^{d_{EB} \times N}$:

$$\boldsymbol{A}_{\check{\boldsymbol{A}}_{i,j}} = \text{SoftMax}\left(\text{ReLU}\left(\vec{\boldsymbol{E}}_1 \overleftarrow{\boldsymbol{E}}_2\right)\right) \quad (5\text{-}10)$$

where $\boldsymbol{A}_{\check{\boldsymbol{A}}_{i,j}}$ represents the self-adaptive edge weight between two census tracts i and j, $\vec{\boldsymbol{E}}_1$ is the source node embedding and $\overleftarrow{\boldsymbol{E}}_2$ is the target node embedding. The ReLU function is used to eliminate weak connections, and the SoftMax function is applied to normalize $\boldsymbol{A}_{\check{\boldsymbol{A}}}$.

The prior knowledge of the adjacency matrix is incorporated into $\boldsymbol{A}_{\check{\boldsymbol{A}}}$ by injecting them into initialized states of $\vec{\boldsymbol{E}}_1$ and $\overleftarrow{\boldsymbol{E}}_2$ [135]. Assume the functional similarity matrix is used as the prior knowledge (Functional similarity is selected here since it outperforms others (Table 5)), initialized states of $\vec{\boldsymbol{E}}_1$ and $\overleftarrow{\boldsymbol{E}}_2$ are computed as:

$$\boldsymbol{A}_{\boldsymbol{OD}} = \boldsymbol{P}\text{Diag}(\boldsymbol{H})\boldsymbol{Q}^{\boldsymbol{T}} \quad (5\text{-}11)$$

$$\vec{\boldsymbol{E}}_{I_1} = \boldsymbol{P}_{:,0:d_{EB}}\text{Diag}\left(\sqrt{\boldsymbol{H}_{0:d_{EB},:}}\right) \quad (5\text{-}12)$$

$$\vec{\boldsymbol{E}}_{I_2} = \text{Diag}\left(\sqrt{\boldsymbol{H}_{0:d_{EB},:}}\right)\boldsymbol{Q}_{:,0:d_{EB}} \quad (5\text{-}13)$$

where $\boldsymbol{P} \in \mathbb{R}^{N\times N}, \boldsymbol{H} \in \mathbb{R}^{N}, \boldsymbol{Q} \in \mathbb{R}^{N\times N}$ are singular value decomposition (SVD) of $\boldsymbol{A}_{\boldsymbol{OD}}$; $\vec{\boldsymbol{E}}_{I_1} \in \mathbb{R}^{N\times d_{EB}}$, $\vec{\boldsymbol{E}}_{I_2} \in \mathbb{R}^{d_{EB}\times N}$ are initialized states of $\vec{\boldsymbol{E}}_1$ and $\overleftarrow{\boldsymbol{E}}_2$; Diag(.) is the diagonal function.

All adjacency matrices are stacked vertically to construct the final adjacency matrices $\widehat{\boldsymbol{A}} = \text{stack}(\boldsymbol{A}_{\boldsymbol{D}}, \boldsymbol{A}_{\boldsymbol{F}}, \boldsymbol{A}_{\boldsymbol{OD}}, \boldsymbol{A}_{\check{\boldsymbol{A}}}) \in \mathbb{R}^{4\times N\times N}$. $\widehat{\boldsymbol{A}}$ is then fed into GCN for multi-view graph convolution. Note that adjacency matrices can be easily removed or added depending on the data accessibility. For instance, if all prior knowledge is unavailable, $\widehat{\boldsymbol{A}}$ is equal to $\boldsymbol{A}_{\check{\boldsymbol{A}}}$, and $\vec{\boldsymbol{E}}_1$ and $\overleftarrow{\boldsymbol{E}}_2$ can be initialized randomly.

5.2.3 Zone-specific Mix-hop GCN (ZMGCN)

At each time t, the spectral-based GCN [140] is applied to the population inflow to exploit signal correlations in the spatial dimension. Letting "$\star_{\mathcal{G}}$" be the graph convolution operator on graph $\mathcal{G}$, the spectral convolution is defined as the multiplication of a signal $\boldsymbol{X}_{:,\boldsymbol{t}}$ with a kernel f_Θ [141]:

$$f_\Theta \star_{\mathcal{G}} \boldsymbol{X}_{:,t} = f_\Theta(\boldsymbol{L})\boldsymbol{X}_{:,t} = f_\Theta(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\boldsymbol{T}})\boldsymbol{X}_{:,t} = \boldsymbol{U}f_\Theta(\boldsymbol{\Lambda})\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}_{:,t} \quad (5\text{-}14)$$

where $\boldsymbol{X}_{:,t} \in \mathbb{R}^{N\times d_I}$ is the signal at time t (e.g., at the first layer, $\boldsymbol{X}_{:,t} = \left[\widetilde{\boldsymbol{Y}}_{:,t}, \boldsymbol{Z}_{:,t}\right] \in \mathbb{R}^{N\times(1+d_F)}$); $\boldsymbol{U} \in \mathbb{R}^{N\times N}$ is the graph Fourier basis and $\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{X}_{:,t}$ is the graph Fourier transform of $\boldsymbol{X}_{:,t}$; $\boldsymbol{U}$ can be obtained by performing the eigenvalue decomposition on the normalized graph Laplacian matrix $\boldsymbol{L}$; $\boldsymbol{L} = \boldsymbol{I_N} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \in \mathbb{R}^{N\times N}$, where $\boldsymbol{I_N} \in \mathbb{R}^{N\times N}$ is an identity matrix, $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is one of an adjacent matric from $\widehat{\boldsymbol{A}}$, and $\boldsymbol{D} \in \mathbb{R}^{N\times N}$ is the diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$; $\boldsymbol{\Lambda} = \mathrm{Diag}([\lambda_0, \dots, \lambda_{N-1}]) \in \mathbb{R}^{N\times N}$ is the diagonal matrix of eigenvalues of $\boldsymbol{L}$.

Since directly performing the eigenvalue decomposition on $\boldsymbol{L}$ is expensive when N is large ($\mathcal{O}(N^2)$), this study adopts the Chebyshev polynomials [206] to approximate the kernel f_Θ. Each order of Chebyshev polynomial is viewed as a hop in the graph with a specific structure. The graph convolution can be rewritten as:

$$f_\Theta \star_{\mathcal{G}} \boldsymbol{X}_{:,t} = f_\Theta(\boldsymbol{L})\boldsymbol{X}_{:,t} \approx \sum_{k=0}^{K-1} T_k\left(\tilde{\boldsymbol{L}}\right)\boldsymbol{X}_{:,t}\boldsymbol{W}_k \quad (5\text{-}15)$$

where $\boldsymbol{W} \in \mathbb{R}^{K\times d_I\times d_O}$ is the polynomial weights and $\boldsymbol{W}_k \in \mathbb{R}^{d_I\times d_O}$ is its k$^{\text{th}}$-hop weight matrix; d_I and d_O are the dimension of input and output, respectively; $\tilde{\boldsymbol{L}} = \frac{2}{\lambda_{max}}\boldsymbol{L} - \boldsymbol{I_N}$ is the scaled Laplacian matrix, where $\lambda_{max} = \max(\lambda_0, \dots, \lambda_{N-1})$;

$T_k(\tilde{\boldsymbol{L}}) \in \mathbb{R}^{N\times N}$ is the k[th] Chebyshev polynomial approximation, which can be recursively computed as: $T_k(\tilde{\boldsymbol{L}}) = 2\tilde{\boldsymbol{L}}T_{k-1}(\tilde{\boldsymbol{L}}) - T_{k-2}(\tilde{\boldsymbol{L}}), T_0(\tilde{\boldsymbol{L}}) = \boldsymbol{I_N}, T_1(\tilde{\boldsymbol{L}}) = \tilde{\boldsymbol{L}}$.

One issue of Eq. (*5-15*) is that for each hop, all zones share the same weight matrix $\boldsymbol{W}_k$, which is not optimal for population inflow forecasting since mobility patterns among census tracts are diverse. To increase the local diversity, this study introduces zone-specific parameterization in graph convolution [201]. Specifically, $\boldsymbol{W}_k$ is modified as the multiplication of a node embedding matrix $\boldsymbol{E}_{\mathcal{G}} \in \mathbb{R}^{N\times d_{EB}}$ and a node-aware weight matrix $\boldsymbol{\Psi}_k \in \mathbb{R}^{d_{EB}\times d_I\times d_O}$:

$$f_\Theta \star_{\mathcal{G}} \boldsymbol{X}_{:,t} \approx \sum_{k=0}^{K-1} T_k(\tilde{\boldsymbol{L}})\boldsymbol{X}_{:,t}\boldsymbol{E}_{\mathcal{G}}\boldsymbol{\Psi}_k \qquad (5\text{-}16)$$

The reason for introducing a $\boldsymbol{E}_{\mathcal{G}}$ rather than directly expanding the size of $\boldsymbol{W}_k$ to $\mathbb{R}^{N\times d_I\times d_O}$ is to reduce parameter numbers, which can enhance computational efficiency, particularly for large graphs. Also, $\boldsymbol{E}_{\mathcal{G}}$ is allowed to be shared among hops and adjacency matrices to mitigate the memory burden. Similar to the previous initialization of node embedding matrices $\overrightarrow{\boldsymbol{E}}_1$ and $\overleftarrow{\boldsymbol{E}}_2$, the $\boldsymbol{E}_{\mathcal{G}}$ is initialized by their static features $\boldsymbol{S}$. The initialized state of $\boldsymbol{E}_{\mathcal{G}}$ can be computed as:

$$\boldsymbol{P}_S \mathrm{Diag}(\boldsymbol{H}_S)\boldsymbol{Q}_S{}^{\boldsymbol{T}} = \boldsymbol{S} \qquad (5\text{-}17)$$

$$\boldsymbol{E}_{I_{\mathcal{G}}} = \mathrm{ReLU}(\boldsymbol{S}\boldsymbol{Q}_S\boldsymbol{W}_S + \boldsymbol{b}_S) \qquad (5\text{-}18)$$

where $\boldsymbol{E}_{I_{\mathcal{G}}}$ is the initialized state of $\boldsymbol{E}_{\mathcal{G}}$; $\boldsymbol{S} \in \mathbb{R}^{N\times d_S}$ are the static variables; $\boldsymbol{P}_S \in \mathbb{R}^{N\times d_{EB}}$, $\boldsymbol{H}_S \in \mathbb{R}^{d_{EB}}$, $\boldsymbol{Q}_S \in \mathbb{R}^{d_S\times d_{EB}}$ are outputs of principal component analysis (PCA) of $\boldsymbol{S}$; $\boldsymbol{S}\boldsymbol{Q}_S \in \mathbb{R}^{N\times d_{EB}}$ is the projection of the first k principal components of $\boldsymbol{S}$; $\boldsymbol{W}_S \in \mathbb{R}^{d_{EB}\times d_{EB}}$ and $\boldsymbol{b}_S \in \mathbb{R}^{d_{EB}}$ are the weight and bias.

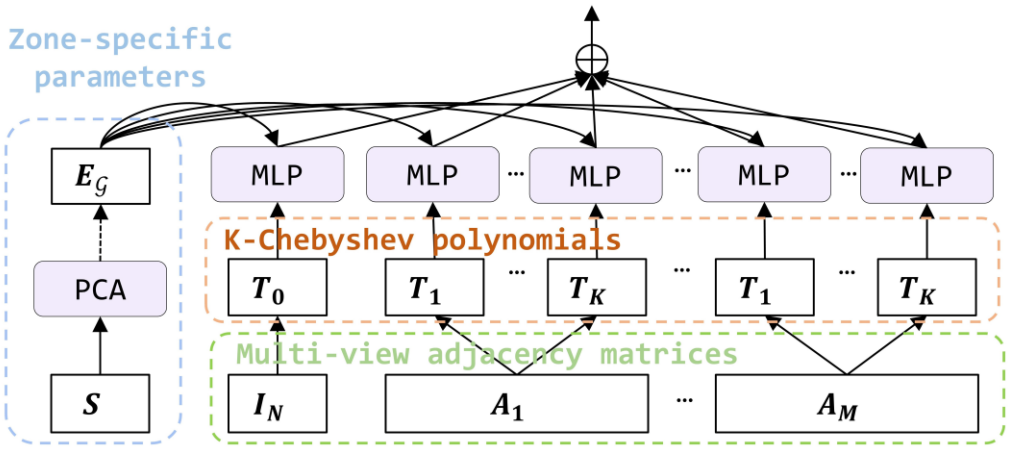


Figure 5-3 The ZMGCN architecture

The aforementioned GCN can be easily generalized to multi-view graphs by computing Chebyshev polynomials for each adjacency matrix and fusing them with learnable weights. Instead of directly summing up the outputs of all hops, this study employs a mix-hop manner (**Figure 5-3**) for hop fusion with hop-wise learnable weight. Moreover, to avoid repeatedly computing I_N, I_N is separately computed from other hops whose order $k > 1$. The multi-view GCN can be written as:

$$f_\Theta \star_{\mathcal{G}} \boldsymbol{X}_{:,t} \approx \gamma_0 \boldsymbol{I_N} \boldsymbol{X}_{:,t} \boldsymbol{E}_{\mathcal{G}} \boldsymbol{\Psi}_0 + \sum_{m=1}^{M} \sum_{k=1}^{K-1} \gamma_k^{(m)} T_k\left(\tilde{\boldsymbol{L}}^{(m)}\right) \boldsymbol{X}_{:,t} \boldsymbol{E}_{\mathcal{G}} \boldsymbol{\Psi}_k^{(m)} \qquad (5\text{-}19)$$

where $\boldsymbol{\Psi}_k^{(m)}$ is the zone-aware weight matrix of the kth Chebyshev polynomial approximation for the m[th] adjacency matrix; M is the number of adjacency matrices; $\boldsymbol{\Psi}_0$ is the zone-aware weight matrix for the self-loop matrix; $\tilde{\boldsymbol{L}}^{(m)}$ is the scaled Laplacian matrix derived from the m[th] adjacency matrix ($\tilde{\boldsymbol{L}}^{(m)} = \frac{2}{\lambda_{max}^{(m)}}\left(\boldsymbol{I_N} - \boldsymbol{D}^{(m)^{-\frac{1}{2}}} \boldsymbol{A}^{(m)} \boldsymbol{D}^{(m)^{-\frac{1}{2}}}\right) - \boldsymbol{I_N}$); γ_0 and $\gamma_k^{(m)}$ are the Softmax-transformed fusion weights for each hop. In sum, Eq. (*5-19*) can be viewed as a high-level representation of

population inflow exploiting the mixed information from $(K-1)$-hop neighborhoods, where neighborhoods are determined by different adjacency matrices.

5.2.4 Graph convolutional recurrent neural network (GCRNN)

This study integrates the ZMGCN into the GRU-based RNN, named the graph convolutional recurrent neural network (GCRNN), to jointly capture spatiotemporal dependencies of population inflow. The form of the GCRNN is as follows:

$$\boldsymbol{\Gamma}_{u_{:,t}} = \sigma\left(f_{\Theta_u} \star_{\boldsymbol{\mathcal{G}}} \left[\boldsymbol{X}_{:,t}, \boldsymbol{H}_{:,t-1}\right] + \boldsymbol{E}_{\mathcal{G}}\boldsymbol{b}_u\right) \quad (5\text{-}20)$$

$$\boldsymbol{\Gamma}_{r_{:,t}} = \sigma\left(f_{\Theta_r} \star_{\boldsymbol{\mathcal{G}}} \left[\boldsymbol{X}_{:,t}, \boldsymbol{H}_{:,t-1}\right] + \boldsymbol{E}_{\mathcal{G}}\boldsymbol{b}_r\right) \quad (5\text{-}21)$$

$$\widetilde{\boldsymbol{H}}_{:,t} = \tanh\left(f_{\Theta_H} \star_{\boldsymbol{\mathcal{G}}} \left[\boldsymbol{X}_{:,t}, \boldsymbol{\Gamma}_{r_{:,t}} \odot \boldsymbol{H}_{:,t-1}\right] + \boldsymbol{E}_{\mathcal{G}}\boldsymbol{b}_c\right) \quad (5\text{-}22)$$

$$\boldsymbol{H}_{:,t} = \left(1 - \boldsymbol{\Gamma}_{u_{:,t}}\right) \odot \boldsymbol{H}_{:,t-1} + \boldsymbol{\Gamma}_{u_{:,t}} \odot \widetilde{\boldsymbol{H}}_{:,t} \quad (5\text{-}23)$$

where $t = [t_0 - d_E, t_0 - d_E + 1, \ldots, t_0 - 1]$ is the time index of each step in the historical window and t_0 is the current time; $\star_{\boldsymbol{\mathcal{G}}}$ denotes the ZMGCN defined in Eq. (*5-19*) and $f_{\Theta_u}, f_{\Theta_r}, f_{\Theta_H}$ are corresponding kernels; $\boldsymbol{X}_{:,t} \in \mathbb{R}^{N \times d_I}$ is the input signal at time t; $\boldsymbol{H}_{:,t} \in \mathbb{R}^{N \times d_H}$ is the hidden state at time t, which is a linear combination of the previous state $\boldsymbol{H}_{:,t-1}$ and the candidate's state $\widetilde{\boldsymbol{H}}_{:,t}$; $\boldsymbol{\Gamma}_{u_{:,t}} \in \mathbb{R}^{N \times d_H}$ and $\boldsymbol{\Gamma}_{r_{:,t}} \in \mathbb{R}^{N \times d_H}$ are the update gate and reset gate, respectively; $\boldsymbol{E}_{\mathcal{G}} \in \mathbb{R}^{N \times d_{EB}}$ is the node embedding matrix and $\boldsymbol{b}_u, \boldsymbol{b}_r, \boldsymbol{b}_c \in \mathbb{R}^{d_{EB} \times d_H}$ are zone-aware biases.

Temporal patterns of population inflow are conditional on static features of census tracts. However, directly appending static variables to $\boldsymbol{X}_t$ may pollute the sequence data with non-sequential information [207]. To avoid it, this study includes the effects of static variables by using their principal components as the initial hidden state of GCRNN:

$$\boldsymbol{H}_{:,t_0-d_E-1} = \boldsymbol{SQ}_S\boldsymbol{W}_0 + \boldsymbol{b}_0 \quad (5\text{-}24)$$

where $\boldsymbol{H}_{:,t_0-d_E-1} \in \mathbb{R}^{N\times d_H}$ is the initial state of GCRNN, $\boldsymbol{SQ}_S \in \mathbb{R}^{N\times d_{EB}}$ are principal components of $\boldsymbol{S}$ derived from Eq. (*5-18*), $\boldsymbol{W}_0 \in \mathbb{R}^{d_{EB}\times d_H}$ and $\boldsymbol{b}_0 \in \mathbb{R}^{d_H}$ are the weight and bias.

The adjacency matrix may not accurately reflect the real zonal connectivity, which would induce irrelevant noise into the graph convolution. Hence, a residual connection is included along the GCRNN using the pure GRU as a bypass path, i.e., replacing the $f_\Theta \star_{\boldsymbol{\mathcal{G}}}$ in Eqs. (*5-20*) to (*5-24*) as $\boldsymbol{W}_u \in \mathbb{R}^{d_I\times d_O}$, $\boldsymbol{W}_r \in \mathbb{R}^{d_I\times d_O}$, and $\boldsymbol{W}_H \in \mathbb{R}^{d_I\times d_O}$, respectively. Assume $\ddot{\boldsymbol{H}}_{:,t} \in \mathbb{R}^{N\times d_H}$ is the output of the pure GRU, the final output of the GCRNN can be expressed as:

$$\breve{\boldsymbol{H}}_{:,t} = \sigma(\boldsymbol{\alpha}_t)\boldsymbol{H}_{:,t} + (1-\sigma(\boldsymbol{\alpha}_t))\ddot{\boldsymbol{H}}_{:,t} \quad (5\text{-}25)$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{d_E}$ is the fusion weight and $\boldsymbol{\alpha}_t$ is the weight for the tth step in the historical window.

The GCRNN can be easily extended to multiple layers by stacking the modules vertically and using the output of the last layer as the input of the next layer. For example, letting p be the layer index and P be # layers, the input of the pth layer $\boldsymbol{X}_{:,t}^{(p)}$ is equal to the hidden states of the (p-1)th layer $\breve{\boldsymbol{H}}_{:,t}^{(\boldsymbol{p-1})}$, where $p = 1,2,\dots,P$.

5.2.5 Multi-step output

Traditional RNN generates the prediction based on its last hidden state (i.e., $\boldsymbol{H}_{:,t_0-1}$), which may lose accuracy when making multi-step predictions due to the memory vanishing in long sequences. The attention-based mechanism has been proposed by including information from all hidden states [126]. This study takes inspiration from

it by including all hidden states to generate multi-step predictions. Besides, instead of employing an RNN-based decoder to recursively generate multi-step outputs, this study directly places a 2D CNN to transform hidden states into normalized d_D-step population inflow. Then, an inverse normalization function is applied to the normalized outcome to scale it back to the original representation:

$$\widehat{\boldsymbol{Y}}_{:,t_0:(t_0+d_D)} = \text{Inverse}(\text{Conv2D}(\boldsymbol{H}_{:,(t_0-d_E):t_0})) \quad (5\text{-}26)$$

where $\boldsymbol{H}_{:,(t_0-d_E):t_0} \in \mathbb{R}^{d_E \times N \times d_H}$ are all hidden states from GCRNN; $\text{Conv2D}(.)$ is a 2D CNN with kernel size = (1, d_H), # input channels = d_E and # output channels = d_D; $\text{Inverse}(.)$ is the inverse normalization function; $\widehat{\boldsymbol{Y}}_{:,t_0:(t_0+d_D)} \in \mathbb{R}^{N \times d_D}$ is the forecasted d_D-step-ahead population inflow for all census tracts.

The mean absolute error (MAE) is used as the loss function:

$$\mathcal{L}(\mathcal{F}_\theta) = \sum_{i=1}^{N} \sum_{t=t_0}^{t_0+d_D-1} \left|\hat{y}_{i,t} - y_{i,t}\right| \quad (5\text{-}27)$$

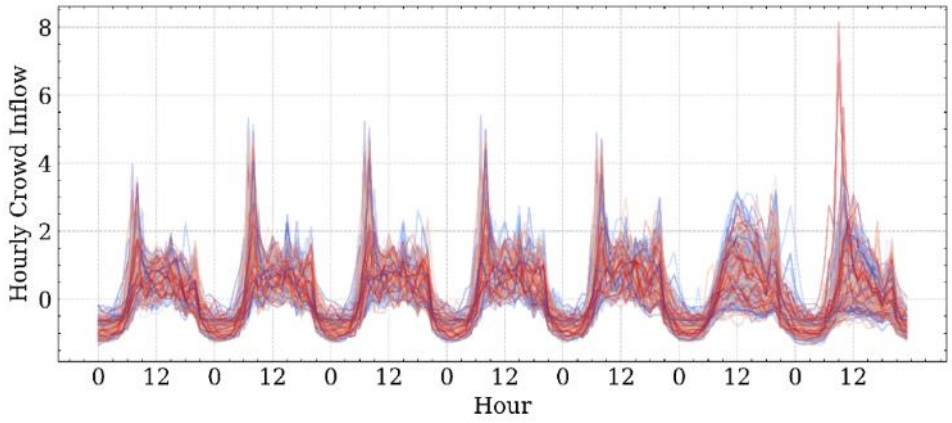
where $y_{i,t}$ is the population inflow of census tract *i* at time t and $\hat{y}_{i,t}$ is its prediction; $\mathcal{F}_\theta$ represents all parameters in the nonlinear mapping $\mathcal{F}(.)$, which can be updated by the model according to their stochastic gradients and learning rate.

5.3 Experiments

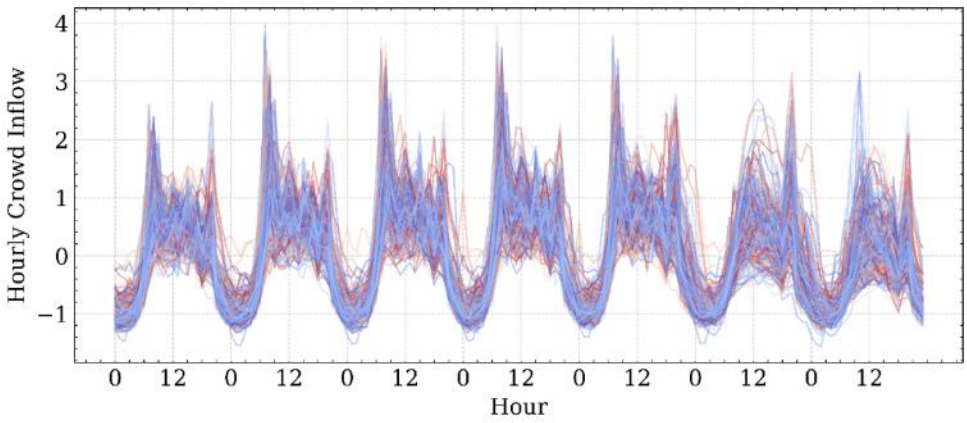
5.3.1 Data description

Current public spatiotemporal datasets are mainly related to traffic flow, while a comprehensive citywide population flow dataset which includes various external variables is absent. This study collected and prepared such datasets and made them public on GitHub (https://github.com/SonghuaHu-UMD/MultiSTGraph). The

population inflow is calculated using data from SafeGraph [22], a data company that aggregates anonymized MDLD in the US. All MDLD are de-identified and contain no private personal information. Specifically, the Core Places US dataset is used to obtain the geographical coordinates of each POI. Then, the Weekly Places Patterns (v2) dataset is used to extract the POI-level hourly visit. Last, the hourly visit is aggregated at a census tract level. The weekly OD volume is extracted as well for graph adjacency matrix building. Finally, two cities are selected as case studies, namely Washington, D.C., and Baltimore. Their data statistics are reported in **Table *5-1***. Visualization of the census tract-level weekly average population flow time series is shown in **Figure 5-4**.



(a) Baltimore



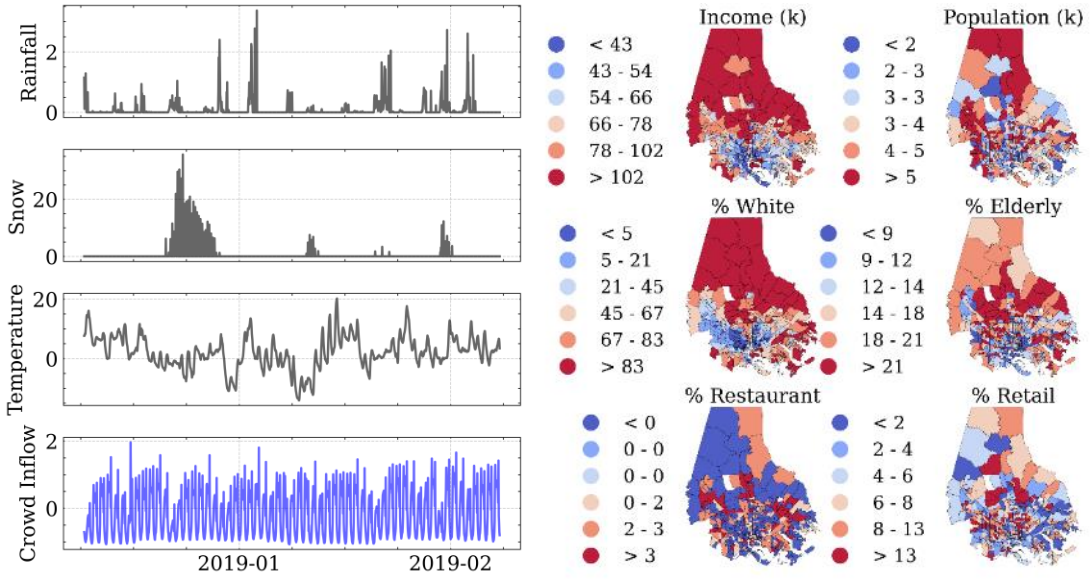
(b) Washington, D.C.

Figure 5-4 Normalized time series of weekly average population inflow

A set of auxiliary variables are collected, including time-varying variables (holiday, weekend, precipitation, temperature, snowfall) and static variables (demographics, socioeconomics, land use). An illustration of some of the collected variables is depicted in **Figure 5-5**. Among them, socioeconomics and demographics are from the 2015–2019 ACS of the US Census Bureau. POI features are from SafeGraph. Partisanship is from the 2016 presidential election result provided by the MIT election lab. Weather conditions are from NOAA's National Centers for Environmental Information. Datasets are split into training sets, validation sets, and test sets according to chronological order. The split ratio is 7:1.5:1.5 for both datasets.

Table 5-1 Training dataset statistics for Multi-ATGCN

	Washington, D.C.	**Baltimore (and surrounding counties)**
Date Range	01/01/2019 - 05/31/2019	
# Zones	237	403
# Samples	858,888	1,460,472
Sample Rate	1 hour	
Input length (d_E)	24 hours	
Output length (d_D)	3 hours, 6 hours, 12 hours, 24 hours	
Mean	30.169	14.410
Standard deviation	84.023	29.300
Auxiliary time-varying variables ($\boldsymbol{Z}$)	Holiday, weekend, precipitation, temperature, snowfall	
Static variables ($\boldsymbol{S}$)	*Demographics*: % non-Hispanic Whites, % African Americans, % Asians, % Hispanics, % males, % residents 18-44, % residents 45-64, % residents >65. *Socioeconomics*: Total population, % urbanized populations, median household income, % Democrats, % Republicans, % highly-educated residents. *Land use*: Area, # residential POIs, # retail trade, # personal and public services, # educational institutions, # recreation, # restaurants, # other POIs.	



(a) Time-varying (b) Static

Figure 5-5 Illustration of external variables in Baltimore

5.3.2 Baselines for comparison

The performance of Multi-ATGCN is extensively compared with a variety of widely used baselines and state-of-the-art models, including:

- FNN: A simple two-layer FNN with ReLU as the activation function.
- LSTM [121]: A sequence-to-sequence (S2S) RNN using LSTM as recurrent units.
- GRU [208]: A S2S RNN using GRU as recurrent units.
- STGCN [133]: A integration of spatial GCN and temporal 1D CNN;
- DCRNN [205]: A diffusion convolutional RNN that models temporal dynamics using GRU and captures spatial dependency via diffusion GCN.
- ASTGCN [204]: An attention-based spatiotemporal GCN, which considers multiple temporal heads and integrates attention mechanisms into GCN.

- GWNET [134]: A spatiotemporal GCN that integrates self-adaptive diffusion GCN for spatial modeling with stacked dilated 1D CNNs for temporal modeling.
- AGCRN [201]: An adaptive graph convolutional RNN that enhances graph convolutions by zone-specific parameters and self-adaptive graph learning.
- GMAN [209]: A multi-attention GCN that leverages the node2vec algorithm [210] to learn node structural information while performing spatiotemporal attention mechanisms.
- MTGNN [135]: A general GCN framework for multivariate time series forecasting, which includes a graph generation module for graph self-learning, a mix-hop propagation layer for spatial modeling, and a dilated inception layer for temporal modeling.
- STGODE [211]: A spatiotemporal graph ordinary differential equation network which captures spatiotemporal dynamics through a tensor-based ordinary differential equation.
- STG-NCDE [212]: A spatiotemporal graph neural controlled differential equation (NCDE) which connects two NCDEs for spatial and temporal processing.

5.3.3 Experiment settings

All baselines, including the Multi-ATGCN, are implemented in Python with PyTorch 1.10.2 and executed on servers with NVIDIA Tesla T4 GPU. The Adam optimizer is employed to minimize the model loss with the learning rate decaying (Starting from 0.003, decaying by 75% once the number of epochs reaches 5, 10, 20, and 30 epochs). Each model is run 50 epochs and an early stop strategy with a patience of 10 is used by monitoring the loss in the validation set. The batch size is set as 16.

Gradient clipping (maximum norm = 5) is performed during the training process to mitigate exploding gradients. Dropout (ratio = 0.1) is applied before the output layer to mitigate overfitting. The best hyperparameters are chosen using the asynchronous successive halving algorithm (SHA) [213]. For easy comparison among different models, the code and data formats follow the framework proposed by [214]. Codes are available at: https://github.com/SonghuaHu-UMD/MultiSTGraph.

This study deploys three widely used metrics to evaluate model performances on the testing set, including MAE, MAPE, and RMSE. Each experiment is repeated ten times with different random seeds and average metrics are reported. Similarly to [125], this study set the lower bound of hourly population inflow as 10. Alternatively, only data points with a value greater than 10 in the testing set are used for model evaluation. Low-demand scenarios are less important for travel demand modeling, regional planning, or other real-world applications. In addition, including these small data points would substantially increase the MAPE, which may gloss over the real model performance (**Figure *5-8***).

5.3.4 Implementation details of baselines

1) **FNN**: A two-layer FNN with a hidden size of 128 and using the ReLU function between the two layers as activation.

2) **LSTM**/**GRU**: The LSTM and GRU are implemented in an S2S manner to recursively generate multi-step-ahead output. The encoder and decoder follow the same structure, each contains 2 layers of LSTM (GRU) with 64 hidden units. An FNN is applied to the output of the RNN at each time step to convert it to the final

prediction. Models are trained using the teacher-forcing strategy with a ratio of 0.5. The learning rate is set to 0.01 with a decaying ratio of 0.1 in 5, 20, and 40 epochs.

3) **STGCN** [133]: Two spatiotemporal convolutional (ST-Conv) blocks are stacked, followed by an output layer containing two temporal CNN and one FNN. The channels of three layers in ST-Conv blocks are [64, 32, 1] and [64, 32, 128], respectively. Both the graph convolution and temporal convolution kernel sizes are set to 3. Similarly to this study, the Chebyshev polynomials approximation is used for GCN. The learning rate is set to 0.001 with a decaying ratio of 0.7 in every 5 epochs.

4) **DCRNN** [205]: DCRNN is implemented in an S2S manner to recursively generate multi-step-ahead output. The encoder and decoder follow the same structure, each contains two layers of diffusion convolutional GRUs with 64 hidden units. The dual random walk approach is adopted for the diffusion process. The learning rate is set to 0.01 with a decaying ratio of 0.1 in 5, 20, and 40 epochs.

5) **ASTGCN** [204]: ASTGCN builds three attention-based STGCN for three types of temporal heads (i.e., close, period, and trend) respectively, and fuses their outputs using a parametric-matrix-based weighting function. The hidden units for graph convolution and temporal convolution are set to 64 with a kernel size of three. The learning rate is set to 0.0001.

6) **GWNET** [134]*:* GWNET stacks 2 spatial-temporal layers. Each spatial-temporal layer is constructed by a graph convolution layer and a gated temporal convolution layer. The hidden units for all convolution networks are set to 32 with a kernel size of 2. 1×1 convolution with output channels of 256 is set for skip

connection. Two 2D CNNs are stacked as the final output layers by first projecting the channels to 512 and then downsampling to the output dimension.

7) **AGCRN** [201]: The AGCRN consists of an encoder and a 2D CNN which is used to replace the decoder. The encoder is constructed by 2 layers of adaptive zone-specific graph convolution GRU with 64 hidden units. The dimension of node embedding is set as 10, and the order of Chebyshev polynomials is set as 2. The learning rate is set to 0.003 with a decaying ratio of 0.75 in 5, 15, 30, and 40 epochs.

8) **GMAN** [209]*:* First, the node2vec algorithm is used to learn a node embedding vector with a dimension of 64 and combined it with temporal embedding vectors. For node2vec, the number of random walks is set as 100 with 50 iterations. Then, an encoder and a decoder both with 2 ST-Attention blocks, one transform attention layer (number of attention heads is set as 2), and 2 FNNs are constructed to generate the final output. The learning rate is set to 0.001 with a decaying ratio of 0.7 when the loss does not decrease for 5 epochs.

9) **MTGNN** [135]*:* In MTGNN, temporal convolution and graph convolution are interleaved with each other to capture temporal and spatial dependencies respectively. The channel of convolutional layers is set as 32, and # layers is set as 3. The temporal inception layer consists of four filter sizes, viz. 1×2, 1×3, 1×6, and 1×7. The output module consists of two 1×1 convolution layers. MTGNN also includes a graph self-learning module with a node embedding dimension of 40.

10) **STGODE** [211]: The hidden dimensions of temporal dilation convolution blocks are set to 64, 32, 64, and 3 Spatial-Temporal Graph ODE blocks are contained in each layer. The regularized hyperparameter is set to 0.8. The thresholds of the

spatial adjacency matrix are set to 10 and 0.5 respectively, and the threshold of the semantic adjacency matrix is set to 0.6. The model is trained using Adam optimizer with a learning rate of 0.01.

11) **STG-NCDE** [212]: The order of Chebyshev polynomials is set to 2 and the zone embedding size is set to 8. The dimensionality of the hidden vector is set to 64. The learning rate is set to 0.001 and the weight decay is 0.001.

5.4 Baseline comparison

Table *5-2* shows the average performances of all models on the two testing datasets, with the forecasting horizon varying from 3 to 24 hours. Overall, Multi-ATGCN achieves state-of-the-art results on most of the tasks, outperforming all baselines over different horizons. Compared with the best baseline (underlined), Multi-ATGCN yields a 1.9-2.8%, 2.8-3.1%, 3.8-7.4%, and 5.1-6.4% reduction in MAE for predicting over 3, 6, 12, and 24 horizons, respectively. Such a reduction is also observed in other metrics but is slightly less salient. One noteworthy finding is that the performance improvement of Multi-ATGCN increases with the longer prediction horizons, which can be explained by several reasons. First, long-horizon prediction relies more on period and trend temporal information; hence, models that include multiple temporal heads such as Multi-ATGCN (ASTGCN as well) become superior in long-horizon scenarios. Second, long-horizon prediction is more complex. Hence, models that can integrate more information such as multi-view spatial structures and auxiliary effects may gain more benefits.

Among baselines, the performance of the same model varies substantially across datasets and forecasting horizons. For example, ASTGCN performs poorly in

3-step prediction but outperforms most models in 24-step prediction. AGCRN, on the other hand, performs well in 3-step prediction but degrades significantly at longer horizons. GMAN ranks high in Baltimore city data but its performance is less robust when shifting to the D.C. area. The number of hyperparameters in GMAN is large, which may cause the GMAN to be unstable across different scenarios. Despite some variance, FNN and simple RNNs such as LSTM and GRU constantly exhibit the poorest performance while models such as MTGNN and STGCN always rank in the first tier, which is consistent with previous comparison studies [214].

Table 5-2 Model performances comparison (population inflow forecasting)

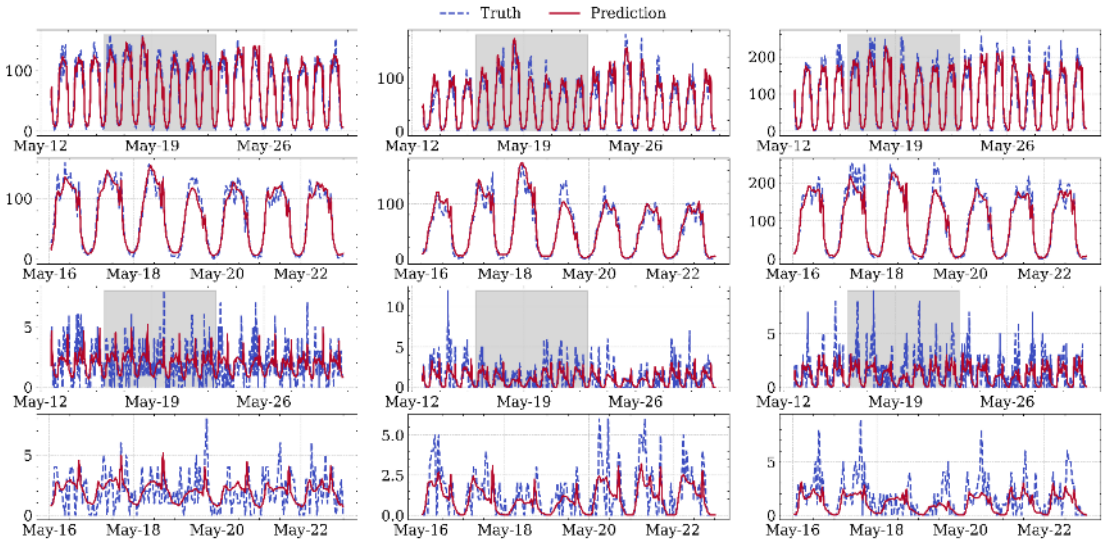
Model	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
	Baltimore			**Washington, D.C.**		
Horizon = 3						
FNN	9.82 (26.7%)	19.02 (29.9%)	0.29 (19.0%)	14.73 (30.3%)	44.53 (36.6%)	0.29 (18.6%)
LSTM	8.34 (13.7%)	17.56 (24.0%)	0.27 (11.3%)	14.38 (28.6%)	48.06 (41.3%)	0.29 (18.4%)
GRU	8.21 (12.3%)	17.29 (22.8%)	0.26 (9.9%)	14.11 (27.2%)	47.05 (40.0%)	0.28 (17.9%)
ASTGCN	8.14 (11.5%)	16.09 (17.1%)	0.26 (10.7%)	12.16 (15.5%)	33.71 (16.2%)	0.27 (12.9%)
DCRNN	7.98 (9.8%)	15.96 (16.4%)	0.26 (8.2%)	12.50 (17.9%)	38.41 (26.5%)	0.27 (12.4%)
STG-NCDE	7.96 (9.5%)	14.06 (5.1%)	0.25 (5.7%)	11.89 (13.6%)	33.35 (15.4%)	0.26 (12.7%)
GWNET	7.76 (7.3%)	14.28 (6.5%)	0.25 (5.9%)	11.51 (10.8%)	31.77 (11.1%)	0.25 (6.1%)
GMAN	7.53 (4.4%)	14.09 (5.3%)	0.24 (2.3%)	10.79 (4.9%)	30.16 (6.4%)	0.24 (1.1%)
STGODE	7.47 (3.6%)	13.79 (3.3%)	0.25 (3.4%)	11.10 (13.4%)	32.58 (13.4%)	0.26 (10.8%)
STGCN	7.42 (2.9%)	13.74 (2.9%)	0.24 (2.2%)	11.46 (10.3%)	32.41 (12.9%)	0.25 (6.5%)
AGCRN	7.40 (2.7%)	13.94 (4.2%)	0.24 (2.1%)	10.47 (1.9%)	28.54 (1.1%)	0.24 (1.0%)
MTGNN	7.40 (2.8%)	13.63 (2.1%)	0.24 (1.7%)	10.49 (2.1%)	28.64 (1.4%)	0.24 (1.3%)
Multi-ATGCN	**7.20**	**13.34**	**0.24**	**10.27**	**28.23**	**0.23**
Horizon = 6						
FNN	10.44 (28.3%)	20.46 (27.7%)	0.31 (21.0%)	16.38 (31.7%)	51.63 (39.3%)	0.32 (18.9%)

LSTM	8.77 (14.6%)	18.40 (19.6%)	0.28 (12.1%)	14.41 (22.3%)	51.12 (38.7%)	0.29 (11.4%)
STG-NCDE	8.76 (14.5%)	15.84 (6.6%)	0.27 (8.1%)	12.82 (12.6%)	34.31 (8.7%)	0.29 (9.1%)
GRU	8.58 (12.7%)	18.00 (17.8%)	0.28 (10.1%)	13.84 (19.1%)	46.24 (32.2%)	0.28 (9.4%)
DCRNN	8.27 (9.5%)	16.62 (11.0%)	0.27 (7.1%)	12.65 (11.5%)	38.05 (17.7%)	0.27 (5.8%)
ASTGCN	8.25 (9.2%)	16.42 (9.9%)	0.27 (8.8%)	11.87 (5.7%)	32.86 (4.7%)	0.27 (4.9%)
GWNET	8.25 (9.2%)	15.96 (7.3%)	0.27 (6.8%)	12.07 (7.2%)	33.28 (5.9%)	0.26 (1.3%)
AGCRN	8.23 (9.0%)	15.79 (6.2%)	0.27 (7.0%)	11.95 (6.3%)	34.37 (8.9%)	0.27 (3.7%)
STGODE	8.03 (6.8%)	15.77 (6.2%)	0.27 (7.2%)	11.83 (5.4%)	33.19 (5.6%)	0.27 (4.2%)
STGCN	7.87 (4.8%)	15.09 (1.9%)	0.26 (4.5%)	11.93 (6.1%)	33.29 (5.9%)	0.27 (3.5%)
GMAN	7.82 (4.2%)	15.37 (3.7%)	0.26 (5.4%)	12.25 (8.6%)	36.50 (14.2%)	0.26 (1.3%)
MTGNN	7.73 (3.1%)	14.92 (0.8%)	0.25 (2.3%)	11.52 (2.8%)	33.80 (7.3%)	0.26 (1.0%)
Multi-ATGCN	**7.49**	**14.80**	**0.25**	**11.20**	**31.33**	**0.26**
Horizon = 12						
FNN	11.29 (31.4%)	22.22 (31.4%)	0.32 (19.9%)	18.24 (36.6%)	58.45 (44.2%)	0.33 (20.5%)
LSTM	9.12 (15.1%)	19.14 (20.3%)	0.28 (8.2%)	13.74 (15.8%)	44.73 (27.1%)	0.28 (6.8%)
GRU	9.10 (14.9%)	19.10 (20.2%)	0.28 (8.0%)	13.76 (15.9%)	42.44 (23.2%)	0.29 (8.6%)
DCRNN	9.04 (14.3%)	18.26 (16.5%)	0.28 (7.3%)	13.34 (13.3%)	38.31 (14.9%)	0.28 (7.3%)
AGCRN	9.01 (14.0%)	17.52 (13.0%)	0.28 (7.0%)	12.62 (8.3%)	35.74 (8.8%)	0.27 (3.1%)
STG-NCDE	8.95 (13.4%)	16.87 (9.6%)	0.28 (6.5%)	13.10 (11.7%)	37.63 (13.4%)	0.29 (9.8%)
ASTGCN	8.67 (10.7%)	17.24 (11.6%)	0.28 (6.5%)	13.13 (11.9%)	35.85 (9.1%)	0.28 (5.9%)
STGODE	8.53 (9.2%)	16.67 (8.5%)	0.27 (4.9%)	12.51 (7.5%)	36.62 (11.0%)	0.28 (5.7%)
GMAN	8.41 (7.9%)	16.64 (8.3%)	0.27 (4.5%)	13.01 (11.1%)	37.22 (12.4%)	0.28 (5.4%)
GWNET	8.39 (7.7%)	16.49 (7.5%)	0.26 (1.7%)	12.58 (8.1%)	35.08 (7.1%)	0.27 (2.8%)
STGCN	8.10 (4.3%)	16.24 (6.1%)	0.26 (1.6%)	12.49 (7.4%)	36.40 (10.5%)	0.27 (2.9%)
MTGNN	8.05 (3.8%)	15.99 (4.6%)	0.26 (1.3%)	12.83 (9.9%)	35.81 (9.0%)	0.28 (5.7%)
Multi-ATGCN	**7.75**	**15.25**	**0.26**	**11.57**	**32.59**	**0.26**
Horizon = 24						
FNN	11.91 (34.0%)	23.41 (32.4%)	0.34 (23.8%)	19.50 (37.2%)	63.93 (47.0%)	0.35 (20.5%)

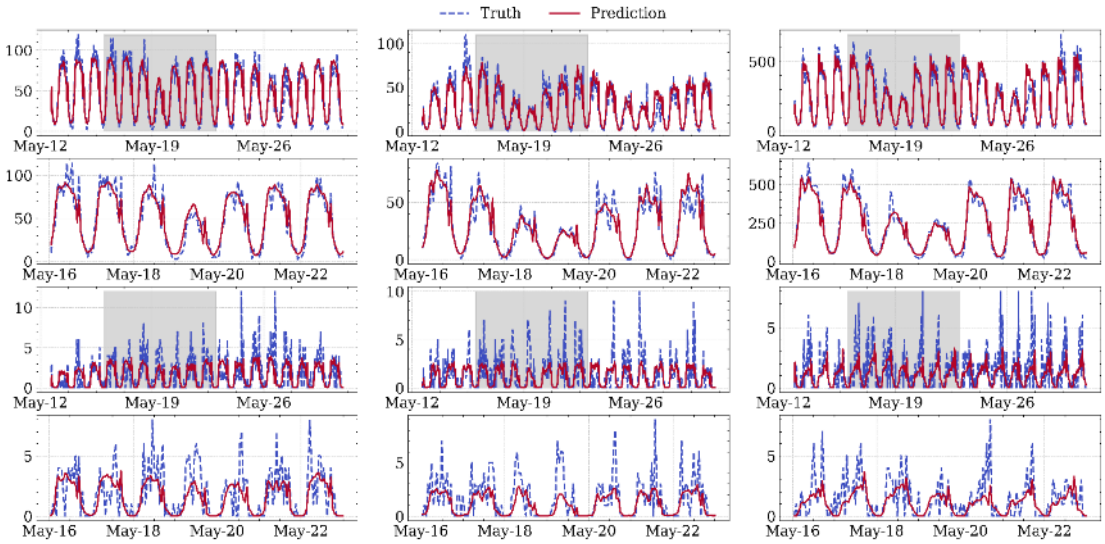
LSTM	9.81 (19.8%)	20.58 (23.1%)	0.30 (14.1%)	16.01 (23.5%)	52.38 (35.3%)	0.32 (12.5%)
GRU	9.49 (17.2%)	19.92 (20.5%)	0.29 (11.2%)	15.31 (20.0%)	47.52 (28.7%)	0.31 (11.5%)
AGCRN	9.31 (15.5%)	18.74 (15.5%)	0.29 (9.8%)	14.70 (16.7%)	41.09 (17.6%)	0.30 (9.1%)
DCRNN	9.26 (15.0%)	19.03 (16.8%)	0.28 (9.1%)	14.78 (17.2%)	42.22 (19.8%)	0.31 (10.6%)
STG-NCDE	9.19 (14.4%)	17.95 (11.8%)	0.29 (11.7%)	14.58 (16.0%)	39.40 (14.0%)	0.32 (11.7%)
STGODE	9.04 (12.9%)	17.46 (9.3%)	0.29 (9.3%)	13.37 (8.5%)	38.75 (12.6%)	0.30 (5.6%)
GMAN	8.98 (12.4%)	18.45 (14.2%)	0.29 (11.1%)	13.53 (9.5%)	38.31 (11.6%)	0.29 (6.2%)
GWNET	8.95 (12.2%)	17.75 (10.8%)	0.28 (7.9%)	14.04 (12.8%)	41.47 (18.3%)	0.30 (7.2%)
MTGNN	8.60 (8.6%)	17.33 (8.6%)	0.28 (6.6%)	13.85 (11.6%)	38.07 (11.0%)	0.30 (7.1%)
ASTGCN	8.54 (8.0%)	16.83 (5.9%)	0.27 (5.4%)	13.08 (6.4%)	35.76 (5.3%)	0.29 (5.2%)
STGCN	8.28 (5.1%)	16.53 (4.2%)	0.27 (3.1%)	13.11 (6.6%)	38.11 (11.1%)	0.29 (4.6%)
Multi-ATGCN	**7.87**	**15.83**	**0.26**	**12.24**	**33.87**	**0.28**

Note: Percentages in brackets are the increase in model performance brought by Multi-ATGCN using the model in that row as the baseline. The underlined cell is the best baseline.

Figure *5-6* shows the 24-step forecasting results of Multi-ATGCN in census tracts where the model shows the best and worst performance. As shown, population inflow in regions with the best performances presents more rhythmic patterns and thus is easier to predict. Outcomes of Multi-ATGCN well fit with observations, successfully capturing the daily and weekly patterns with reasonable magnitudes. On the other hand, population inflow in regions with poor prediction accuracy fluctuates more randomly. For those regions, although the Multi-ATGCN cannot well fit observations, it still shows the ability to generate a stable output following the average temporal patterns. Another noteworthy finding is that the well-performing census tracts have a much larger population inflow ($>10^2$) compared to the poorly-performed regions (<10), indicating a spatial fairness issue in model outcomes [200].



(a) Baltimore



(b) Washington, D.C.

Figure 5-6 Forecasting results of the top and last three census tracts (24-step)

Note: Top or last was measured by the MAPE of Multi-ATGCN. The first (third) row shows the forecasting results of the top (last) three census tracts across the testing set, with the gray areas zooming in and showing in the next row.

5.5 Model analysis

5.5.1 Performance across census tracts

Checking how the model performs across different zones allows us to compare and detect model weaknesses. **Figure *5-7*** shows how model performances vary across census tracts with different POI counts based on the 24-hour forecasting for Baltimore. As shown, MAPE broadly decreases with the increase of POI count, while MAE and RMSE follow an increasing pattern. It is plausible since MAPE is a relative metric while MAE and RMSE are absolute metrics. On the one hand, zones with fewer POIs are less likely to attract higher population flow, resulting in lower absolute metrics. On the other hand, zones with fewer POIs are more difficult to predict due to their higher randomness (**Figure *5-6***) and higher sensitivity to external interventions [12], leading to higher relative metrics. It is also worth mentioning that Multi-ATGCN presents a predominantly superior performance in data-sparse zones. Compared with the best baselines, i.e., STGCN and ASTCGN, the best baseline on 24-hour forecasting, Multi-ATGCN leads to an 8.7%, 19.1%, and 13.3% reduction in MAPE, RMSE, and MAE, respectively, in census tracts with the fewest POIs. Such an improvement demonstrates that efforts in involving zone-specific parameters, capturing external effects, and learning complex spatial structures can successfully improve the model's capability in handling more intractable tasks.

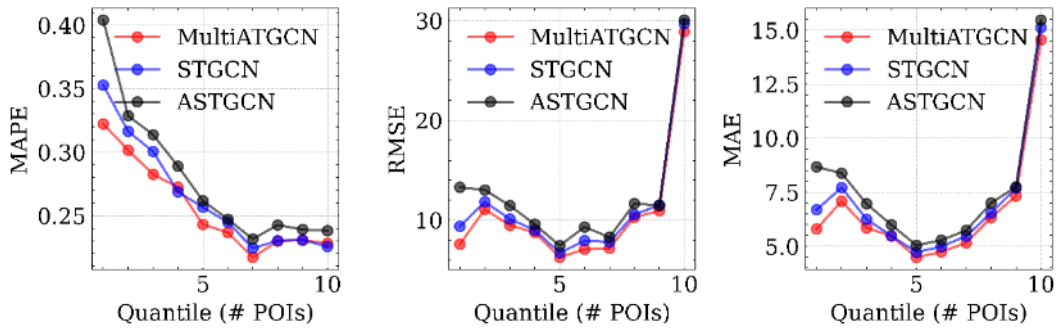


Figure 5-7 (Top 3) Model performance varying by POI counts

Note: POI counts are categorized by deciles.

5.5.2 Effects of the lower bound

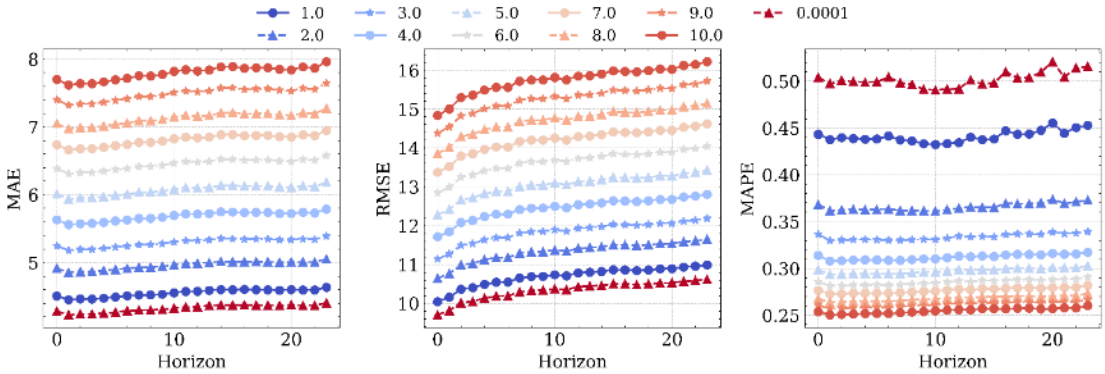
Three evaluation metrics are used to measure the performance of all models:

$$\text{MAPE} = \frac{1}{|\Omega|}\sum_{i=1}^{N}\sum_{t=t_0}^{t_0+d_D-1}\frac{|\hat{y}_{i,t}-y_{i,t}|}{|y_{i,t}|}I(y_{i,t}\geq\varepsilon) \quad (5\text{-}28)$$

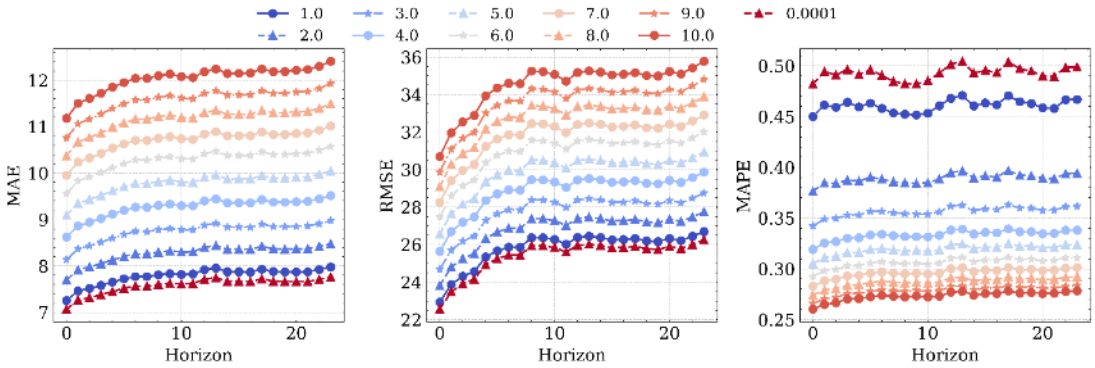
$$\text{MAE} = \frac{1}{|\Omega|}\sum_{i=1}^{N}\sum_{t=t_0}^{t_0+d_D-1}|\hat{y}_{i,t}-y_{i,t}|\,I(y_{i,t}\geq\varepsilon) \quad (5\text{-}29)$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}\sum_{t=t_0}^{t_0+d_D-1}(\hat{y}_{i,t}-y_{i,t})^2}{|\Omega|}I(y_{i,t}\geq\varepsilon)} \quad (5\text{-}30)$$

where $y_{i,t}$ is the population inflow of census tract i at time t and $\hat{y}_{i,t}$ is its prediction; N is the number of census tracts; d_D is the length of the prediction window; Ω is the set of observations that meets $y_{i,t}\geq\varepsilon$; ε is the lower bound; $I(.)$ is the indicator, which is 1 when $y_{i,t}\geq\varepsilon$ and 0 otherwise.



(a) Baltimore



(b) Washington, D.C.

Figure 5-8 Model performance varying by lower bounds

The lower bound ε is to exclude extremely small values in the testing set, which would significantly affect evaluation metrics. The model performance varying across different ε is shown in **Figure *5-8***. Overall, with the increase of the lower bound, the MAE and RMSE greatly increase while the MAPE significantly decreases. For example, changing the lower bound from 0.0001 to 10, the MAE for the Baltimore city dataset increases from 4.326 to 7.874, the RMSE increases from 10.327 to 15.835, while the MAPE decreases from 0.502 to 0.255. Another takeaway message here is that the decreasing rate of MAPE is gradually flattening. Increasing

the lower bound from 0.0001 to 2 would lead to a 0.136 drop in MAPE while increasing the lower bound from 2 to 4 only leads to another 0.054 drop.

5.5.3 Ablation study

An ablation study is conducted on the D.C. data to validate the effectiveness of key components that contribute to the model performance when making the 24-step prediction. The Multi-ATGCNs without different components are outlined as follows:

- w/o Auxiliary: Multi-ATGCN without auxiliary variables.
- w/o Closeness: Multi-ATGCN without the closeness temporal head.
- w/o Period: Multi-ATGCN without the period trend head.
- w/o Trend: Multi-ATGCN without the trend temporal head.
- w/o ZBN: Multi-ATGCN w/o zone-based normalization. Specifically, a global z-score normalization is applied to all crow inflow.
- w/o ZSP: Multi-ATGCN without zone-specific parameters. Specifically, the node embedding dimension d_{EB} is set as 1, and $\boldsymbol{E}_{\mathcal{G}}$ is set as a fixed tensor filled with 1.
- w/o GCN: Multi-ATGCN without the graph convolution. Specifically, the GCRNN is replaced by a two-layer pure GRU-based RNN.
- w/o FNNO: Multi-ATGCN without an FNN layer that uses all hidden states. Specifically, only the last hidden state from GCRNN is fed into the output layer.

Table 5-3 Ablation study for Multi-ATGCN

	MAE			**RMSE**			**MAPE**		
	Mean	**St.d.**	Δ%	**Mean**	**St.d.**	Δ%	**Mean**	**St.d.**	Δ%
w/o Auxiliary	8.113	0.125	3.04%	16.370	0.711	3.38%	0.265	0.004	2.09%
w/o Closeness	8.144	0.114	3.43%	16.747	0.249	5.76%	0.269	0.004	3.83%
w/o Trend	7.954	0.117	1.02%	16.384	0.197	3.47%	0.264	0.004	1.82%
w/o Period	8.144	0.100	3.44%	16.774	0.357	5.94%	0.267	0.003	2.81%
w/o ZSP	7.976	0.076	1.30%	16.085	0.255	1.59%	0.261	0.002	0.57%

w/o ZBN	8.036	0.110	2.06%	16.319	0.289	3.06%	0.265	0.003	1.92%
w/o GCN	8.151	0.086	3.53%	16.520	0.651	4.33%	0.266	0.002	2.41%
w/o FNNO	8.314	0.204	5.60%	16.743	0.703	5.74%	0.270	0.006	3.93%
Multi-ATGCN	**7.874**	**0.078**		**15.834**	**0.606**		**0.260**	**0.002**	

Note: Δ% is the change in model performance versus Multi-ATGCN.

The mean and st.d. of MAE, RMSE, and MAPE on the testing set over 10 repetitions for each version of Multi-ATGCN are reported in **Table *5-3***. The fully-connected output layer that uses all hidden states exhibits the greatest contribution to model performance, validating that including all hidden states of the GCRNN is helpful for long-horizon prediction. This also explains why RNN-based models such as AGCRN deteriorate significantly as the prediction horizon increases. The effect of GCN is evident as well, indicating the importance of enabling the information flow among interdependent zones. The effects of closeness and period temporal head (Eqs. (*5-3*) and (*5-4*)) are equivalently great, while the benefit brought by the trend temporal head (Eq. (*5-5*)) is low. The auxiliary information also significantly increases the model performance, which confirms the importance of including contextual information. Last, the two types of zone-specific processing, i.e., zone-based normalization and zone-specific parameters, both moderately enhance the model performance, indicating the necessity of involving local individual details.

5.5.4 Complexity analysis

To evaluate the computational cost, the number of parameters and training time of Multi-ATGCN are compared with other baselines running on the D.C. data for 24-hour forecasting. As shown in **Table 5-4**, Multi-ATGCN has the second most parameters as a sacrifice for extracting multiple temporal heads, integrating various auxiliary information, constructing multi-view graphs, and learning zone-specific

patterns. However, the training and evaluation speed of Multi-ATGCN is comparable to many state-of-the-art models, since it generates all predictions at once using the 2D CNN instead of iteratively using the S2S framework (e.g., DCRNN). Considering the salient performance improvement and the relatively fast computation speed, the overall computational cost of Multi-ATGCN is moderate.

Table 5-4 Comparison of computation cost for population inflow forecasting

Model	# Parameters	Training time/per epoch	Evaluation time
GRU	144045	8.93 s	0.88s
LSTM	186925	9.30 s	0.92 s
AGCRN	752730	35.53 s	4.44 s
DCRNN	372353	102.45 s	12.15 s
ASTGCN	988260	85.40 s	11.49 s
STGCN	732577	58.61 s	6.57 s
GMAN	380033	218.88 s	16.95 s
GWNET	350716	47.37 s	3.90 s
MTGNN	718840	31.31 s	2.95 s
STGODE	1613964	89.74 s	5.85 s
STG-NCDE	376284	263.58 s	20.24 s
Multi-ATGCN	**1603463**	**46.30 s**	**5.78 s**

5.5.5 Parameter study

A parameter study is conducted on five core hyperparameters of Multi-ATGCN, including the node embedding dimension, orders of Chebyshev polynomials, # RNN hidden units, # closeness temporal heads, and # period temporal heads. Experiments are run on Baltimore city data for 24-step prediction and results are depicted in **Figure *5-9***. As shown, Multi-ATGCN achieves the best performance when the node embedding dimension = 20, orders of Chebyshev polynomials = 2, # RNN hidden units = 64, # closeness temporal heads = 2, and # period temporal heads = 1. One finding is that with the increase of each hyperparameter, the model loss decreases at first and then slightly rebounds. All these hyperparameters would increase the model complexity as their value increases. Hence, an excessively small hyperparameter

would simplify the model and thus lead to underfitting. On the other hand, a large hyperparameter would significantly increase the parameter numbers, making the model harder to optimize and causing over-fitting. Overall, it would be a good practice to find the most appropriate hyperparameter configuration for each scenario to achieve the best model performance.

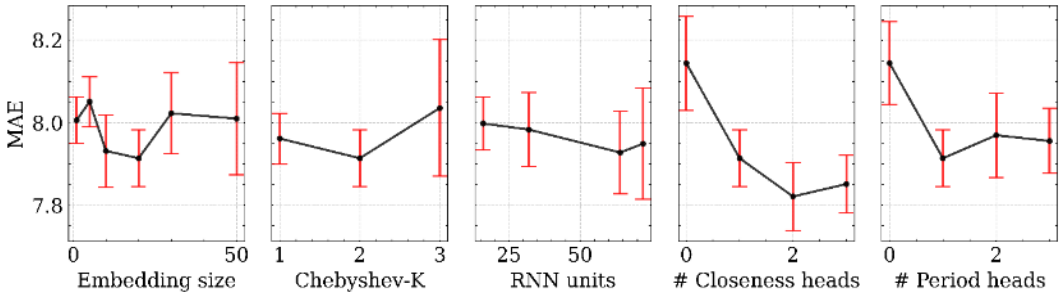


Figure 5-9 Influence of different core parameters on model performance

5.5.6 Graph learning

To validate the effectiveness of the proposed graph construction module, this study constructs graph adjacency matrices based on different algorithms and reports the model performance in **Table *5-5***. Experiments are run on Baltimore city data for 24-step prediction. Seven types of adjacency matrices are compared. The identity matrix is used as the baseline that assumes each zone is entirely self-dependent. Functional similarity, OD volume, and distance closeness are pre-defined matrices that need prior knowledge of zonal connectivity. In addition, two types of adaptive adjacency matrices are analyzed. The unidirectional method follows the definition in Eq. (*5-10*) ($\mathrm{SoftMax}\left(\mathrm{ReLU}\left(\vec{\boldsymbol{E}}_1\overleftarrow{\boldsymbol{E}}_2\right)\right)$), while the bidirectional method simplifies Eq. (*5-10*) by assuming a symmetric matrix ($\mathrm{SoftMax}\left(\mathrm{ReLU}\left(\vec{\boldsymbol{E}}_1\vec{\boldsymbol{E}}_1^{T}\right)\right)$). Lastly, the proposed multi-view method is the fusion of all adjacency matrices (Eq. (*5-19*)). As shown, the multi-

view approach achieves the lowest mean MAE, RMSE, and MAPE. The adjacency matrix measured by functional similarity is the second-best, followed by the OD-based measure. Although the two self-adaptive methods perform slightly worse compared to pre-defined methods, their performances are still remarkable even without any given prior knowledge. Finally, the distance matrix only performs slightly better than the identity matrix, which implies that directly using distance to measure the zonal connectivity may not explicitly capture the real graph structure.

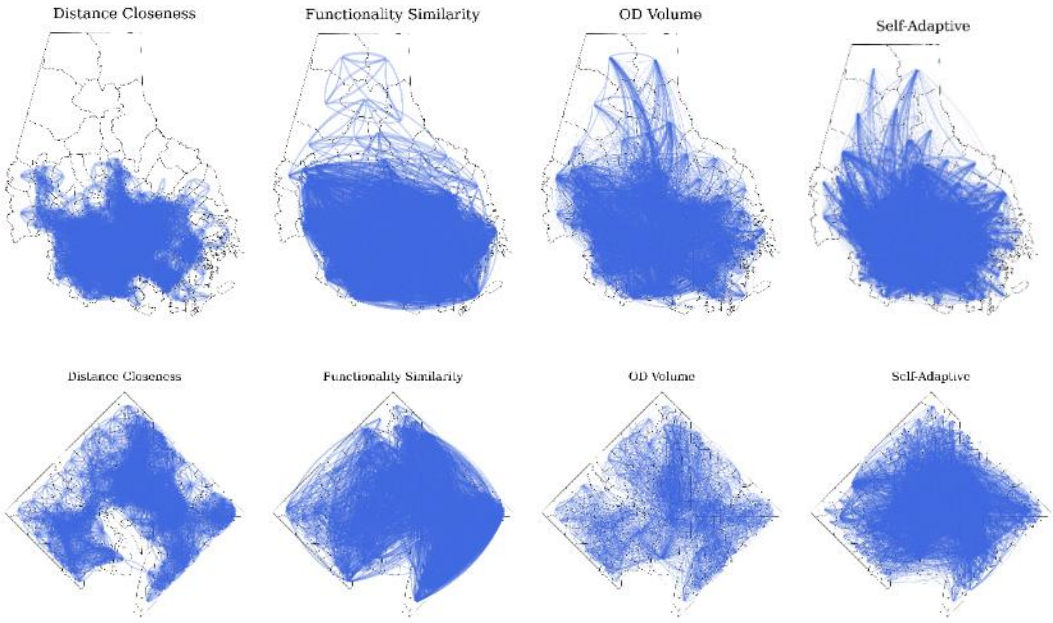
Table 5-5 Comparison of different adjacency matrices in Multi-ATGCN

	MAE			**RMSE**			**MAPE**		
Methods	**Mean**	**St.d.**	Δ%	**Mean**	**St.d.**	Δ%	**Mean**	**St.d.**	Δ%
Identity (I_N)	12.708	0.332	3.82%	35.479	0.970	4.75%	0.289	0.007	3.01%
Distance closeness ($\boldsymbol{A_D}$)	12.666	0.323	3.48%	35.258	0.971	4.10%	0.287	0.006	2.66%
Adaptive, bidirectional	12.617	0.330	3.08%	35.209	0.888	3.95%	0.289	0.007	3.10%
Adaptive, unidirectional ($\boldsymbol{A_{\tilde{A}}}$)	12.587	0.277	2.83%	35.147	0.854	3.77%	0.288	0.006	3.02%
OD volume ($\boldsymbol{A_{OD}}$)	12.508	0.272	2.19%	34.684	0.670	2.40%	0.287	0.008	2.47%
Functional similarity ($\boldsymbol{A_F}$)	12.490	0.246	2.04%	34.909	0.799	3.07%	0.284	0.006	1.29%
Multi-view ($\boldsymbol{\hat{A}}$)	12.243	0.205		33.871	0.609		0.281	0.005	

Note: Δ% is the change in model performance versus Multi-ATGCN using $\boldsymbol{I_N}$ as the adjacency matrix.

To further compare different adjacency matrices, the spatial distributions of four types of adjacency matrices are shown in **Figure 5-10**. Distance closeness is not plausible when geographic units are irregular since the distance between two zones is highly influenced by their area. Large zones, such as those located in suburbs, are more likely to be tagged as "isolated" by the distance closeness method since the distance from their centroids to others is inevitably long. However, the population inflow in these areas is not independent of other flows. Adjacency matrices measured by functional similarity and OD volume do not neglect these suburban areas, but the

two measures also present highly different spatial distributions. Lastly, the self-adaptive adjacency matrix learned by the model is different from the other three pre-defined matrices. It appears more like a fusion of all of the three pre-defined matrices, indicating that the real zonal connectivity is hard to interpret and cannot be simply described by one measure.



(a) Distance closeness (b) Functional similarity (c) OD volume (d) Self-adaptive learning

Figure 5-10 Spatial patterns of four types of adjacency matrices

5.6 Discussion

This section proposes a comprehensive GCN-based framework, the Multi-ATGCN, for citywide population inflow forecasting considering complex spatiotemporal dependency and heterogeneous external effects. By integrating a variety of deep learning techniques and spatiotemporal information, Multi-ATGCN demonstrates strong flexibility, comparable efficiency, and superior performance in multi-step time series forecasting. Specifically, multiple temporal components are extracted to

represent complex temporal dynamics. Multi-view self-adaptive adjacency matrices are constructed to comprehensively describe spatial structures. Parameter initialization and time sequence concatenation are further employed to learn from auxiliary variables. Last, all information is fused and passed through a zone-specific mix-hop GCRNN for jointly handling spatiotemporal dependency.

Experiments on two real-world datasets show that Multi-ATGCN achieves state-of-the-art performance, and the advantages are more evident in data-sparse zones and long-horizon prediction. Compared with the best baseline, Multi-ATGCN yields a 5.1-6.4% reduction in MAE for 24-step prediction. Such an improvement is even more salient in data-sparse zones, yielding a 13.3% reduction in MAE in census tracts with the fewest POIs. Although with high accuracy, the training speed of Multi-ATGCN is comparable to other models due to its non-recursion design in the decoder. The ablation study further demonstrates the importance of different components in improving the model performance, among which the fully-connected decoder and the GCN are the two components that bring the largest improvement. The comparison among different types of adjacency matrices indicates that the adjacency matrix would significantly affect the model performance. The multi-view approach achieves the best performance, followed by functional similarity and OD-based measures.

Several limitations are recognized and deserve further research. First, in addition to forecasting population inflow, it is equally important to quantify how population inflow contributes to traffic conditions on the road network. Specifically, the outcomes of Multi-ATGCN can be fed into dynamic traffic assignments to generate citywide, road-level, time-dependent traffic volume and speed in the future.

Second, only the population inflow is forecasting, which is relatively simple to meet different requirements in travel demand modeling. Further studies should consider expanding the Multi-ATGCN for supporting different scenarios such as multi-task learning (i.e., multi-modal multi-activity travel demand forecasting), OD matrices forecasting, and walk-forward validation (i.e., predicting human mobility in an online manner, particularly useful under unexpected interventions). Last but not least, the findings in this study are data-specific. A verification of more types of datasets is warranted to further test the model's generalizability.

6 Chapter 6: Individual trip itinerary forecasting

The previous chapter introduces a novel graph-based deep learning framework for population inflow time series forecasting. However, population inflow belongs to aggregated travel demand measure. One main advantage of MDLD is that it contains rich individual trip information, providing the opportunity to model travel demand in a bottom-up way [23, 35], which is highly compatible with the current disaggregated activity-based model paradigm [24, 25]. Hence, developing a powerful knowledge fusion and discovery framework to forecast future travel demand at an individual level based on abundant historical individual trip information is important.

Compared with aggregated-level population flow forecasting, individual-level trip itinerary forecasting is more challenging because of highly-random spatiotemporal patterns of individual travels (**Figure 6-2**), multi-structure forecasting tasks (i.e., jointly forecasting time, activity, and location), large-scale training datasets (i.e., each observed device is an entity), imbalanced distributions of places and activities (i.e., a small portion of places attracts a large portion of visits, as indicated by blue spots in **Figure 6-1**), cold start issue (i.e., discovering the potential links between new (or seldomly-visited) locations and individuals), and heterogeneous external effects (i.e., effects of individual features, weather and holidays). Due to these challenges, individual-level trip itinerary forecasting is more difficult, reporting much lower accuracy in prior studies compared with aggregated-level forecasting [33, 165].

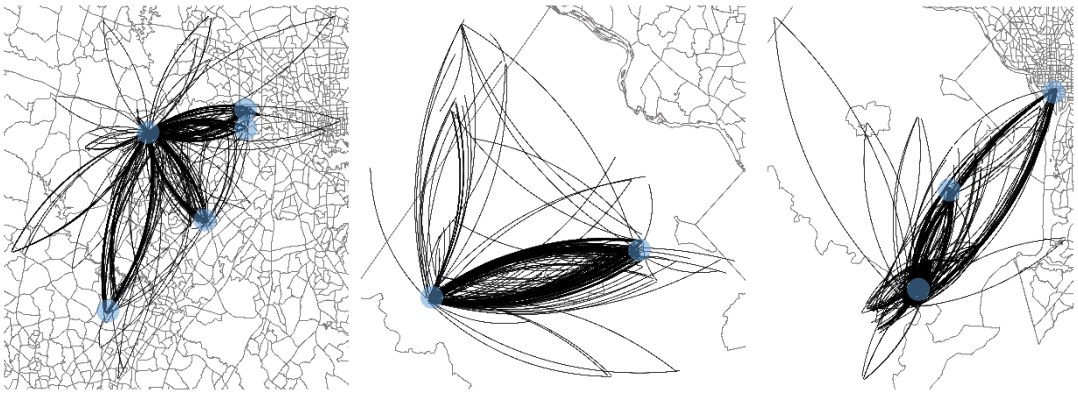


Figure 6-1 Spatial distribution of individual trip origins and destinations

Note: Data show one-month trips generated by three randomly-selected devices. Blue spots mark the locations with the most frequent visits by each device.

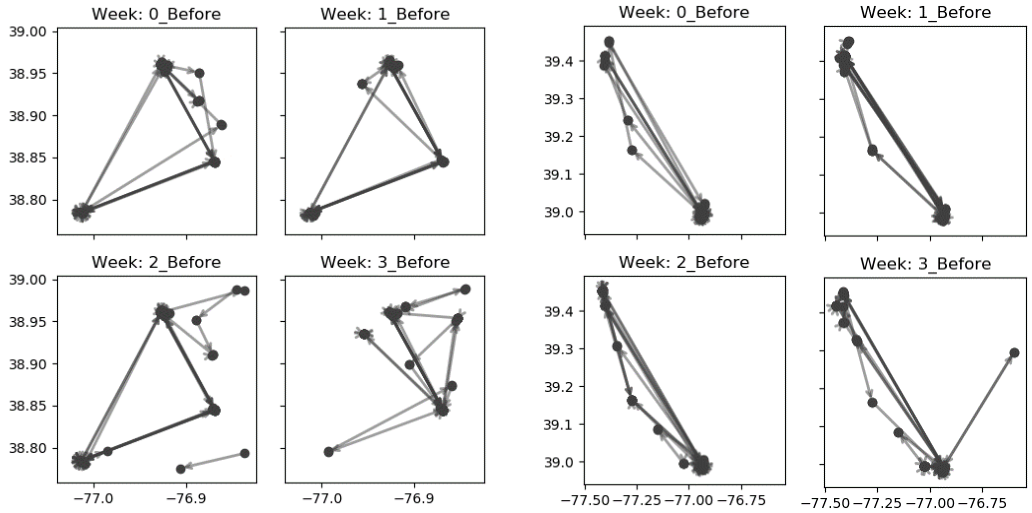


Figure 6-2 Weekly evolution of the spatial distribution of individual trips

Note: Data shows one-month origins and destinations of trips generated by two randomly-selected devices. Each panel shows the trips generated in a specific week ranging from week 0 to week 3.

To this end, this chapter proposes a hierarchical activity-based framework (HABF) for simultaneously predicting the activity, departure time, and location of the next trip for each observed mobile device (**Figure 6-3**). First, a pipeline of data preprocessing is conducted, including activity labeling, missing trip imputation, short

trip linking, and user filtering. The main goal of this step is to construct a reliable training dataset from the raw trip roster for model learning. Then, an Interpretable Hierarchical Transformer (IHTF) is proposed for predicting the daily activity chain in each hour for each observed device, incorporating features from travelers, trips, and external environments. IHTF considers heterogeneous external effects and can handle large-scale datasets benefiting from its transformer-based network design. Meanwhile, loss functions in the semantic segmentation domain [43] are introduced to address the imbalanced classification issue. Last, a location generator is designed to generate locations based on predicted activity chains and historically visited places. The location generator addresses the cold start issue by combining the local and global probability. The whole framework was trained and tested on a county-level dataset covering 2-month trips from over 18,000 devices with their home locations located in Montgomery County, MD, and showed acceptable prediction accuracy.

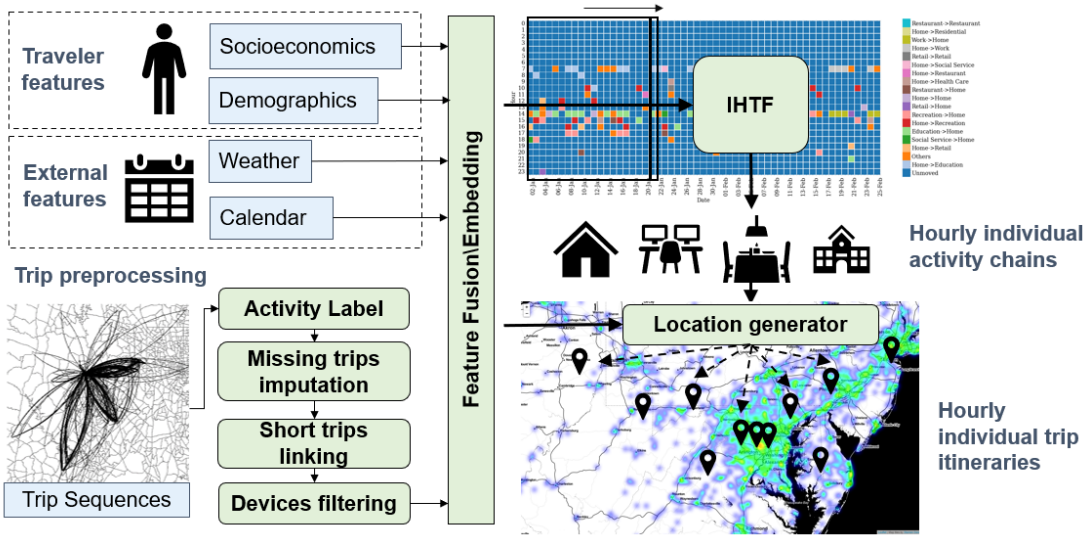


Figure 6-3 Hierarchical activity-based framework (HABF)

6.1 Trip preprocessing

6.1.1 Activity labeling

Linking land use of a diverse region that individual visits with the explicit activity that the individual engages in are challenging [77]. Hence, most previous studies only considered three activity types, i.e., home, work, and others. This section aims to cover 11 types of activities, including home, work, retail, restaurant, education, recreation, health care, social service, residential, personal services, and others. The activity types are categorized based on the NAICS code of POIs from SafeGraph, with the correspondence shown in **Table *6-1***. Note that home and work are fully extracted from the MDLD other than determined by the NAICS code of POIs; hence, their relations with NAICS code are not reported in **Table *6-1***. The description of the NAICS code can be found at https://www.naics.com/.

Table 6-1 Correspondence table between activity type and NAICS code

Activity	NAICS Code of POIs	POI Count
Person Service	'52', '54', '5321', '5322', '5323', '5324', '5331', '8111', '8112', '8113', '8114', '8121', '8122', '8123', '8129'	14465
Social Service	'56', '92', '8131', '8132', '8133', '8134', '8139', '8141',	11622
Retail	'42', '44', '45'	57573
Residential	'5311', '5312', '5313'	10151
Education	'61', '6244'	10382
Health Care	'6211', '6212', '6213', '6221', '6222', '6223', '6231', '6214', '6215', '6216', '6219', '6232', '6233', '6239', '6241', '6242', '6243'	16125
Recreation	'71'	20706
Restaurant	'7211', '7212', '7213', '7223', '7224', '7225'	61727
Others	'00', '11', '21', '22', '23', '31', '32', '33', '48', '49', '51', '55'	9311
Home	-	-
Work	-	-
Sum		212062

This section proposes a probabilistic method for determining the activity type based on 1) the closeness of trip origin/destination to the device's home/work location

and/or other POIs, and 2) the attraction of the POIs measured by the average historical hourly visit. The underlying assumption is that people are more likely to visit the POIs which are closer to their destinations and show higher popularity. Specifically, if the origin/destination is within 500 m of their home/work location, the activity type will be directly assigned as home/work. For those origins/destinations far from their home and work locations, the nearest 10 POIs within 500-meter buffers are selected and attached a probability based on their distance to the origin/destination and their attraction measured by the number of visits. Then, the probability is summed up by POI types, and the POI type with the maximum probability is chosen as the final activity type. Assume in a 500 m buffer, there are total I POIs belonging to K types; for each type, the number of POIs is N_k. The probability of the visit belonging to the kth type, i.e., $P_{k,t}$, is computed as Eq. *(6-1)*, and the final activity type, i.e., $\dot{A}$, is computed as Eq. *(6-2)*:

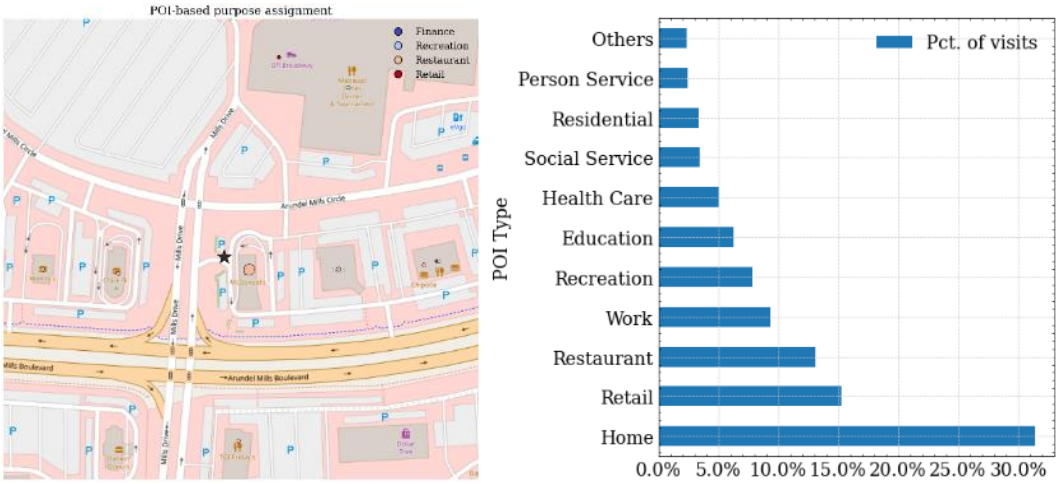
$$P_{k,t} = \sum_{n=1}^{N_k} \frac{v_{n,t}/d_i^2}{\sum_{i=1}^{I} v_{i,t}/d_i^2} \qquad (6\text{-}1)$$

$$\dot{A} = argmax_k(P_{k,t}), k = 1,2,\ldots,K \qquad (6\text{-}2)$$

where $v_{n,t}$ is the average number of historical visits to POI *n* at hour *t*; d_i is the distance from the origin/destination to POI *n*.

Figure 6-4 (a) illustrates the process of finding the activity type for one trip. The black star shows the trip destination while the spots show different POIs with their color varying by POI types and their size varying by the probability (Eq. *(6-1)*). The activity is finally defined as restaurant, which is consistent with the McDonald's shown on the map. Note that due to the lack of ground truth, the accuracy of this

probabilistic activity labeling algorithm cannot be tested. However, it is currently the best way that can be designed for assigning activities in this study. **Figure 6-4** (b) shows the distribution of activities, where home, retail, restaurant, work, and recreation are the top five activity types, accounting for 77.10% of total trips.



(a) (b)

Figure 6-4 Illustration of activity labeling (a) and distribution of activities (b)

6.1.2 Missing trips imputation

Although MDLD-driven travel demand can achieve a high consistency with surveys when the spatial aggregation unit is large [3, 71], the consistency will be greatly attenuated under finer zoning systems. The accuracy will be further attenuated when zooming into individuals. One of the main reasons is that MDLD cannot ensure the recording of all the movements of each device. To address this, this section proposes several rules for filling the missing trips. First, unless the device is on a long-distance tour away from home (greater than 50 miles), it should start and end at home within a day (04:00 AM – 04:00 AM +1) [25, 71]. Second, for each device, trip sequences should be spatiotemporally continuous. Alternatively, the destination of the previous

trip should be spatiotemporally close to the origin of the next trip. For each device, if these requirements are violated, a new trip will be inserted following the rules:

- A daily tour should start from home; if not, generate a "$Home \rightarrow O_{next}$" trip for that device, where O_{next} is the origin of the first observed trip on that day.
- A daily tour should end at home; if not, generate a "$D_{pre} \rightarrow Home$" trip for that device, where D_{pre} is the destination of the last observed trip on that day.
- Two successive trips should be continuous; if not (i.e., if $Dist(D_{pre}, O_{next}) > 1mile$), generate a "$D_{pre} \rightarrow O_{next}$" trip for that device, where D_{pre} is the destination of the previous trip and O_{next} is the origin of the next trip.

After generating origins and destinations, the departure time of each synthetic trip is then assigned based on its historical probability grouped by activity types and days of the week. If the device has engaged in the activity before, the departure time is assigned based on the combination of individual and global probability. Otherwise, the departure time is generated entirely based on the global probability:

$$P_{i,k,w,t} = \begin{cases} \delta \frac{n_{i,k,w,t}}{\sum_{t=E_i}^{S_i} n_{i,k,w,t}} + (1-\delta) \frac{\ddot{n}_{k,w,t}}{\sum_{t=E_i}^{S_i} \ddot{n}_{k,w,t}}, if \sum_{t=0}^{23} n_{i,k,w,t} > 0 \\ \frac{\ddot{n}_{k,w,t}}{\sum_{t=E_i}^{S_i} \ddot{n}_{k,w,t}}, otherwise \end{cases} \quad (6\text{-}3)$$

where $n_{i,k,w,t}$ is the total number of trips generated by device *i* on the day of week *w* at hour *t* engaging in activity *k;* $\ddot{n}_{k,w,t}$ is the total number of trips generated by all devices on the day of week *w* at hour *t* engaging in activity *k;* E_i is the end time of the trip before the synthetic trip and S_i is the start time of the trip after the synthetic trip*;* δ is a parameter controlling the weight of individual probability (set as 0.9 in this study).

Example: Assume that device A0, misses a trip starting from home that needs to be synthetic. The missing trip occurs on a day with day of week = 0 (i.e., Monday). Its previous trip ends at 03:00 am and its next trip starts at 10:00 am. Hence, the synthetic trip may depart at any hour between 03:00 am and 10:00 am. For each hour, the probability of departing is calculated based on the historical number of trips departing at that time. For example, the local probability of the trip departing between 03:00 – 04:00 am is $n_3 / \sum_{t=3}^{9} n_t$, the global probability of the trip departing between 03:00 – 04:00 am is $\ddot{n}_3 / \sum_{t=3}^{9} \ddot{n}_t$, and the final probability is their weighted sum: $\delta n_3 / \sum_{t=3}^{9} n_t + (1-\delta)\ddot{n}_3 / \sum_{t=3}^{9} \ddot{n}_t$.

Table 6-2 Example of a frequency lookup table

Day of week	Hour	Activity	# trips (Device A0)	# trips (All devices)
0	0	Home	n_0	$\ddot{n}_0$
0	1	Home	n_1	$\ddot{n}_1$
0	2	Home	n_2	$\ddot{n}_2$
0	3	Home	n_3	$\ddot{n}_3$
…	…	…	…	…
0	23	Home	n_{23}	$\ddot{n}_{23}$

6.1.3 Short trips linking and clustering

To reduce the computation burden, this section aggregates the trips on an hourly basis. Alternatively, this section assumes at most one trip can be generated by a device during an hour. Therefore, trips with departure times within an hour are linked to a new trip. Meanwhile, two continuous trips with overly close distances (below 0.5 miles) and time intervals (below 4 minutes) are linked to decrease the randomness brought by short trips. Last, extremely short trips, i.e., trips with distances under 0.1 miles and travel times shorter than 1 minute are removed.

Another widely existing issue is the oscillation of GPS points collected by mobile devices, which also leads to the identification of some unreal short trips. The

reason is that positioning noises result in multiple candidates that are in the same place being estimated at slightly different coordinates [13, 17]. To account for this, for each user, the Density-based spatial clustering of applications with noise (DBSCAN) [215] is further employed to cluster the trip origins and destinations that are too close. The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. There are two parameters, min_samples, and epsilon, to define the clusters. Higher min_samples or lower epsilon suggest a higher density necessary to form a cluster. In this study, the epsilon parameter, which is the max distance that points can be from each other to be considered a cluster, is set as 0.5 miles. The min_samples parameter, which is the minimum cluster size, is set as 1. **Figure 6-6** illustrates the trip spatial distribution of two randomly-selected devices before and after applying the DBSCAN. As shown, the DBSCAN successfully merges those spatially close trip origins and destinations, simplifying the mobility graph structure while retaining the initial spatial resolution and locations.

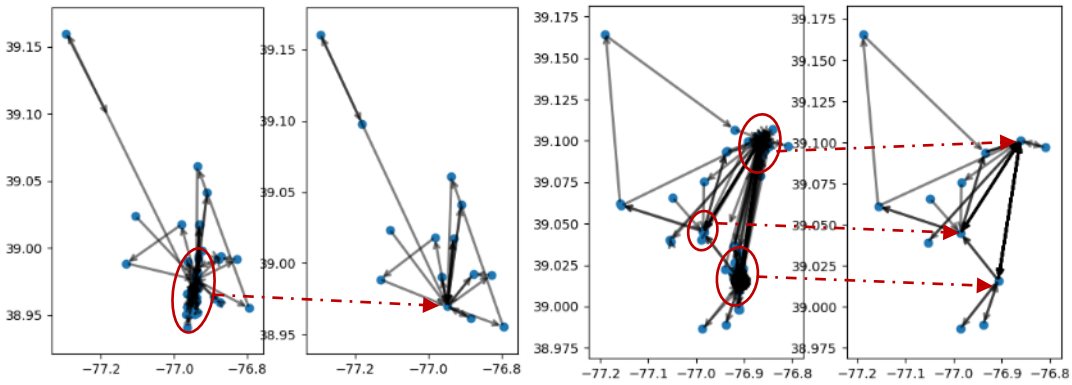


Figure 6-5 Illustration of short trip merging using DBSCAN

Note: The first (third) panel represents the two-month trip spatial distribution before DBSCAN while the second (fourth) panel represents the corresponding distribution after DBSCAN. The red markers and arrows denote the main changes clustered by DBCAN.

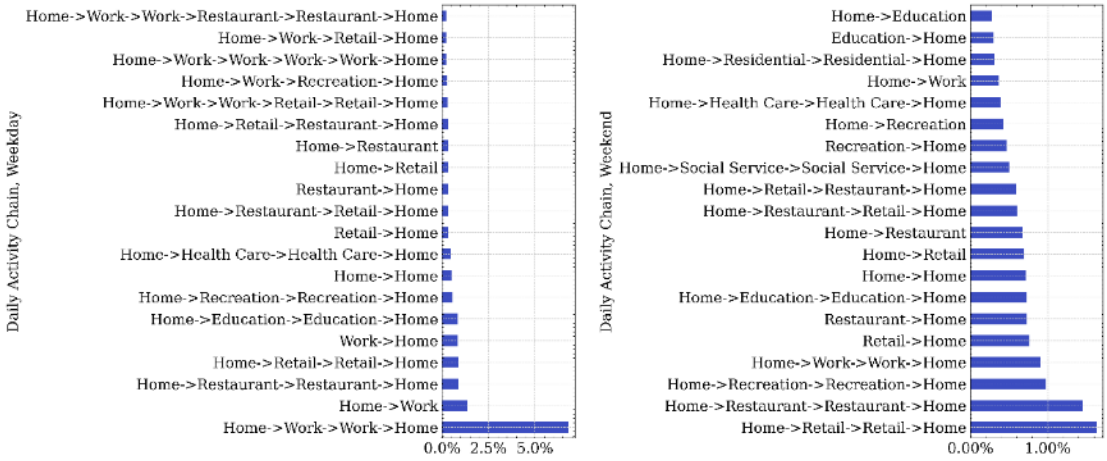
6.1.4 Devices filtering

The previous steps can only handle devices with a small fraction of missing observations, while for those with lots of missing records or with highly irregular mobility patterns, the aforementioned steps remain unable to provide solutions. Hence, in the last step, devices with very few or too many records are removed to avoid polluting the model learning process. Specifically, only devices generating at least one trip per week and at most 70 trips per week (10 trips/day) are selected, which leads to 18,927 devices being selected from the initial 145,189 devices.

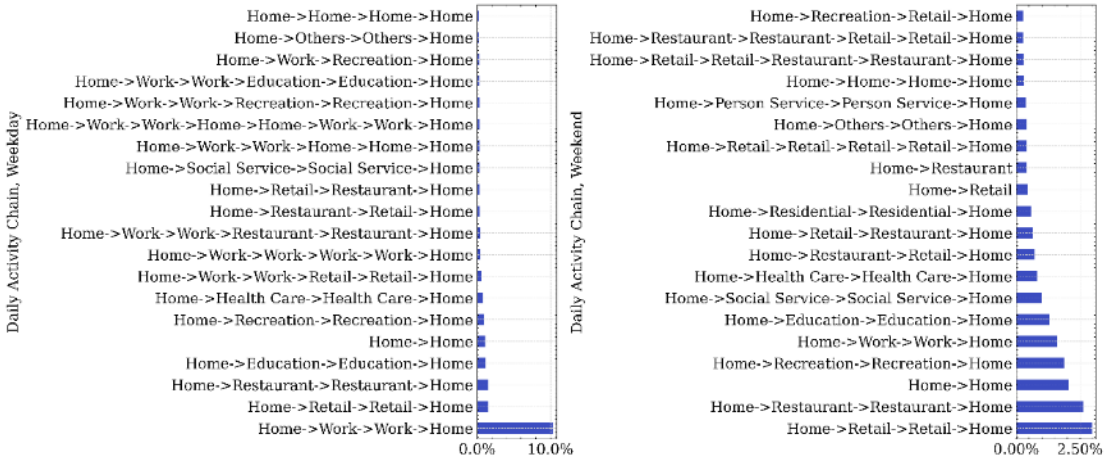
6.1.5 Before/After comparison

Figure 6-6 shows the distribution of daily activity chains before and after the data preprocessing, and **Table *6-3*** reports the top 20 daily activity chains which show the greatest changes after trip processing. Weekdays and weekends are reported separately. A distinguishable improvement can be observed after data preprocessing. Most processed daily tours now start and end at home, and % Home->Work->Work->Home increases from 6.81% to 10.31% during weekdays. Meanwhile, the number of short and incomplete tours is greatly decreased. For example, % Home->Work decreases from 1.36% to 0.14%, and % Work->Home decreases from 0.85% to 0.07% during weekdays. Moreover, most daily tours now show a continuous activity chain, i.e., the activity of the previous destination is the same as the activity of the next origin. Another interesting but intuitive finding is the difference between weekdays and weekends. Most of the daily activity chains during weekdays belong to "Home → Work → Work → Home", while during weekends the most frequent daily

activity chain is “Home → Retail → Retail → Home”. Also, different types of daily chains are distributed more evenly during the weekends compared to weekdays.



(a)



(b)

Figure 6-6 Distribution of daily activities before (a)/after (b) trip preprocessing

Table 6-3 Daily activities (%) comparison before/after trip preprocessing

	Post-processing		Pre-processing		$\Delta = Post - Pre$	
Daily activity chain	Week day	Weekend	Week day	Weekend	Week day	Weekend
Home->Work->Work->Home	10.31	1.61	6.81	0.91	3.50	0.71
Home->Work	0.14	0.06	1.36	0.36	-1.21	-0.31
Work->Home	0.07	0.03	0.85	0.25	-0.78	-0.21
Home->Retail->Retail->Home	1.53	2.96	0.90	1.63	0.63	1.33

Home->Home	1.11	2.04	0.51	0.72	0.60	1.33
Home->Restaurant->Restaurant->Home	1.47	2.62	0.91	1.44	0.56	1.17
Home->Recreation->Recreation->Home	0.97	1.88	0.55	0.98	0.42	0.90
Home->Health Care->Health Care->Home	0.82	0.83	0.46	0.39	0.36	0.44
Home->Education->Education->Home	1.19	1.29	0.84	0.73	0.34	0.56
Retail->Home	0.07	0.15	0.34	0.76	-0.27	-0.61
Restaurant->Home	0.07	0.15	0.32	0.73	-0.25	-0.58
Home->Work->Work->Retail->Retail->Home	0.55	0.11	0.30	0.05	0.24	0.06
Home->Work->Work->Work->Work->Home	0.48	0.08	0.25	0.04	0.23	0.04
Home->Work->Work->Restaurant->Restaurant->Home	0.45	0.10	0.24	0.04	0.21	0.06
Recreation->Home	0.06	0.11	0.23	0.47	-0.17	-0.37
Home->Work->Work->Recreation->Recreation->Home	0.29	0.05	0.12	0.01	0.17	0.04
Home->Work->Work->Home->Home->Home	0.34	0.05	0.18	0.03	0.16	0.02
Home->Social Service->Social Service->Home	0.34	0.99	0.19	0.50	0.15	0.49
Education->Home	0.05	0.07	0.20	0.30	-0.15	-0.23
Home->Retail->Retail->Work->Work->Home	0.22	0.06	0.08	0.02	0.14	0.04

Note: Data are ranked by the difference between the post and pre-processed data on weekdays.

The comparison of trip statistics before and after trip preprocessing is reported in **Table *6-4***. Overall, the daily number of trips per person has largely increased and tends to be more consistent with the daily trip rate reported by NHTS (3.37). In addition, the average travel time and travel distance both increase after trip preprocessing, which is plausible due to the short trip linking and merging.

Table 6-4 Trip statistics before/after trip preprocessing

	Before	After
# devices	145,189	18,927
Average # trips per device per day	1.16	3.52
Average travel time per trip (minute)	26.68	28.91
Median travel time per trip (minute)	16.63	15.95
Average travel distance per trip (mile)	11.64	13.31
Median travel distance per trip (mile)	4.15	4.21

6.2 Hourly activity chain prediction

6.2.1 Objectives and challenges

The main objective of this section is to predict the hourly activity chain of each device in the next 24 hours based on its historical information from the previous week. The reason that this study first predicts activities instead of places is threefold. First, activity patterns are more regular than places. For example, a person may go to restaurants every noon while the visited restaurant may vary. Second, directly predicting the places is more difficult since there are thousands of places that a person can visit, while the number of activity types is constrained to eleven in this study. Last, directly feeding the places into the model also leads to cold start issues since the model only gets the knowledge about previously visited places. Separately predicting place and activity while leaving the place prediction to a probabilistic module given the forecasted activities could well solve the issues.

Forecasting the hourly activity chain for each device, however, is still challenging due to the following issues. The main challenge is that at an individual level, high randomness still exists in temporal patterns of activity chains. **Figure 6-7** shows the heatmap of hourly activities of a randomly-selected device. Some regular patterns can be observed in **Figure 6-7**. For example, this device frequently makes "Home → Work" trips during 07:00 – 08:00 am and completes "Work → Home" trips during 02:00 – 03:00 pm. However, there are also some randomly distributed activities scattered over other hours, which increases the difficulties of forecasting. Similar patterns can be found in **Figure 6-8,** which depicts the daily total trip counts across the two months separated by activity types. The main finding is that some

activities are characterized by a higher regularity that allows the model to easily detect predictable trends, for example, the "Home ↔ Work", while the others exhibit a higher degree of spontaneity, indicating a higher difficulty of prediction.

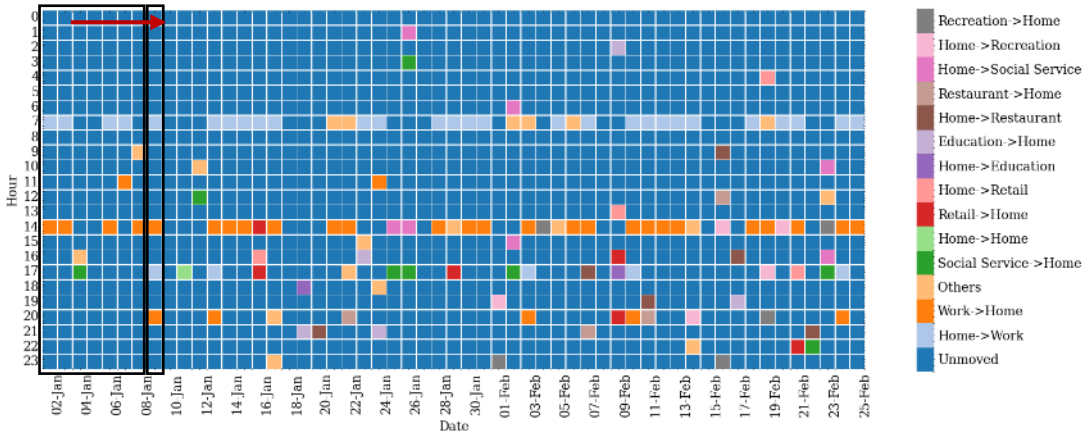


Figure 6-7 Tile plot of a device's hourly activities in two months

Note: The y-axis is the hour of the day and the x-axis is the date in the first two months of 2020. Each tile represents the activities the device engaged in during that hour, labeled by a chain connecting activity types of the origin and destination.

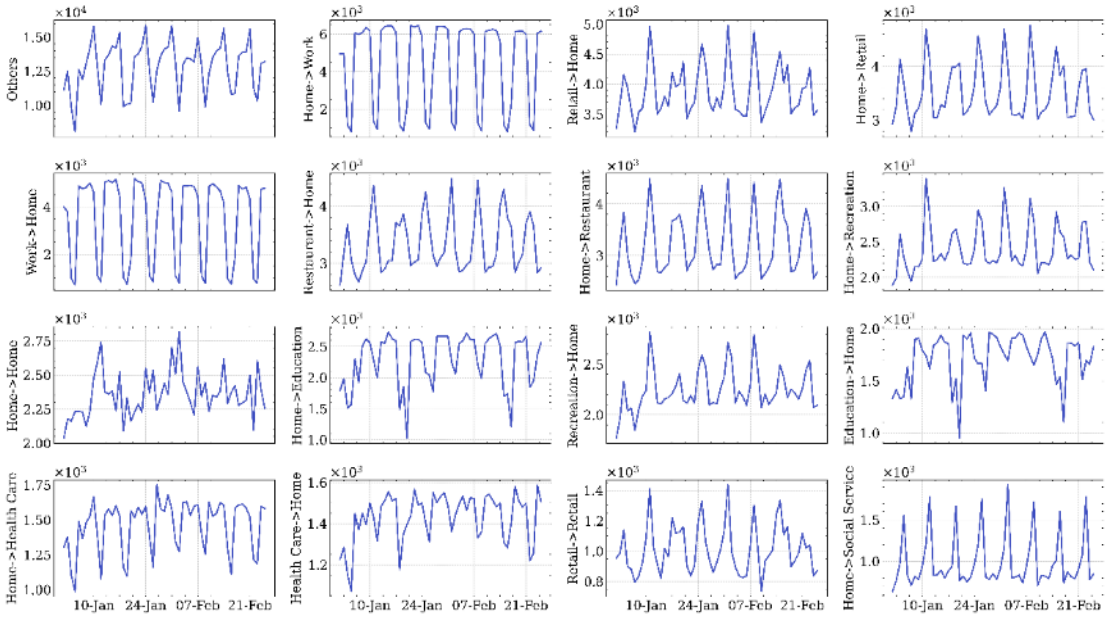


Figure 6-8 Daily evolution of trip counts by activity types

Another main challenge is the imbalanced distribution of activity chains. At an hourly level, there are totally 122 types of activity chains. To simplify the classification problem, this study only considers the top 30 types and packs the others as a new type "Others". Among the 31 types of activities, "Stationary" (the blue tile in **Figure 6-7**) accounts for 86.64%, "Others" accounts for 2.85%, "Home → Work" accounts for 1.00%, "Retail → Home" accounts for 0.87%, "Home → Retail" accounts for 0.79%, and "Work → Home" accounts for 0.78% (**Figure 6-9**). As a result, classical classifiers tend to ignore minority classes while concentrating on classifying the majority ones accurately (i.e., "Stationary" in this study). This deviates from the research goal since the minority, i.e., those time slots with trips being generated, are the target that this study needs to accurately predict.

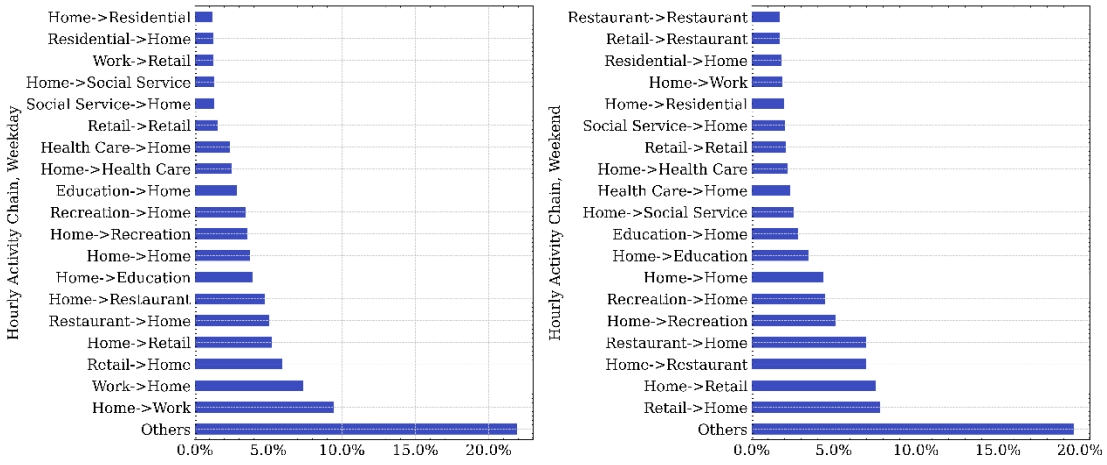


Figure 6-9 Distribution of hourly activity chains: weekday vs. weekend

The last challenge is that human travel behavior is a function of various factors. For example, workers and nonworkers may have different travel patterns; holidays may induce irregular travel behaviors; weekday and weekend patterns may also be diverse (**Figure 6-10**). Hence, the model should have the capability to learn

from multi-structural features, including static/time-varying variables and external/internal variables.

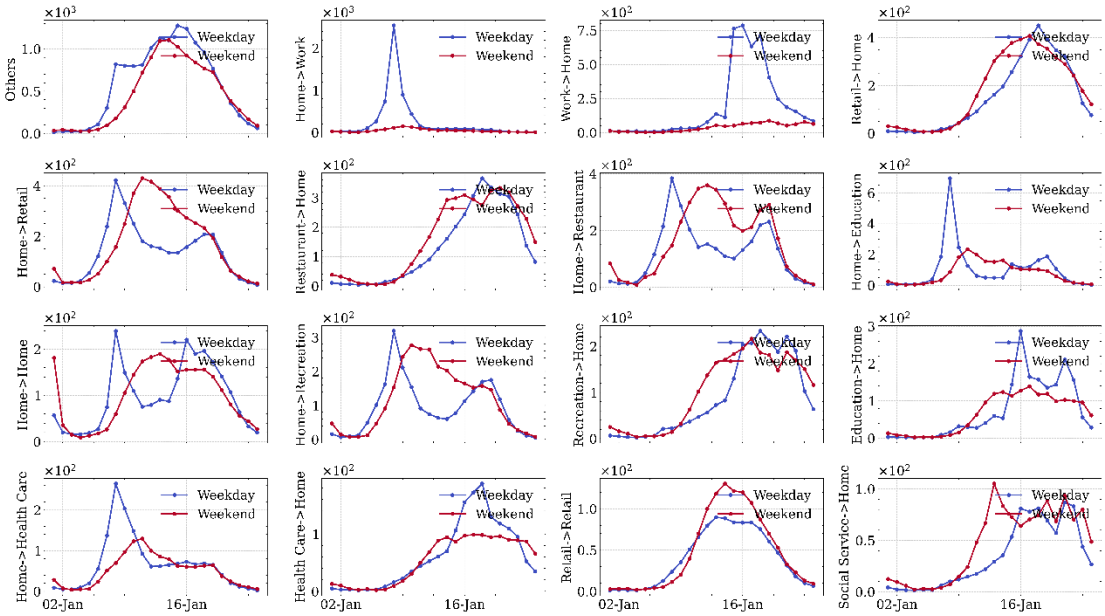


Figure 6-10 Hourly trip counts by activity types: weekday vs. weekend

6.2.2 Problem statement

Assume there are *I* devices in the training set, each device *i* is associated with a set of external variables $\boldsymbol{X}_i = [\boldsymbol{x}_{i,1}, \boldsymbol{x}_{i,2}, \ldots \boldsymbol{x}_{i,d_T}]^T \in \mathbb{R}^{d_T \times d_V}$ and a time sequence of activity chains $\boldsymbol{Y}_i = [y_{i,1},\ y_{i,2},\ \ldots\ y_{i,d_T}]^T \in \mathbb{R}^{d_T}$, where $\boldsymbol{x}_{i,t} = [x_{i,t}^{(1)},\ x_{i,t}^{(2)},\ \ldots,\ x_{i,t}^{(d_V)}] \in \mathbb{R}^{1 \times d_V}$ is the set of external variables at hour *t*; $x_{i,t}^{(k)}$ is the k^{th} external variable; $y_{i,t}$ is the activity pair on hour *t* generated by device *i*, which is defined as "$AO_{i,t} \rightarrow AD_{i,t}$", where $AO_{i,t}$ is the activity type of the trip origin and $AD_{i,t}$ is the activity type of the trip destination; d_V is the number of external variables, and d_T is the length of the activity chain time series. The general goal of this study is to train a function to predict future $\boldsymbol{Y}_i$ given its past information.

The origin dataset, $[\boldsymbol{X}_i, \boldsymbol{Y}_i]$, comprises a long time sequence for each device, which cannot be directly learned by the model. Hence, a rolling window is applied to convert the raw dataset into a set of learnable samples (**Figure 6-11**). Under a rolling setting with encoder length d_E (i.e. the length of look-back period, set as 24*7 h) and decoder length d_D (i.e. the length of the prediction period, set as 24 h), the goal is to jointly learn a non-linear mapping $\mathcal{F}(.)$ between each pair of encoder and decoder:

$$\widehat{\boldsymbol{y}}_{i,(t+1):(t+d_D)} = \mathcal{F}\left(\boldsymbol{x}_{i,(t-d_E+1):t}, \boldsymbol{y}_{i,(t-d_E+1):t}\right) \quad (6\text{-}4)$$

for all $t \in [d_E, d_E + 1, \dots, d_T - d_D]$ and $i \in [1,2, \dots, I]$; where *t* is the starting time of a rolling window; *I* is the number of devices; $\widehat{\boldsymbol{y}}_{i,(t+1):(t+d_D)} \in \mathbb{R}^{d_D}$ is the d_D-step prediction, $\boldsymbol{x}_{i,(k-d_E+1):k} \in \mathbb{R}^{d_E \times d_V}$ are external variables in past d_E hours; $\boldsymbol{y}_{i,(k-d_E+1):k} \in \mathbb{R}^{d_E}$ are observed activities in past d_E hours.

In the context of activity chain forecasting, external variables $\boldsymbol{x}_{i,t}$ can be further divided into three types: 1) static variables $\boldsymbol{s}_i \in \mathbb{R}^{d_s}$, such as demographics, socioeconomics, work status, and home location of the device; 2) time-varying future-unknown variables $\boldsymbol{u}_{i,t} \in \mathbb{R}^{d_u}$, such as weather conditions; and 3) time-varying predetermined variables $\boldsymbol{v}_{i,t} \in \mathbb{R}^{d_v}$, such as holidays, days of the week, and the time index. The three types of variables have different dimensions and should be handled accordingly. Specifically, static variables do not have temporal dimensions, time-varying future-unknown variables cannot cover future information, while time-varying predetermined variables can include both historical and future information. Under such settings, Eq. *(6-4)* is rewritten as:

$$\widehat{\boldsymbol{y}}_{i,(t+1):(t+d_D)} = \mathcal{F}\left(\boldsymbol{s}_i, \boldsymbol{u}_{i,(t-d_E+1):t}, \boldsymbol{v}_{i,(t-d_E+1):(t+d_D)}, y_{i,(t-d_E+1):t}\right) \quad (6\text{-}5)$$

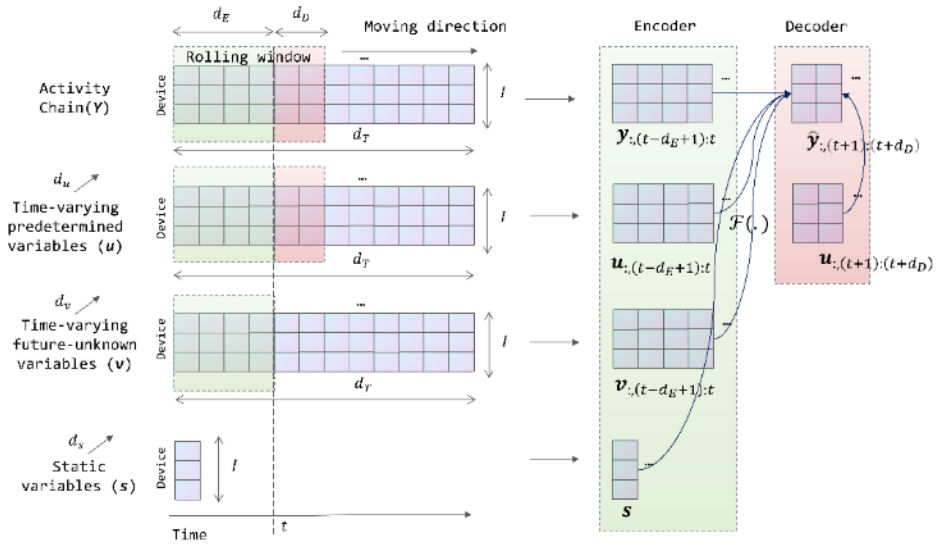


Figure 6-11 Illustration of processing activity chain time series for IHTF

Figure 6-11 illustrates the data structure of the forecasting problem. For each matrix, the y-axis is the device, and the x-axis is the time. Different matrices denote different types of variables. The rolling window comprises an encoder (the green box) and a decoder (the red box). It slides across all devices with a step of 1 hour from the starting time to the end. Samples are then grouped into batches for model training. **Figure 6-11** also demonstrates how different types of variables should be handled. The rolling window is not applied to static variables since they do not have a time dimension. Meanwhile, the decoder in the rolling window is not applied to time-varying future-unknown variables to avoid information leakage.

6.2.3 Proposed approach: Interpretable hierarchical transformer (IHTF)

This section introduces an Interpretable Hierarchical Transformer (IHTF) to predict individual hourly activity chains. The reason to propose the IHTF is three-fold. First, IHTF has a well-designed structure to handle different types of exogenous variables, which is of great importance since individual decision-making can be easily affected

by a range of factors such as individual attributes, external environments, and historical travel habits. Second, IHTF employs the transformer to learn temporal dynamics, which is an all-attention-based structure [127] that can handle the sequence data more efficiently particularly compared with RNNs. The efficiency is nontrivial in individual-level forecasting since the number of mobile devices that needs to predict is huge. This is also the main barrier to continuing the use of graph-based neural networks to forecast individual mobility. In the graph neural networks, each device should be viewed as a node, which would result in a giant graph with ten thousand nodes. Third, due to the attention mechanism and the variable selection module, the IHTF can have some interpretability regarding the learned temporal patterns and the variable importance, which provide insights into the underlying factors that influence travel decisions.

6.2.3.1 Model Structure

The IHTF mainly takes inspiration from the reverse time attention model [216], the transformer [127], and the temporal fusion transformer [217]. **Figure 6-12** shows the architecture of IHTF. Specifically, the IHTF contains several components:

1) Variable preprocessing: a hybrid module to deal with data preprocessing, including variable embedding and data rolling.

2) Gated residual network (GRN): a unit to control the degree of complexity and to skip over irrelevant information [128], including dropout, residual connection, layer normalization, and several non-linear transformations.

3) Variable selection: a network to select and fuse variables.

4) GRU-based positional encoding: an S2S layer using a GRU encoder-decoder [208] to enhance temporal locality sensitivity.

5) Transformer [127]: a module similar to the vanilla transformer, including a multi-head self-attention module, a position-wise feed-forward network, and an encoder-decoder architecture.

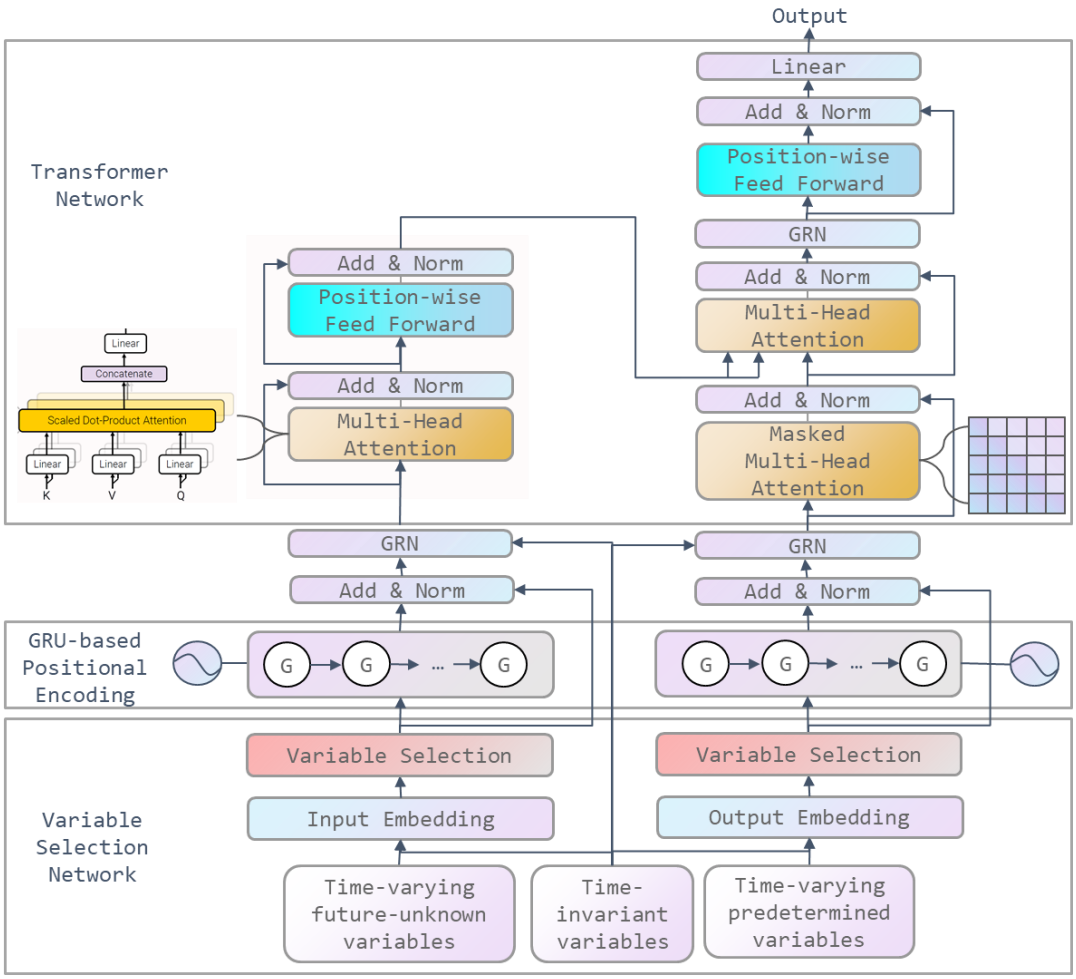


Figure 6-12 IHTF architecture

6.2.3.2 Variable Preprocessing

Variable preprocessing is essential for accelerating model training and enhancing model performance. Let $[\boldsymbol{x}_{i,t}, y_{i,t}] = [\boldsymbol{b}_{i,t}, \boldsymbol{c}_{i,t}]$, where $\boldsymbol{b}_{i,t} \in \mathbb{R}^{d_b}$ is the set of

categorical variables and $\boldsymbol{c}_{i,t} \in \mathbb{R}^{d_c}$ is the set of continuous variables. Similarly to various sequence transduction models [208], this section employed entity embedding to convert categorical variables into learned low-dimensional representations, for which the embedding dimension is empirically determined by the number of categories of $\boldsymbol{b}_{i,t}$. Letting $d_{Eb}^{(k)}$ be the embedding dimension of the k[th] categorical variable, the embedding representation of $\boldsymbol{b}_{i,t}$ can be written as: $\boldsymbol{b}'_{i,t} = \mathrm{Embed}(\boldsymbol{b}_{i,t}) \in \mathbb{R}^{\sum_{k=1}^{d_b} d_{Eb}^{(k)}}$. Next, embedding representations and continuous variables are concatenated to construct the encoder-decoder matrix $[\boldsymbol{b}'_{i,1:d_T}, \boldsymbol{c}'_{i,1:d_T}] \in \mathbb{R}^{d_T \times (d_c + \sum_{k=1}^{d_b} d_{Eb}^{(k)})}$. The rolling window is applied to the matrix, splitting it into multiple slices, with each one containing an encoder in length d_E and a decoder in length d_D (**Figure 6-7**). Hereinafter the index of the device is omitted for simplification since the processing on each device follows the same flow.

6.2.3.3 Gated Residual Network (GRN) Unit

The IHTF mainly relied on the GRN to control the degree of nonlinearity and to skip over unimportant information. GRN is a gating mechanism incorporating dropout, residual connection, layer normalization, and several non-linear transformations [128]. The main form of GRN is as follows [128]:

$$\mathrm{GRN}(\boldsymbol{\chi}) = \mathrm{LayerNorm}\big(\boldsymbol{\chi} + \mathrm{GLU}(\boldsymbol{\delta}_1)\big) \quad (6\text{-}6)$$

$$\mathrm{GLU}(\boldsymbol{\delta}_1) = \sigma(\boldsymbol{W}_1\boldsymbol{\delta}_1 + \boldsymbol{b}_1) \odot (\boldsymbol{W}_2\boldsymbol{\delta}_1 + \boldsymbol{b}_2) \quad (6\text{-}7)$$

$$\boldsymbol{\delta}_1 = \mathrm{Dropout}(\boldsymbol{W}_3\boldsymbol{\delta}_2 + \boldsymbol{b}_3) \quad (6\text{-}8)$$

$$\boldsymbol{\delta}_2 = \mathrm{ELU}(\boldsymbol{W}_4\boldsymbol{\chi} + \boldsymbol{b}_4) \quad (6\text{-}9)$$

where $\boldsymbol{\chi} \in \mathbb{R}^{d_m}$ is the input vector; LayerNorm is a standard layer normalization; $\boldsymbol{\delta}_1 \in \mathbb{R}^{d_m}$ and $\boldsymbol{\delta}_2 \in \mathbb{R}^{d_m}$ are intermediate layers; $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4 \in \mathbb{R}^{d_m \times d_m}$ are weights and $\boldsymbol{b_1}, \boldsymbol{b_2}, \boldsymbol{b_3}, \boldsymbol{b_4} \in \mathbb{R}^{d_m}$ are biases; $\odot$ is the element-wise Hadamard product; $\mathrm{ELU}(.)$ is the element-wise exponential linear unit activation function; $\sigma(.)$ is the element-wise sigmoid activation function. During training, dropout is applied to $\boldsymbol{\delta}_1$ layer as a regularizer to avoid overfitting.

The intuition about the role of GRN is described as follows:

1) When $\boldsymbol{W_4\chi} + \boldsymbol{b_4} > 0$, $\boldsymbol{\delta_2} = \boldsymbol{W_4\chi} + \boldsymbol{b_4}$ and $\boldsymbol{\delta}_1$ is the linear transformation of $\boldsymbol{\chi}$; when $\boldsymbol{W}_4\boldsymbol{\chi} + \boldsymbol{b}_4 \ll 0$, $\boldsymbol{\delta}_2 \to -1$ and $\boldsymbol{\delta}_1$ is a constant; only when $\boldsymbol{W}_4\boldsymbol{\chi} + \boldsymbol{b}_4$ located in a proper range, the nonlinear transformation of $\boldsymbol{\chi}$ is activated. Such nature allows the GRN to control the degree of nonlinearity of input variable $\boldsymbol{\chi}$.

2) When $\boldsymbol{W}_1\boldsymbol{\delta}_1 + \boldsymbol{b}_1 \ll 0$, $\sigma(\boldsymbol{W}_1\boldsymbol{\delta}_1 + \boldsymbol{b}_1) \to 0$ and $\mathrm{GLU}(\boldsymbol{\delta}_1) \to 0$, $\mathrm{GRN}(\boldsymbol{\chi})$ becomes a layer entirely skipping nonlinear parts; when $\boldsymbol{W}_1\boldsymbol{\delta}_1 + \boldsymbol{b}_1 \gg 0$, $\sigma(\boldsymbol{W}_1\boldsymbol{\delta}_1 + \boldsymbol{b}_1) \to 1$ and $\mathrm{GLU}(\boldsymbol{\delta}_1) \to \boldsymbol{W}_2\boldsymbol{\delta}_1 + \boldsymbol{b_2}$, now the nonlinear part contributes most to the layer output. Such nature allows the GRN to control the contribution of nonlinear parts to the GRN output.

6.2.3.4 Variable Selection

Unlike most univariate time series forecasting methods that only utilize self-information to construct autoregression, this section incorporated a variety of exogenous variables into the model. However, relations between those exogenous variables and target activity chains are unknown, and the types and dimensions of exogenous variables are hybrid. Thus, a variable selection network to fuse different variables and assign canonical variable-wise selection weights is essential [216, 217].

Specifically, the IHTF assigned a weight vector to each exogenous variable and weighted combined the product of weights and variables to generate final selection outputs. The learned weights of exogenous variables can also serve as the feature importance for model interpretation. Considering the nature of the data sources used in this study, the variable selection network is divided into three parts: static variables selection, time-varying encoder variables selection, and time-varying decoder variables selection. **Figure 6-13** shows the high-level architecture of the variable selection network, with individual parts described in detail in the subsequent sections.

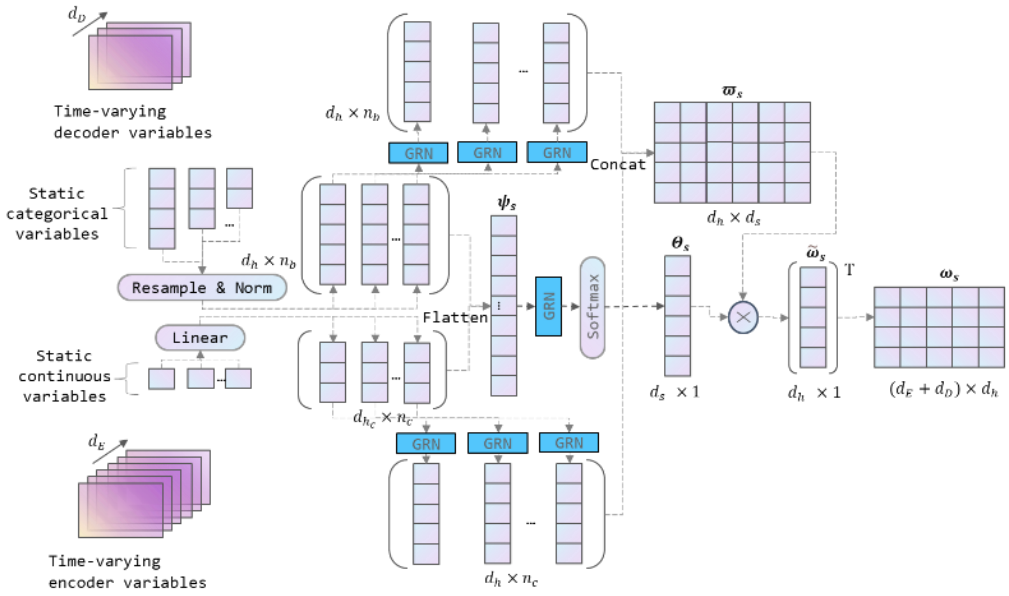


Figure 6-13 Variable selection network architecture

Static variables: Let d_h be the model's hidden size and d_{h_c} be the model's hidden continuous size, each categorical variable is resampled from the dimension $d_{Eb}^{(k)}$ to d_h, and each continuous variable is linearly transformed into a dimension of d_{h_c}. To reduce the computation load, the temporal dimension was removed since all variables are static. All transformed variables were concatenated and flattened as a

vector $\boldsymbol{\psi_s} \in \mathbb{R}^{n_b d_h + n_c d_{h_c}}$, where n_b is the number of categorical static variables and n_c is the number of continuous static variables. $\boldsymbol{\psi_s}$ was then fed into a GRN, followed by a Softmax activation function, to generate the vector of selection weights for static variables:

$$\boldsymbol{\Theta_s} = \text{Softmax}\big(\text{GRN}(\boldsymbol{\psi_s})\big) \in \mathbb{R}^{n_b + n_c = d_s} \qquad (6\text{-}10)$$

Meanwhile, each variable was fed through the variable-wise GRN separately to generate its context $\boldsymbol{\varpi_s^{(k)}} \in \mathbb{R}^{d_h}$, where *k* denotes the kth static variable. All static variables were concatenated as a matrix $\boldsymbol{\varpi_s} = [\boldsymbol{\varpi_s^{(1)}}, \ldots, \boldsymbol{\varpi_s^{(d_s)}}] \in \mathbb{R}^{d_h \times d_s}$. The output of static variables selection $\boldsymbol{\tilde{\omega}_s}$ is the weighted sum of variable contexts: $\boldsymbol{\tilde{\omega}_s} = \boldsymbol{\varpi_s \Theta_s} \in \mathbb{R}^{d_h}$. Last, $\boldsymbol{\tilde{\omega}_s}$ was expanded to the whole range by repeating $d_E + d_D$ times. The final output of the static variables selection layer is denoted as $\boldsymbol{\omega_s} = [(\boldsymbol{\tilde{\omega}_s}^{\boldsymbol{T}})_{\times (d_E + d_D)}] \in \mathbb{R}^{(d_E + d_D) \times d_h}$.

Time-varying encoder variables: Time-varying encoder variables include time-varying future-unknown variables $\boldsymbol{u}_{i,t} \in \mathbb{R}^{d_u}$ and time-varying predetermined variables $\boldsymbol{v}_{i,t} \in \mathbb{R}^{d_v}$. The same flow in selecting static variables is followed but remained the temporal dimension across the whole process. Alternatively, the variable selection weights $\boldsymbol{\Theta_u} \in \mathbb{R}^{d_E \times (d_u + d_v)}$ and the variable context $\boldsymbol{\varpi_u} \in \mathbb{R}^{d_E \times d_h \times (d_u + d_v)}$ both expanded their dimension in the temporal direction, and the final output of the time-varying encoder variables selection layer is denoted as $\boldsymbol{\omega_u} = \text{Squeeze}(\boldsymbol{\varpi_u \Theta_u}) \in \mathbb{R}^{d_E \times d_h}$.

Time-varying decoder variables: Time-varying decoder variables include time-varying predetermined variables $\boldsymbol{v}_{i,t} \in \mathbb{R}^{d_v}$. Only those future-known variables

are fed into the decoder to prevent leftward information leakage, i.e., prevent learning knowledge from data in the future. Again, the selection process is similar to selecting static variables except the variable dimension is expanded in the temporal dimension. Finally, for time-varying decoder variables, outputs include the variable selection weights $\boldsymbol{\theta_v} \in \mathbb{R}^{d_D \times d_v}$, the variable context $\boldsymbol{\varpi_v} \in \mathbb{R}^{d_D \times d_h \times d_v}$, and the layer outcome $\boldsymbol{\omega_v} = \text{Squeeze}(\boldsymbol{\varpi_v \theta_v}) \in \mathbb{R}^{d_D \times d_h}$.

6.2.3.5 GRU-based Positional Encoding

The vanilla transformer (ref?) contains no recurrence but has a positional encoding module to inject the order of the sequence into the input embedding. The major weaknesses of such design are its insensitivity to the local context, while most time series are highly influenced by their locally surrounding values [218]. To enhance the locality, this study incorporates an S2S layer using a GRU encoder-decoder [208] to replace the standard positional encoding. The main form of GRU is as follows:

$$\boldsymbol{c}_t = \left(1 - \boldsymbol{\Gamma}_{u,t}\right) \odot \boldsymbol{c}_{t-1} + \boldsymbol{\Gamma}_{u,t} \odot \tilde{\boldsymbol{c}}_t \quad (6\text{-}11)$$

$$\boldsymbol{\Gamma}_{u,t} = \sigma(\boldsymbol{W}_u[\boldsymbol{c}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_u) \quad (6\text{-}12)$$

$$\boldsymbol{\Gamma}_{r,t} = \sigma(\boldsymbol{W}_r[\boldsymbol{c}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r) \quad (6\text{-}13)$$

$$\tilde{\boldsymbol{c}}_t = \tanh\left(\boldsymbol{W}_c\left[\boldsymbol{\Gamma}_{r,t} \odot \boldsymbol{c}_{t-1}, \boldsymbol{x}_t\right] + \boldsymbol{b}_c\right) \quad (6\text{-}14)$$

where $\boldsymbol{c}_t \in \mathbb{R}^{d_h}$ is the activation of the GRU at hour *t*, which is a linear combination of the previous activation $\boldsymbol{c}_{t-1} \in \mathbb{R}^{d_h}$ and the candidate activation $\tilde{\boldsymbol{c}}_t \in \mathbb{R}^{d_h}$; $\boldsymbol{\Gamma}_{u,t} \in \mathbb{R}^{d_h}$ is the update gate that decides how much the unit updates its activation at hour *t*; $\boldsymbol{x}_t \in \mathbb{R}^{d_h}$ is the input vector at hour *t*, here $\boldsymbol{x}_t$ refers to the output of variable selection, e.g., $(\boldsymbol{\omega_v})_t$ or $(\boldsymbol{\omega_u})_t$; $\boldsymbol{\Gamma}_{r,t} \in \mathbb{R}^{d_h}$ is the reset gate to determine how much

the candidate activation depends on the previous activation; $\boldsymbol{W}_u, \boldsymbol{W}_r, \boldsymbol{W}_c \in \mathbb{R}^{d_h \times d_h}$ are weights and $\boldsymbol{b}_u, \boldsymbol{b}_r, \boldsymbol{b}_c \in \mathbb{R}^{d_h}$ are the biases.

Following the transformer structure, the output of the variable selection network, $\boldsymbol{\omega_u}$ (weighted time-varying encoder variables) and $\boldsymbol{\omega_v}$ (weighted time-varying decoder variables), are separately fed into the GRU encoder-decoder, generating $\boldsymbol{\omega'_u} \in \mathbb{R}^{d_E \times d_h}$ and $\boldsymbol{\omega'_v} \in \mathbb{R}^{d_D \times d_h}$, respectively. $\boldsymbol{\omega'_u}$ and $\boldsymbol{\omega'_v}$ were then fed into GRN and stacked as the output of GRU encoder-decoder, i.e., $\boldsymbol{\Psi} = \begin{bmatrix} \mathrm{GRN}_u(\boldsymbol{\omega'_u}) \\ \mathrm{GRN}_v(\boldsymbol{\omega'_v}) \end{bmatrix} \in \mathbb{R}^{(d_E+d_D) \times d_h}$. The weighted static metadata $\boldsymbol{\omega_s}$ are not passed through the GRU encoder-decoder since the order of static sequence is unnecessary. Instead, it is directly combined with $\boldsymbol{\Psi}$ to enrich the static context using GRN with some changes in the bottom layer (i.e., Eq. *(6-9)*):

$$\boldsymbol{\delta_2} = \mathrm{ELU}(\boldsymbol{W_4}\boldsymbol{\Psi} + \boldsymbol{W_5}\boldsymbol{\omega_s} + \boldsymbol{b_4}) \qquad (6\text{-}15)$$

Note that $\boldsymbol{\Psi}$ and $\boldsymbol{\omega_s}$ are in the same dimension (i.e., $\mathbb{R}^{(d_E+d_D) \times d_h}$) and thus can be added. The final output of the GRU encoder-decoder is denoted as $\boldsymbol{\Phi} = \mathrm{GRN}_{\Phi}(\boldsymbol{\Psi}, \boldsymbol{\omega_s}) \in \mathbb{R}^{(d_E+d_D) \times d_h}$.

6.2.3.6 Multi-head Attention

The multi-head attention is borrowed from the transformer to capture the temporal dependence. Aided by the attention-based design, the model can access any part of the history regardless of its distance to the forecasting point, making it more suitable for capturing recurring patterns with long-term dependencies. Meanwhile, time series in the transportation domain often have multiple seasonality. The multi-head attention module allows the model to capture different recurring patterns, enhancing the

performance and flexibility in modeling different types of time series. The attention mechanism used in the transformer is known as one-head scaled dot-product attention. In general, it can be described as mapping a query $\boldsymbol{Q}$ and a set of key ($\boldsymbol{K}$) - value ($\boldsymbol{V}$) pairs to an output:

$$\boldsymbol{Q} = \boldsymbol{\Phi}_{(d_E+1:d_E+d_D),:}\boldsymbol{W_Q}, \boldsymbol{K} = \boldsymbol{\Phi W_K}, \boldsymbol{V} = \boldsymbol{\Phi W_V} \quad (6\text{-}16)$$

$$\text{Att}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V} \quad (6\text{-}17)$$

where $\boldsymbol{Q} \in \mathbb{R}^{d_D \times d_h}, \boldsymbol{K} \in \mathbb{R}^{(d_E+d_D)\times d_h}, \boldsymbol{V} \in \mathbb{R}^{(d_E+d_D)\times d_h}$ are the query, key, and value, which are calculated by multiplying the output of GRU encoder-decoder (i.e., $\boldsymbol{\Phi}$) by their learned weight matrices. Note that only predictions are queried, i.e., the $\boldsymbol{Q}$ was generated only from the decoder sequence of $\boldsymbol{\Phi}$. In addition, decoder masking is applied to the decoder sequence to preserve causal information flow, i.e., to ensure that each time point can only attend to information preceding it.

Let # attention heads be n_h, multi-head attention is to linearly project queries, keys, and values in Eq. *(6-17)* n_h times with different, learned linear projections:

$$\aleph = \text{MultiAtt}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = [\boldsymbol{H}_1, \dots, \boldsymbol{H}_{n_h}]\boldsymbol{W_M} \quad (6\text{-}18)$$

$$\boldsymbol{H}_i = \text{Att}\left(\boldsymbol{Q}\boldsymbol{W}_{\boldsymbol{Q}}^{(i)}, \boldsymbol{K}\boldsymbol{W}_{\boldsymbol{K}}^{(i)}, \boldsymbol{V}\boldsymbol{W}_{\boldsymbol{V}}^{(i)}\right) \quad (6\text{-}19)$$

where $\boldsymbol{H}_i \in \mathbb{R}^{d_D \times d_a}$ is the i[th] attention head; $\boldsymbol{W}_{\boldsymbol{Q}}^{(i)} \in \mathbb{R}^{d_h \times d_a}$, $\boldsymbol{W}_{\boldsymbol{K}}^{(i)} \in \mathbb{R}^{d_h \times d_a}, \boldsymbol{W}_{\boldsymbol{V}}^{(i)} \in \mathbb{R}^{d_h \times d_a}$ are learned weights for the i[th] attention head; $\boldsymbol{W_M} \in \mathbb{R}^{(d_a n_h)\times d_h}$ is the learned weight linearly combining outputs concatenated from all heads; d_a is the dimension of attention, e.g., for one-head attention $d_a = d_h$ and for multi-head attention $d_a = d_h/n_h$; $\aleph \in \mathbb{R}^{d_D \times d_h}$ is the output of the multi-head attention layer.

6.2.4 Model interpretation methods

This study interpreted the IHTF from two aspects. First, variables selection weights, i.e., $\boldsymbol{\Theta_s}, \boldsymbol{\Theta_u}, \boldsymbol{\Theta_v}$, were rescaled into 0-100% respectively to represent the relative variable importance. Note that $\boldsymbol{\omega_s}, \boldsymbol{\omega_u}, \boldsymbol{\omega_v}$ were fed into the model separately; thus a cross-comparison among them was unattainable:

$$\Theta^{(k)}{}_s = \frac{\boldsymbol{\Theta}^{(k)}{}_s}{\sum_{i=1}^{d_s} \boldsymbol{\Theta}^{(i)}{}_s}, \Theta^{(k)}{}_u = \frac{\boldsymbol{\Theta}^{(k)}{}_u}{\sum_{i=1}^{d_u+d_v} \boldsymbol{\Theta}^{(i)}{}_u}, \Theta^{(k)}{}_v = \frac{\boldsymbol{\Theta}^{(k)}{}_v}{\sum_{i=1}^{d_v} \boldsymbol{\Theta}^{(i)}{}_v} \qquad (6\text{-}20)$$

where $\Theta^{(k)}{}_s$ is the relative importance of the k^{th} static variable and $\boldsymbol{\Theta}^{(k)}{}_s$ is the k^{th} vector of variable selection weight $\boldsymbol{\Theta_s}$; $\Theta^{(k)}{}_u$ is the relative importance of the k^{th} encoder variable and $\boldsymbol{\Theta}^{(k)}{}_u$ is the k^{th} vector of variable selection weight $\boldsymbol{\Theta_u}$; $\Theta^{(k)}{}_v$ is the relative importance of the k^{th} decoder variable and $\boldsymbol{\Theta}^{(k)}{}_v$ is the k^{th} vector of variable selection weight $\boldsymbol{\Theta_v}$.

Note that although the relative importance can be extracted from IHTF, it is different from the relative importance in tree-based models, which has been introduced in Chapter 4. The IHTF input structures are different from those of tree-based models. In IHTF, variables were partitioned into three groups (static, time-varying future-unknown, and time-varying predetermined). Feature importance was computed separately in each group and was not comparable among groups. In tree-based models, all features were fed and compared as one group, including the lagged target and all external variables.

The second focus of interpretation was on multi-head attention. Digging into the pattern of attention weight across the encoder sequence can uncover whether the model successfully captures the temporal seasonality [216, 219]. Unlike previous

studies sharing values in each attention head [217], this study retained the head-specific weight for each attention head. Such a design can be more flexible to disentangle the temporal interaction when time series have different seasonality. Specifically, the attention weight of head *i* was formed as:

$$\mathrm{AttW}\left(\boldsymbol{Q}^{(i)}, \boldsymbol{K}^{(i)}\right) = \mathrm{softmax}\left(\frac{\boldsymbol{Q}^{(i)}\boldsymbol{W}_{\boldsymbol{Q}}^{(i)}\left(\boldsymbol{K}^{(i)}\boldsymbol{W}_{\boldsymbol{K}}^{(i)}\right)^{T}}{\sqrt{d_k}}\right) \in \mathbb{R}^{d_D \times (d_E + d_D)} \quad (6\text{-}21)$$

where the notations have the same meaning as Eqs. *(6-16)* to *(6-19)*.

Intuitively, Eq. *(6-21)* can be interpreted as in a specific attention head, when predicting each future step in the decoder sequence $[d_E + 1, d_E + 2, \ldots, d_E + d_D]$, how much attention the model paid to each step across the whole "encoder + masked decoder" sequence $[1,2,\ldots, d_E, d_E + 1, \ldots, d_E + d_D]$. Note that, aided by the decoder masking, attention will only be attached to time steps before the forecasting point. For example, when predicting the $d_E + 2$ step, the attention weight after that step is zero: $[\mathrm{AttW}_1, \mathrm{AttW}_2, \ldots, \mathrm{AttW}_{d_E}, \mathrm{AttW}_{d_E+1}, 0 \ldots, 0]$.

6.2.5 Losses and metrics for imbalanced classification

6.2.5.1 Classification metrics

The outcome of a classification model would be measured via precision, recall, and F1-score, globally or separately by classes. Specifically, precision indicates among those predicted positive, how many of them are actually positive. Recall indicates how many of the actual positives are successfully captured by the model (In the information domain, recall is the fraction of the relevant documents that are successfully retrieved). The F1 score is a combination of precision and recall, which

can seek a balance between the two measures in an imbalanced classification. Let the confusion matrix of a binary classification being **Table *6-5***, the precision, recall, and F1-score are calculated as Eqs. *(6-22)* to *(6-24)*, respectively.

Table 6-5 Illustration of a confusion matrix (Binary classification)

		Predicted condition	
	Total population = P+N	**Positive (PP)**	**Negative (PN)**
Actual condition	**Positive (P)**	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

$$Precision = \frac{TP}{TP + FP} \qquad (6\text{-}22)$$

$$Recall = \frac{TP}{TP + FN} \qquad (6\text{-}23)$$

$$F1 = \frac{2TP}{2TP + FP + FN} \qquad (6\text{-}24)$$

where TP is true positive (the model correctly predicts the positive class); FP is false positive (the model incorrectly predicts the positive class); FN is false negative (the model incorrectly predicts the negative class).

In a multiclass task, precision, recall, and F1 score can be applied to each class independently. This study mainly focuses on the “Home->Work”, “Work->Home”, “Others”, and “Stationary”, which are the most widely considered activity types in previous studies. In addition, a global measure is calculated via a weighted average of measures across all classes, with the weight calculated by the support of each class. Last, since the “Stationary” class accounts for a disproportionate portion among all classes while this study cares more about those non-stationary activities, another global measure excluding the “Stationary” is calculated to measure the model

performance in capturing non-stationary activities. Using the F1 score as an example, the two global metrics can be calculated as:

$$Avg(F1) = \sum_{\mathbb{C}} \frac{1}{n_c} \frac{2TP_c}{2TP_c + FP_c + FN_c} \quad (6\text{-}25)$$

$$Avg\_m(F1) = \sum_{\mathbb{C}\setminus\{'Stationary'\}} \frac{1}{n_c} \frac{2TP_c}{2TP_c + FP_c + FN_c} \quad (6\text{-}26)$$

where $\mathbb{C}$ is the set of all activities, n_c is # samples in each activity (i.e., $P_c + N_c$ in **Table 6-5**), $Avg(F1)$ is the global average of F1 score across all classes and $Avg\_m(F1)$ is the global average of F1 score across all classes except "Stationary".

6.2.5.2 Classification losses under imbalanced distribution

Unlike the regression problem, the aforementioned metrics in the classification problem cannot be directly used as loss functions since they are not differentiable. Hence, the cross-entropy loss is widely used as the loss function to approximate the metrics. However, one main issue that should be addressed in this study is the imbalanced distribution of activity labels. As shown in **Figure 6-7**, most (86.50%) activity types are stationary. If the classical cross-entropy loss is used as the loss function, the model tends to predict all the activity as stationary, leading to poor performance, particularly for the minority classes. Therefore, the loss functions that are designed for imbalanced classification should be introduced. This section tested a set of popular loss functions borrowed from the semantic segmentation domain [43] to address the imbalanced classification issue, including Cross-entropy Loss, Focal Loss [220], Dice Loss [221], and Tversky Loss [222], as well as their weighted forms.

Cross-entropy Loss: In a multi-class classification problem, the standard Cross-entropy Loss for one sample is defined as:

$$CE(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{c=1}^{C} \alpha_c y_c log(p_c) \quad (6\text{-}27)$$

where $\boldsymbol{y}$ is the one-hot encoding scheme of ground truth and y_c is its c[th] value; $\boldsymbol{p}$ is the vector recording the predicted values for each class and p_c is its c[th] value; C is the total number of classes in the classification problem; α_c is the weight for class *c*, which is set as 1 for standard cross-entropy loss.

Example: In a 3-class problem ($C = \mathbf{3}$), a sample's true label is class 0, then $\boldsymbol{y} = [1,0,0]$. Assume the model give a prediction $\boldsymbol{p} = [0.7,0.2,0.1]$. Then the unweighted Cross-entropy Loss is: $CE(\boldsymbol{p}, \boldsymbol{y}) = -(1 * log(0.7) + 0 * log(0.2) + 0 * log(0.1))$.

Focal Loss: The Focal Loss extends on the Cross-entropy Loss by adding a focusing parameter $(1 - p_c)^{\gamma}$. The idea is to differentiate between easy and hard samples and focus learning on hard samples:

$$FL(\boldsymbol{p}, \boldsymbol{y}) = -\sum_{c=1}^{C} \alpha_c (1 - p_c)^{\gamma} y_c log(p_c) \quad (6\text{-}28)$$

The Focal Loss is parameterized by γ, which controls the degree of down-weighting of easy-to-classify samples ($\gamma = 2$ in this study). Note that when $\gamma = 0$, $FL(\boldsymbol{p}, \boldsymbol{y}) = CE(\boldsymbol{p}, \boldsymbol{y})$. Setting $\gamma > 0$ reduces the relative loss for well-classified samples ($p_c > 0.5$), putting more focus on hard, misclassified samples.

Dice Loss: The Sørensen-Dice coefficient [221], aka the F1 score (Eq. *(6-24)*), is the most widely used metric for evaluating segmentation performance. It has been adapted as a differentiable loss function, i.e., the Dice Loss:

$$DL(\boldsymbol{p}, \boldsymbol{y}) = 1 - \sum_{c=1}^{C} \alpha_c \frac{2 y_c p_c + \varepsilon}{y_c + p_c + \varepsilon} \quad (6\text{-}29)$$

where ε is a smooth term to avoid extremely small $y_c + p_c$ ($\varepsilon = 10^{-5}$ in this study) Note that another form of Dice Loss uses ${y_c}^2 + {p_c}^2$ instead of $y_c + p_c$ in the denominator [223].

The main reason to use Dice Loss directly is that the actual goal of a classification model is to maximize the F1 score, while Cross-entropy Loss is just a proxy that is easier to maximize using backpropagation. However, the gradient of Dice Loss is more complex compared with Cross-entropy Loss, which would easily lead to an unstable training process when using Dice Loss as the objective function.

Tversky Loss: The Tversky Loss is similar to Dice Loss but enables controlling for the model focus on precision or recall by assigning two weights α and β to false positives (FPs) and false negatives (FNs), respectively:

$$TL(\boldsymbol{p}, \boldsymbol{y}) = 1 - \sum_{c=1}^{C} \alpha_c \frac{y_c p_c + \varepsilon}{y_c p_c + \alpha(1 - y_c) p_c + \beta y_c (1 - p_c) + \varepsilon} \qquad (6\text{-}30)$$

Note that when $\alpha = \beta = 0.5$, Tversky Loss is equal to Dice Loss. In order to weigh FNs more than FPs for highly imbalanced data, β is often set higher than α by placing more emphasis on FNs, which could result in a better balance of precision and recall, thereby improving performance for imbalanced data [222].

In all aforementioned losses, a class-wise weight coefficient, α_c, can be applied to each class to further address the imbalance:

$$\alpha_c = 1 - \frac{n_c}{\sum_{c=1}^{C} n_c} \qquad (6\text{-}31)$$

where n_c is the number of samples belonging to class *c* in the training dataset.

6.2.6 Experiment settings

6.2.6.1 Data description

As stated before, the whole model was trained and tested on a county-level dataset covering 2-month trips (January 1st to February 29th, 2020) from over 18,000 devices with their home located in Montgomery (MG) County, MD. **Figure 6-14** shows the spatial heatmap of home locations and trip destinations of observed devices. As shown, although the devices' homes are all located in MG county, their trips are spread across Maryland and extend outside the state (e.g., to Philadelphia and D.C.).

This study avoids including the data after February 2020 due to the abnormal effects of the COVID-19 pandemic on human travel behaviors. The encoder length and decoder length are set as 21*24 and 1*24, respectively. Alternatively, this study aims to use the hourly activities on the previous 21 days to predict the hourly activities on the next day. Datasets are split into training sets, validation sets, and test sets according to chronological order. The split ratio is 7:1.5:1.5.

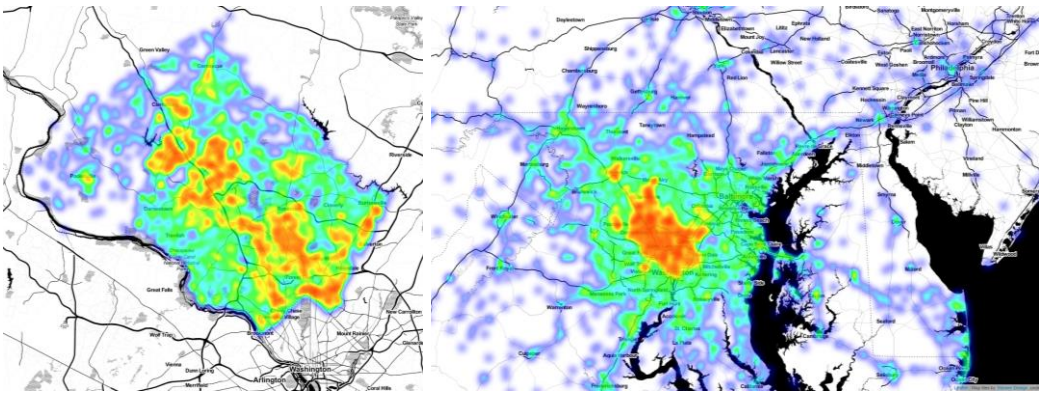


Figure 6-14 Heatmap of home (a) and destinations (b) of observed devices

6.2.6.2 Model training settings

The Adam algorithm was employed to minimize the model loss. Several callbacks were applied to monitor and adjust the model during the training process. The learning rate was set as 0.001 and was reduced by a factor of 10 when the validation loss did not improve for 4 epochs. Early stopping (patience = 6 epochs) was used to address overfitting and determines the optimal number of epochs, with the maximal epoch setting as 10 (each epoch contains 569,955 batches). Gradient clipping is enabled to avoid exploding gradients when gradients' global norm is greater than 0.5. Models were implemented in *Python* environment using *PyTorch* and were tuned using *Optuna* [198]. Similar to the previous chapter, hyperparameters were tuned via random search using the successive halving algorithm (SHA) [198]. IHTF reached its best performance when # GRU layers = 2, hidden size of GRU = 64, # attention heads = 6, drop-out rate = 0.3, and embedding size = 8.

All IHTF models can be trained with GPU acceleration without requiring massive computational resources. For example, using a Tesla P100 GPU with batch size = 32, an IHTF running through all batches in an epoch (consisting of 569,955 batches) cost roughly 10.95 h. After training, the IHTF took around 3.5 minutes to generate 24-hour forecasting for all devices. IHTF training and hyperparameter tuning can be further accelerated with hardware-specific optimizations.

6.2.6.3 Baseline settings

The forecasting performance of IHTF is compared with a variety of time series forecasting models that can be used in forecasting categorical targets, including traditional statistical models and deep learning models. The weighted focal loss is used as the loss function for all deep learning methods. The baseline models are:

- AS: A model that predicts all future activities as “Stationary”.
- HA: A probabilistic model that assigns the activity based on historical frequency.
- HMM [113]: A hidden Markov model.
- S2S-LSTM [224]: A LSTM-based S2S RNN model (2 layers, 64 hidden units).
- S2S-GRU [208]: A GRU-based S2S RNN model (2 layers, 64 hidden units).
- VTF [127]: A vanilla transformer.

6.2.7 Model prediction

6.2.7.1 Baseline comparison

Table *6-6* shows the average performances of all models on the testing dataset, with each panel ascendingly ranking by the value of $Avg\_m(.)$. Overall, in terms of recall and F1 score, IHTF outperforms all baselines in predicting non-stationary classes such as “Home->Work”, “Work->Home”, and “Others”, while in predicting the “Stationary” class, IHTF does not exhibit the best performance. This also leads to a significantly higher global recall and F1 score (i.e., the $Avg\_m(.)$ in **Table *6-6***) in IHTF for all non-stationary classes, while the global recall and F1 score in IHTF for all classes (i.e., the $Avg(.)$ in **Table *6-6***) is not the best. On the contrary, regarding the precision metric, IHTF outperforms baselines in predicting the “Stationary” class while for other non-stationary classes, it does not show the best score. In sum, IHTF tends to be more “adventurous” in predicting non-stationary activities benefiting from its stronger capability in mining complex patterns and ingesting multi-source information. This leads to a higher recall and F1 score but a relatively lower precision in predicting non-stationary classes. It is also worth noting that $Avg\_m(.)$ would be

a better metric in this study since it avoids the effects of the predominated "Stationary" class. An AS model that simply predicts all future activities as "Stationary" would lead to an 83.78% $Avg(F1\ Score)$, which, however, is meaningless in the real-world applications.

Comparing all baseline models, although the performance ranking was slightly different across metrics, deep learning models such as S2S-GRU, S2S-LSTM, and VTF broadly belonged to the first tier. These models all have complex network architectures to capture complex nonlinear temporal dependence from all individuals. HA and AS constantly presented the poorest performance. It is intuitive since the two models have simple designs, predicting the future value entirely based on historical frequency. Under such settings, many external effects and hidden patterns cannot be captured. Performance of HMM steadily laid in the middle tier, which confirms the difficulties in making accurate predictions using statistical models in highly-random individual-specific datasets.

Table 6-6 Model performance comparison: IHTF vs. baselines

Precision						
	Home->Work	Work->Home	Others	Stationary	$Avg(.)$	$Avg\_m(.)$
AS	0.00%	0.00%	0.00%	88.97%	79.16%	0.00%
HA	11.40%	6.56%	8.66%	90.21%	80.92%	28.61%
HMM	25.51%	13.15%	12.91%	90.79%	81.89%	37.11%
IHTF	35.14%	21.38%	17.57%	**93.86%**	**85.29%**	39.20%
S2SLSTM	34.39%	20.05%	17.03%	91.74%	83.18%	41.87%
S2SGRU	35.85%	19.30%	19.42%	91.91%	83.46%	42.72%
VTF	**41.14%**	**26.47%**	**20.83%**	92.15%	83.92%	**46.01%**
Recall						
	Home->Work	Work->Home	Others	Stationary	$Avg(.)$	$Avg\_m(.)$
AS	0.00%	0.00%	0.00%	100.00%	88.97%	0.00%
HA	11.18%	6.53%	9.27%	89.76%	80.55%	6.23%
HMM	24.71%	13.84%	13.85%	90.67%	81.81%	10.35%

S2SLSTM	33.92%	20.63%	18.70%	91.10%	82.69%	14.85%
S2SGRU	34.51%	18.80%	21.12%	91.56%	83.20%	15.81%
VTF	40.98%	25.85%	23.22%	**91.80%**	**83.68%**	18.15%
IHTF	**68.63%**	**44.39%**	**53.83%**	88.08%	81.72%	**30.41%**
F1 Score						
	Home->Work	Work->Home	Others	Stationary	$Avg(.)$	$Avg\_m(.)$
AS	0.00%	0.00%	0.00%	94.16%	83.78%	0.00%
HA	11.29%	6.54%	8.96%	89.99%	80.73%	10.21%
HMM	25.10%	13.49%	13.36%	90.73%	81.85%	16.07%
S2SLSTM	34.16%	20.33%	17.82%	91.42%	82.93%	21.80%
S2SGRU	35.16%	19.05%	20.23%	91.73%	83.33%	22.93%
VTF	41.06%	26.16%	21.96%	**91.98%**	**83.79%**	25.81%
IHTF	**46.48%**	**28.86%**	**26.49%**	90.88%	82.86%	**32.15%**

6.2.7.2 Loss functions comparison

Table *6-7* presents how the performance of IHTF varies across different loss functions. Seven types of loss functions are compared, including different forms (Dice Loss, Cross-entropy Loss, Tversky Loss with different α and β, and Focal Loss) and different weights (with or without weight). To reduce the computational load, only 1,000 devices are sampled from over 18,000 devices in the training set to test the performance of different loss functions. All the loss functions can be viewed as a strategy to balance the trade-off between predicting the minority (i.e., non-stationary classes) and the majority (i.e., stationary classes), which can be manifested by the trade-off between precision and recall.

Overall, the weighted Focal Loss performs the best recall and F1 score across all loss functions in predicting non-stationary classes such as “Home->Work”, “Work->Home”, and “Others”, indicating it successfully helps the model to focus more on the minority classes that are more difficult to predict. The classical (unweighted) Cross-entropy Loss presents the lowest recall and F1 score in predicting

non-stationary classes while it shows the highest score in predicting the stationary class. This indicates it fails to capture the patterns of minority classes, leading to a more conservative forecasting strategy. Tversky Loss adjusts the model focus between non-stationary classes and the stationary class by changing the combination of α and β. A lower α and higher β would result in the model being more adventurously in predicting non-stationary classes (i.e., higher recall and F1 score for non-stationary classes). The Dice Loss (unweighted) is an unweighted version of the Tversky Loss (0.5, 0.5), which presents less focus on minority classes

Table 6-7 Model performance across different loss functions (IHTF)

Precision						
	Home->Work	Work->Home	Others	Stationary	$Avg(.)$	$Avg\_m(.)$
Dice Loss (Unweighted)	59.78%	20.99%	19.30%	90.95%	82.60%	37.19%
Cross Entropy (Weighted)	56.51%	17.36%	19.23%	91.94%	83.38%	37.38%
Tversky Loss (0.6, 0.4) (Weighted)	53.55%	20.41%	20.68%	91.39%	82.99%	37.90%
Tversky Loss (0.7, 0.3) (Weighted)	50.40%	13.71%	22.07%	90.68%	82.34%	38.98%
Focal Loss (Weighted)	35.14%	21.38%	17.57%	93.86%	85.29%	39.20%
Tversky Loss (0.3, 0.7) (Weighted)	47.40%	25.39%	15.94%	**94.55%**	**85.72%**	40.05%
Tversky Loss (0.5, 0.5) (Weighted)	47.47%	37.91%	20.71%	92.47%	84.20%	41.27%
Cross Entropy	**62.98%**	**50.75%**	**51.22%**	89.51%	83.84%	**48.05%**
Recall						
	Home->Work	Work->Home	Others	Stationary	$Avg(.)$	$Avg\_m(.)$
Cross Entropy	25.69%	8.88%	2.26%	**99.81%**	**89.25%**	4.08%
Tversky Loss (0.7, 0.3) (Weighted)	37.25%	11.49%	16.06%	94.79%	85.50%	10.50%
Dice Loss (Unweighted)	32.35%	4.44%	24.78%	93.87%	84.92%	12.71%

Tversky Loss (0.6, 0.4) (Weighted)	35.49%	7.83%	28.61%	91.77%	83.28%	14.83%
Cross Entropy (Weighted)	37.45%	19.58%	31.84%	87.99%	80.17%	17.11%
Tversky Loss (0.5, 0.5) (Weighted)	38.63%	18.02%	35.94%	89.19%	81.47%	19.23%
Tversky Loss (0.3, 0.7) (Weighted)	41.18%	30.03%	48.06%	77.21%	71.57%	26.03%
Focal Loss (Weighted)	**68.63%**	**44.39%**	**53.83%**	88.08%	81.72%	**30.41%**
F1 Score						
	Home->Work	Work->Home	Others	Stationary	$Avg(.)$	$Avg\_m(.)$
Cross Entropy	36.49%	15.11%	4.33%	**94.38%**	**84.68%**	6.72%
Tversky Loss (0.7, 0.3) (Weighted)	42.84%	12.50%	18.59%	92.69%	83.81%	16.06%
Dice Loss (Unweighted)	41.98%	7.33%	21.70%	92.39%	83.62%	18.48%
Tversky Loss (0.6, 0.4) (Weighted)	42.69%	11.32%	24.01%	91.58%	83.04%	20.81%
Cross Entropy (Weighted)	45.05%	18.40%	23.98%	89.92%	81.64%	23.25%
Tversky Loss (0.5, 0.5) (Weighted)	42.59%	24.42%	26.28%	90.80%	82.64%	25.48%
Tversky Loss (0.3, 0.7) (Weighted)	44.07%	27.51%	23.94%	85.01%	77.51%	30.76%
Focal Loss (Weighted)	**46.48%**	**28.86%**	**26.49%**	90.88%	82.86%	**32.15%**

To better demonstrate the model performance in predicting non-stationary activities, this study introduces the MAPE of the number of hourly trips as a new metric to evaluate the model performance. **Figure 6-15** shows the prediction vs. actual evolution of hourly trip counts using different loss functions. As shown, classical (unweighted) Cross-entropy Loss tends to label most of the activity as "Stationary", leading to a much lower number of prediction trips compared with observations. On the contrary, the Tversky Loss (0.3, 0.7) tends to be excessively

aggressive in labeling non-stationary classes, resulting in a much higher number of prediction trips. Focal Loss (weighted), Dice Loss, and Tversky Loss (0.6, 0.4) are the top three loss functions that show the lowest MAPE, implying a good balance between the non-stationary classes and the stationary class is achieved by the model under these loss functions. In sum, a well-designed loss function can significantly improve the model performance in handling imbalanced classification. The weighted Focal Loss was finally chosen as the loss function due to its best performance in capturing minority activities out of the predominated stationary classes.

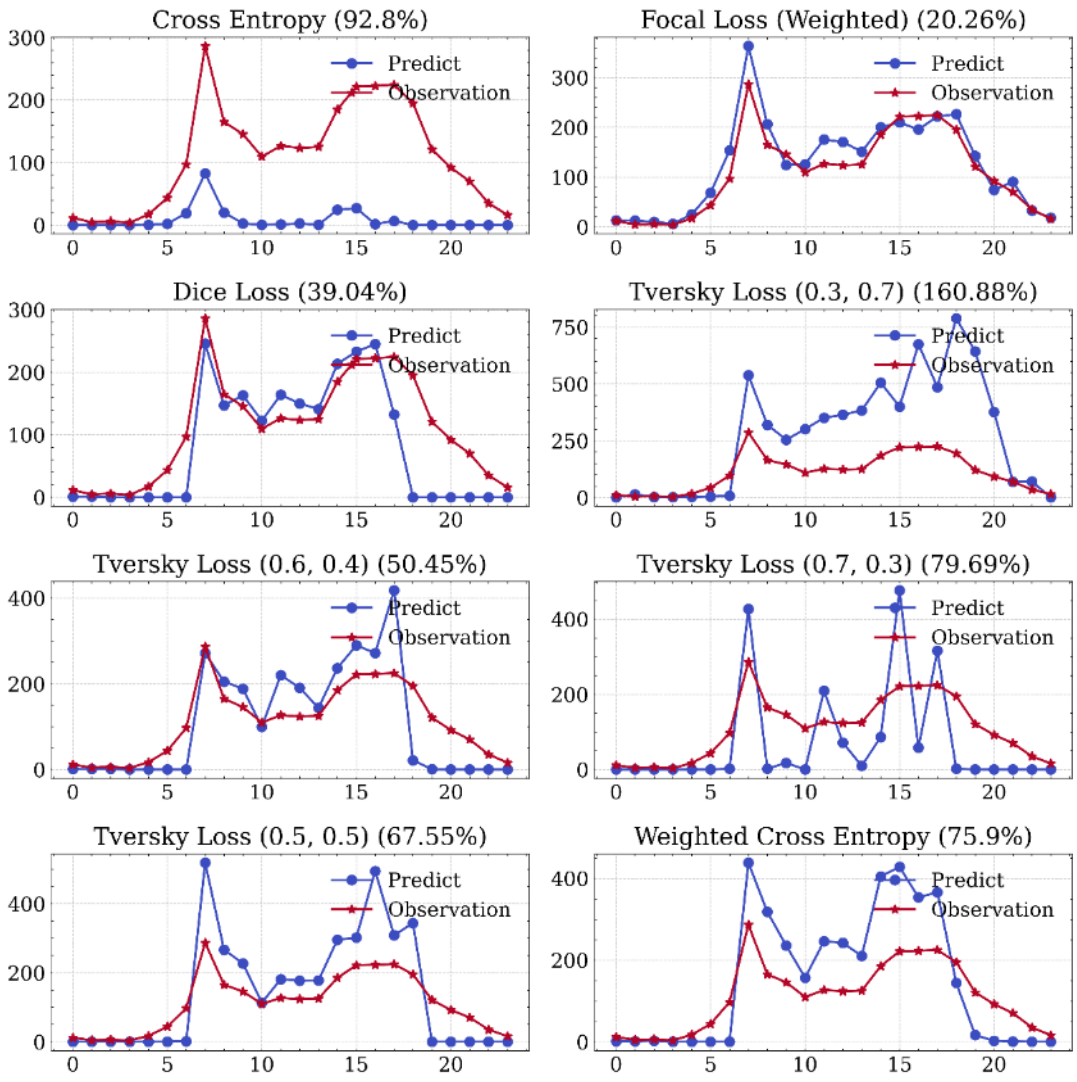
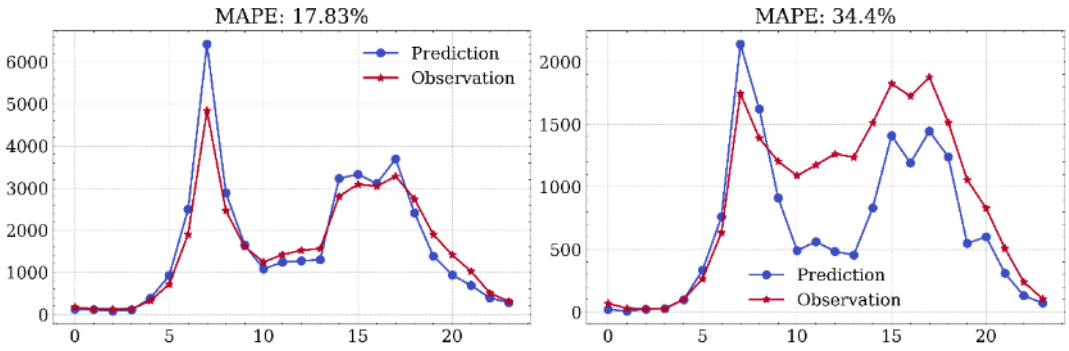


Figure 6-15 Predicted vs. real hourly trip counts using different loss functions

6.2.7.3 Performance comparison: workers vs. nonworkers

Understanding how model performance changes across individuals helps to diagnose the model and find its potential weakness. Since work status is considered one of the most important factors by the model (see **Figure 6-18**), this study separately analyzes the model performance in predicting the activities of workers and nonworkers. Results are reported in **Figure 6-16** and **Table *6-8***. As shown, the model performance in predicting workers is significantly higher than that of nonworkers, manifesting in

lower MAPE in hourly trip counts and higher precision, recall, and F1 score in predicting non-stationary activities. Such a finding is plausible since the activity patterns of workers are more regular than the nonworkers due to their regular commuting travels. For example, the F1 score of prediction the Home->Work for workers reaches 52.84% while for nonworkers they do not have such activities. Future studies should consider building and training different models to learn from the activity patterns of workers and nonworkers separately to explore whether that could improve the total model performance.



(a) (b)

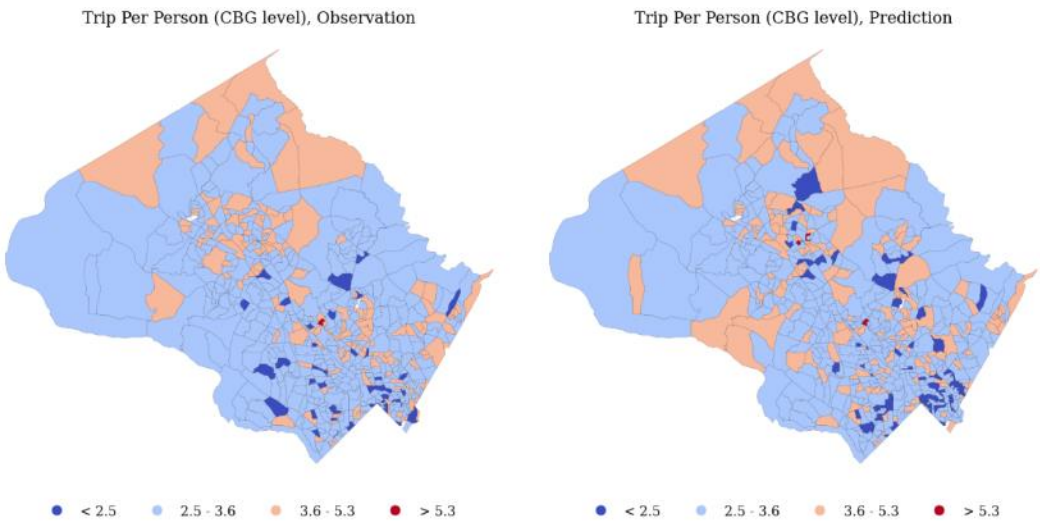
Figure 6-16 Predicted vs. real hourly trip counts: workers (a) vs. nonworkers (b)

Table 6-8 Model performance comparison: workers vs. nonworkers (IHTF)

Workers (12,231)			
Activity type	**Precision**	**Recall**	**F1 Score**
Stationary	93.10%	92.64%	92.87%
Others	17.57%	28.38%	21.70%
Work->Home	27.36%	65.64%	38.63%
Home->Work	39.93%	78.12%	52.84%
$Avg(.)$	83.80%	84.24%	83.53%
$Avg\_m(.)$	**41.11%**	**28.17%**	**29.31%**
Nonworkers (6,696)			
Activity type	**Precision**	**Recall**	**F1 Score**
Stationary	90.45%	94.18%	92.28%
Others	13.29%	10.94%	12.00%

Education->Home	20.15%	30.18%	24.17%
Home->Education	26.08%	48.68%	33.97%
$Avg(.)$	80.73%	83.53%	81.87%
$Avg\_m(.)$	35.50%	14.37%	18.95%

6.2.7.4 Performance across CBG



(a) (b)

Figure 6-17 Spatial distribution of trip rate: observation (a) vs. prediction (b)

Trip rate, i.e., the daily number of trips per person, is an important metric in classical travel demand models and travel surveys. It is thus critical to confirm whether the model can accurately predict trip rates at different spatial resolutions. This study compares the predicted and actual trip rates at both the county level and CBG level and reports the results in **Table *6-9*** and **Figure 6-17**. A highly consistent spatial distribution can be observed in **Figure 6-17** between observed and predicted trip rates, indicating the model can achieve acceptable accuracy in trip rate prediction. However, differences emerge when separately considering the accuracy in predicting workers and nonworkers. Overall, the trip rate of workers can be more accurately predicted by the model, yielding a MAPE of 3.136% at the county level and a MAPE

of 12.6% at the CBG level. However, the prediction accuracy for the trip rate of nonworkers is moderate. Although the observed trip rate of nonworkers is higher than that of workers, the model tends to underestimate the trip rates of nonworkers and generated a lower prediction outcome, which is consistent with the hourly total trip counts shown in **Figure 6-16**. One explanation is that the travel patterns of nonworkers are more randomly distributed. Hence, the model seems less confident in predicting non-stationary activities and tends to predict more labels as "Stationary".

Table 6-9 Model performance comparison: trip rate (IHTF)

County-level Trip Per Person, Daily			
	Prediction	Observation	MAPE
Workers (12,231)	3.223	3.125	3.136%
Nonworkers (6,696)	2.545	3.201	20.494%
CBG-level Trip Per Person, Daily (614 CBGs in MG county)			
	Prediction	Observation	MAPE
Workers (12,231)	3.248	3.271	12.597%
Nonworkers (6,696)	2.595	3.355	27.998%

6.2.8 Model interpretation

One main advantage of IHTF is its strong interpretability achieved by examining its built-in parameters. **Figure 6-18** shows the relative importance of external variables in IHTF. In IHTF, variables were divided into static variables, encoder variables (time-varying variables), and decoder variables (time-varying predetermined variables) and were computed separately. As shown, among the static variables, the work status is the most important factor, followed by the CBG of the devices' home locations. Age, income, and gender are relatively less important, which may be because they are variables inferred from aggregated surveys, rather than real

socioeconomic and demographic characteristics. Among the encoder variables, the historical activity chain is the most important factor, followed by whether the day occurs on a weekend. Last, among decoder variables, the day of the week and whether the day is a weekend are the most important factors. In sum, the relative importance extracted from the IHTF is intuitive. Work status and weekends are important since they can lead to significantly different activity patterns (**Figure 6-10**). The reasonable identification of the important factors also indicates that the IHTF can successfully learn useful knowledge from multi-dimensional external variables.

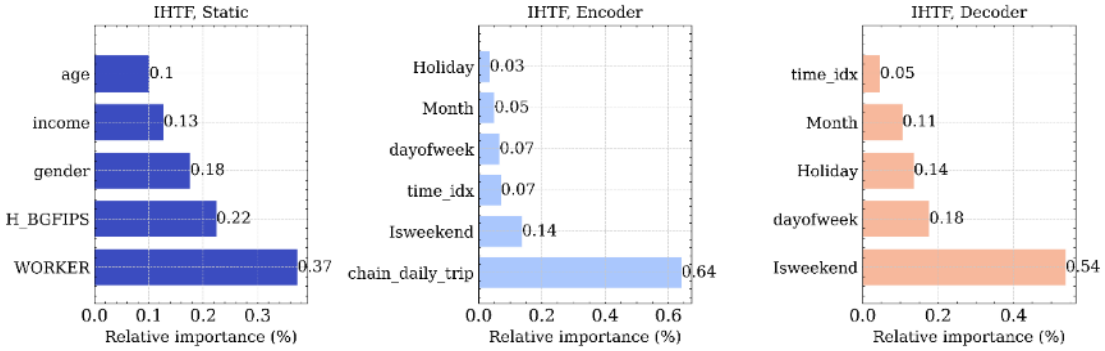


Figure 6-18 Relative feature importance in IHTF

Attention weight can be used to examine how much importance the model attaches to historical activities in predicting future activities. In contrast to other time series methods which rely on prior knowledge to determine seasonality, the attention mechanisms can automatically learn the cyclical patterns from the data itself. **Figure 6-19** depicts temporal patterns of attention weight averaged across all attention heads in a 24-step IHTF. The model was intended to predict the activities in the future 24 hours using data from the previous 21*24 hours (i.e., the previous three weeks). Each curve represents the average attention weight that the model attaches to past hours when predicting the k^{th} hour. A clear daily pattern and weekly pattern can be observed

for each future step respectively, indicating the IHTF successfully captured the underlying seasonality in the activities. Additionally, the attention on the previous hour also spiked but not as strongly as the daily spike, which indicates the activities that happened during the previous hour also show strong relations with the prediction.

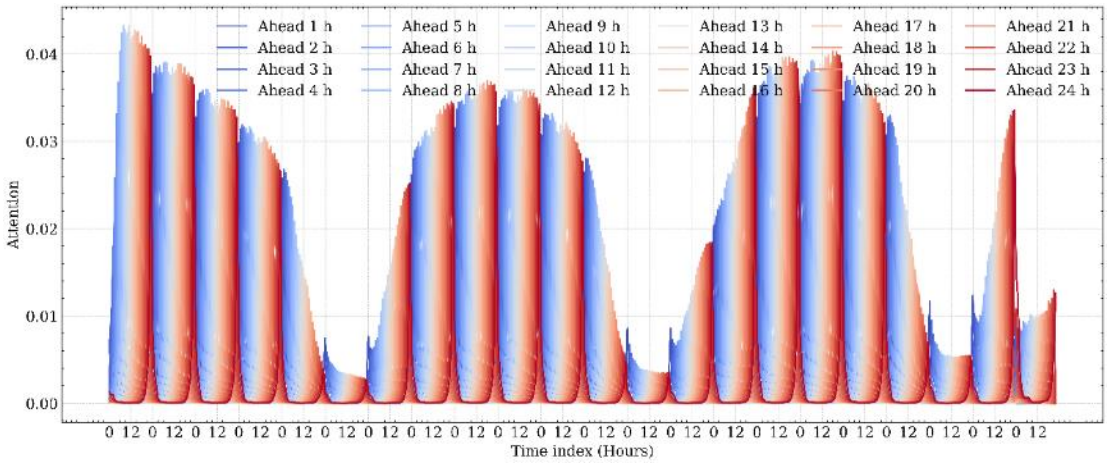


Figure 6-19 Attention weights to previous hours for predicting the next 24 hours

Figure 6-20 shows the attention weight of the 6 attention heads separately for predicting the first future hour. Each curve represents the attention weight of a single attention head. Although 6 attention heads were employed, their patterns were similar since the studied time series only had one seasonality (i.e. the daily pattern), while the weekly patterns can also be represented by different weights on each day of the week. However, different attention heads can capture the nuance in the same seasonality across different time series, which slightly helped increase the model performance. Theoretically, different attention heads can capture different seasonality in time series [127]. Further studies can shift to more complex multi-seasonality time series to test the capability of multi-head attentions.

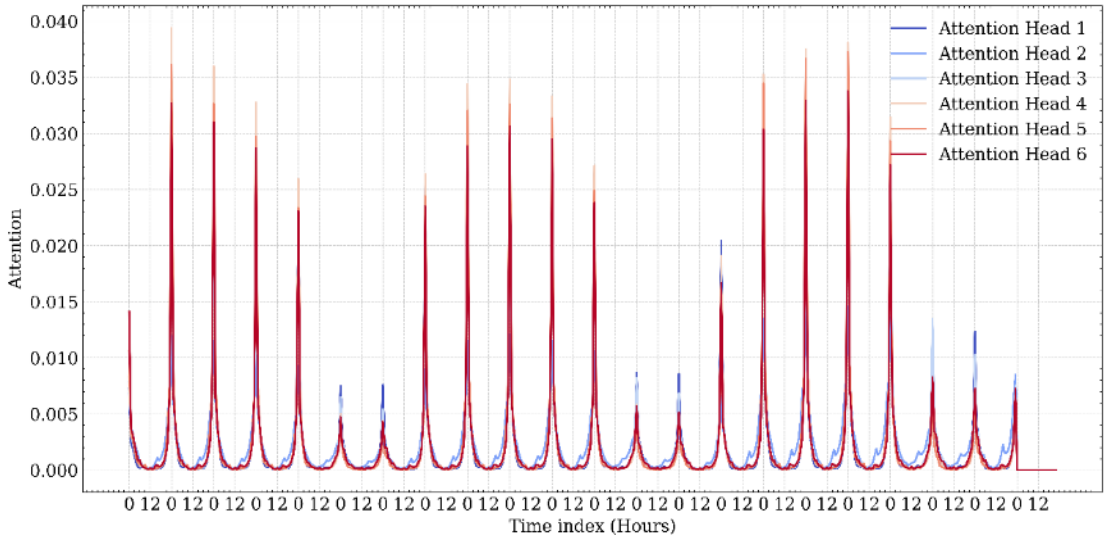


Figure 6-20 Attention weights to previous hours when predicting the first future hour across different attention heads

6.3 Location generation

6.3.1 Time&Activity-aware chain-based probability

After obtaining the prediction of activities in the next 24 hours, a probabilistic location generator is designed to assign specific locations based on predicted activities and departure time. The reason to use the probabilistic model instead of deep learning methods to predict final locations is threefold. First, the support of the set of locations (i.e., the POIs) is extremely large. 212,062 POIs have been recorded to be visited by observed devices in PG county at least one time during the two-month observation period (**Table *6-1***), which means if a classification model is built upon the data, the model should handle a multi-class problem with 212,062 classes. Second, the location visiting pattern for each individual is highly random and sparse. Each individual is characterized by a significant probability to return to a few highly frequented POIs [225], while a large fraction of POIs is explored by the individual

very few times, which will be easily ignored by deep learning models. Last, the cold start issue, i.e., the model cannot predict unobserved locations or devices, has been widely considered an issue for deep learning methods since they only learn patterns from observations. This section proposes a Time&Activity-aware chain-based probabilistic model to address these issues, which includes a global probability to describe the average attraction of each POI to all devices, and a local probability to represent the device-specific attraction between each POI and each device.

Global probability: Considering at hour *t* on day of week *w*, a person *k* intends to participate in activity *c*. Based on the discrete choice model, the person can choose from a set of POIs, I_c, that belongs to activity type *c*, while the choice probability is determined by its utility. Based on the gravity model, this study assumes the utility of choosing a POI is positively related to the time-aware popularity of the POI and negatively related to its distance to the person's home location:

$$U_{k,n,w,t} = v_{n,w,t}\exp(-\frac{d_{k,n}^2}{\sigma(d_{\forall I_c})^2}) \quad (6\text{-}32)$$

where $v_{n,w,t}$ is the total number of historical visits to POI *n* at hour *t* on day of week *w*; $d_{k,n}$ is the distance from the person *k*'s home location to POI *n*, $\sigma(d_{\forall I_c})$ is the st.d. of distance from the person *k*'s home location to all other POIs in I_c.

The global probability of choosing POI *n* is then calculated as the :

$$g_{k,n,w,t} = \frac{e^{U_{k,n,w,t}/\sigma(U_{\forall I_c})}}{\sum_{i\in I_c} e^{U_{k,i,w,t}/\sigma(U_{\forall I_c})}} \quad (6\text{-}33)$$

where $\sigma(U_{\forall I_c})$ is the st.d. of all utilities of POIs in I_c.

Local probability: Considering hour *t* on day of week *w*, and a person *k* intending to participate in activity *c*, the local probability of choosing POI *n* to finish

the activity is entirely inferred from the person's historical visiting patterns. The underlying assumption is people tend to revisit their most familiar places for the same activities, while the places may vary by the calendar factors:

$$l_{k,n,w,t} = \frac{v_{k,n,w,t}}{\sum_{i \in I_{c,k}} v_{k,i,w,t}} \qquad (6\text{-}34)$$

where $v_{k,n,w,t}$ is the total number of historical visits generated by person *k* to POI *n* at hour *t* on day of week *w*; $I_{c,k}$ is the set of POIs that has been visited by person *k* before the time he/she intends to participate in activity *c*.

One of the main differences between local and global probability is that the local probability considers the visit frequency of each POI at an individual level, while the global probability considers the global popularity of each POI for all individuals and adjusts it by individual distance. The final probability is calculated as a weighted sum of local and global probabilities:

$$p_{k,n,w,t} = \delta g_{k,n,w,t} + (1 - \delta) l_{k,n,w,t} \qquad (6\text{-}35)$$

where $\delta \in [0,1]$ ($\delta = 0.1$ in this study) is the weight to control the share of local and global probability. A higher δ indicates that more weight is assigned to the global probability.

The combination of local and global probability helps to solve the cold start issues. Assume that person *k* has never visited POI *n* before, then $l_{k,n,w,t} = 0$. However, since $g_{k,n,w,t} > 0$, the person would still have some chance to visit the POI *n*. In brief, this study uses the global probability to simulate the travelers' exploration behaviors and uses the local probability to capture the historical travel preference.

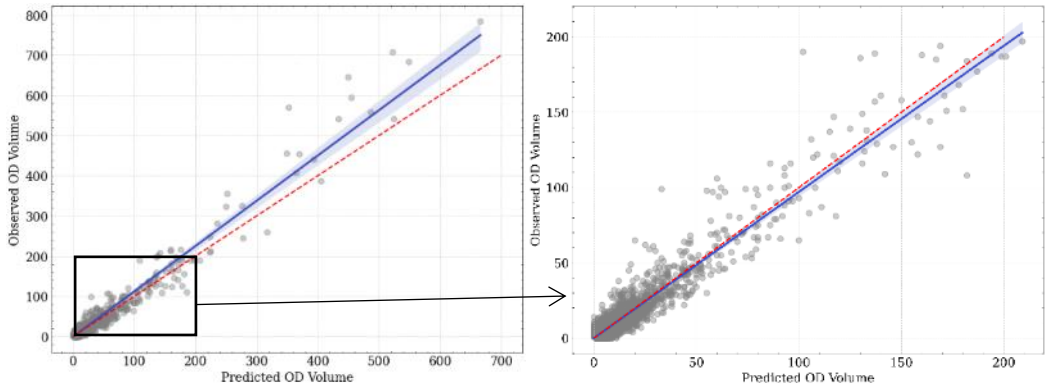
After computing the $p_{k,n,w,t}$ between each POI and each individual, 100 times bootstrap sampling is conducted to generate the POI for each predicted trip with a specific sampling probability, $p_{k,n,w,t}$. The departure time *t*, the type of activity *c*, and the number of trips are all predicted by the IHTF. The final POI is determined by the POI with the highest occurrence among the 100 bootstraps.

6.3.2 OD volume evaluation

The location generation outcomes are evaluated via a set of measures. The first measure is the difference between predicted and actual OD tables. An OD table is the most important input for traffic assignment. Accurately forecasting the OD table indicates the feasibility of modeling the travel demand in a bottom-up manner using the individual MDLD. Specifically, the predicted trip origins and destinations are spatially joined to the CBG and aggregated by OD pairs to finally count the predicted OD volume. The prediction is then compared with the real CBG-level OD volume to calculate measures like MAPE and R^2.

Figure 6-21 shows the scatter plot of the observed and predicted OD volume. Totally, 18,304 OD pairs are plotted (note that there are many trips with origins/destinations outside MG county). As shown, the points approximately lie on the identity line (the red dashed line), with an R^2 of 0.956. The model can accurately capture the small fraction of OD pairs with high volume (**Figure 6-21** (a)), and meanwhile, show acceptable accuracy in fitting those low-volume OD pairs (**Figure 6-21** (b)). Excluding OD pairs with volumes lower than 10, the MAPE reaches 24.74%. **Figure 6-22** further demonstrates the high consistency between the spatial distributions of observed and predicted OD flows. Most of the trips are intra-county

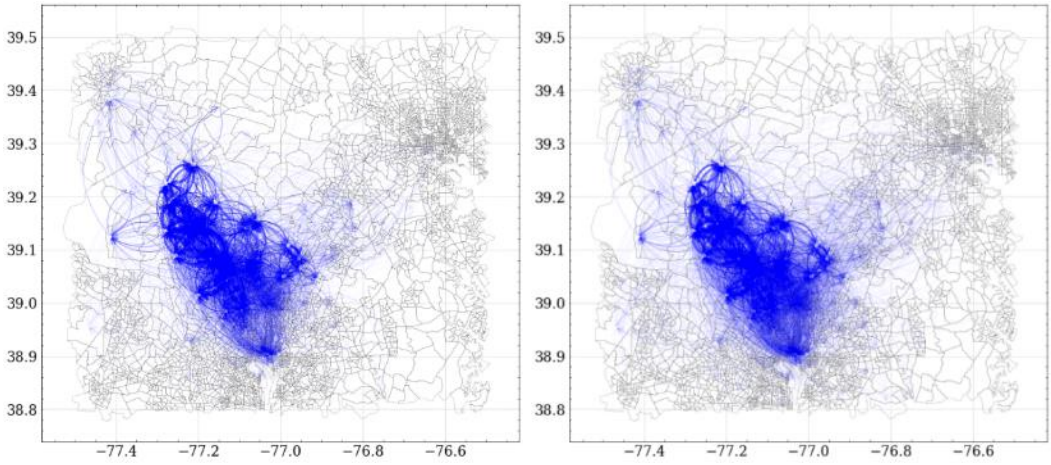
trips and inter-county trips flowing into/out of the D.C. area. Overall, at an aggregated CBG OD level, the method proposed in this study can achieve acceptable accuracy in terms of both the number of trips and the spatial distribution.



(a) (b)

Figure 6-21 Scatter plot of CBG-level OD volume: prediction vs. observation

Note: Panel (b) is the zoom-in view of panel (a) containing points with OD volumes smaller than 200.



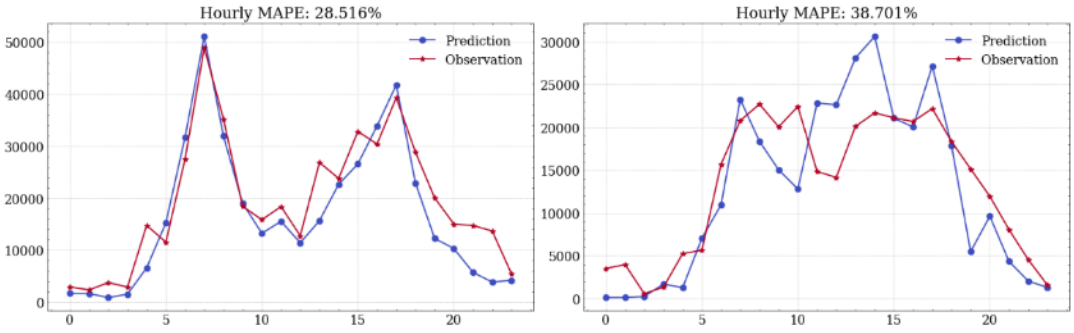
(b) (b)

Figure 6-22 Spatial plot of CBG-level OD flow: prediction (a) vs. observation (b)

6.3.3 Trip distance evaluation

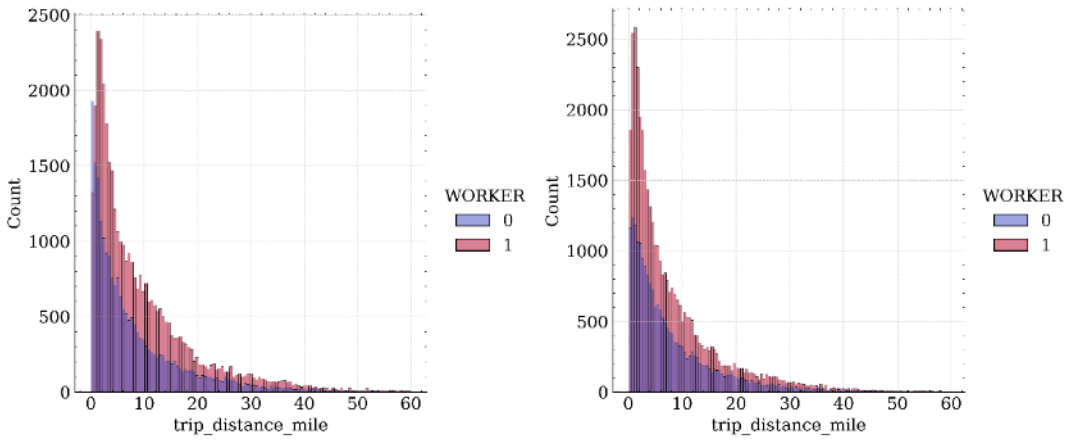
Another measure of the location generation outcomes is the trip distance. Since calculating the pairwise trip network distance for massive trips can be

computationally prohibitive, this study calculates the trip distance via the great circle distance between predicted trip origins and destinations, multiplied by an OD-specific average detour factor (workers: 1.46, nonworkers: 1.43) which is inferred from historical trips. **Figure 6-23** shows the predicted and observed hourly total miles traveled (i.e., the sum of trip distance in each hour) separated by workers and nonworkers. Similar to **Figure 6-16**, the accuracy of predicting workers' total miles traveled is significantly higher than that of nonworkers. However, the accuracy of predicting total miles traveled is overall lower than predicting trip counts (see **Figure 6-16**), which is plausible since the prediction of trip locations is built upon the prediction outcome of trip occurrence. The errors would be accumulated throughout the whole process from predicting trips to predicting locations of the trips.



(a) (b)

Figure 6-23 Predicted vs. observed hourly total miles traveled: workers (a) vs. nonworkers (b)



(a) (b)

Figure 6-24 Predicted (a) vs. observed (b) trip distance distribution

Figure 6-24 shows the distance distribution of predicted vs. observed trips (only trips with distances shorter than 60 miles are plotted). The two distributions present high consistency, indicating the location generator can produce trips with reasonable distance. **Table *6-10*** further shows the difference between the observed and predicted person miles traveled (PMT) as well as trip distances, separated by workers and nonworkers. Overall, the prediction shows high consistency with the observation, with a MAPE of 1.960-2.356% in the PMT and a MAPE of 5.179-11.024% in the trip distance.

Comparing **Table *6-10*** with **Table *6-9***, one interesting finding is that although the predicted trip rate of nonworkers is lower than the observation, the predicted PMT of nonworkers is even greater than the observation. One explanation is that trips predicted by the IHTF mostly belong to relatively long-distance trips. Trips with extremely short distances tend to be more random and thus are less likely to be well predicted. This can also be confirmed by the higher average trip distance

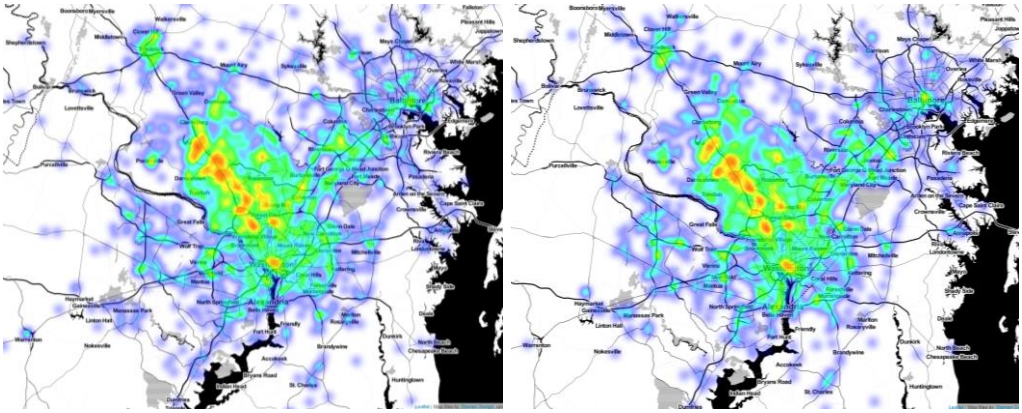
predicted by the model in **Table *6-10*.** Hence, although fewer trips are predicted, the corresponding PMT can remain at a comparable level with observation.

Table 6-10 Model performance in PMT and trip distance: workers vs. nonworkers

Person Miles Traveled (PMT), Daily			
	Prediction	Observation	MAPE
Workers (12,231)	30.150	29.570	1.960%
Nonworkers (6,696)	31.474	30.750	2.356%
Trip distance (Miles)			
	Prediction	Observation	MAPE
Workers (12,231)	9.044	8.146	11.024%
Nonworkers (6,696)	8.610	8.186	5.179%

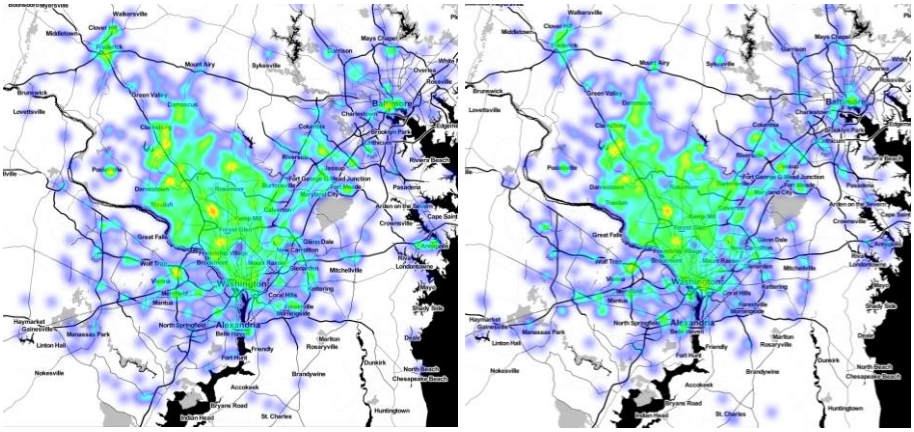
6.3.4 Trip spatial distribution

This study finally evaluates the location generation results via the similarity of the spatial distribution of the predicted and observed trips. **Figure 6-25** and **Figure 6-26** show the heatmap of predicted and observed trip origins in the testing dataset, separated by workers and nonworkers. A high consistency can already be observed between prediction and observation, either for workers or for nonworkers. The heatmaps of workers and nonworkers are also similar but with some nuances. For example, more hotspots located in the PG county can be observed in trips generated by workers, which may be due to the larger number of regular commuting trips.



(a) (b)

Figure 6-25 Heatmap for workers' trips: prediction (a) vs. observation (b)



(a) (b)

Figure 6-26 Heatmap for nonworkers' trips: prediction (a) vs. observation (b)

To evaluate the spatial similarity quantitively, the great circle distance between the predicted and observed trip starts/ends is computed. One issue to be addressed before computing the distance is to eliminate the difference in trip occurrence. For example, the IHTF may predict a trip occurs in a specific hour while the ground truth has none. Hence, for hours without trips occurring, the location of the device is defined as the trip destination of its last trip. After filling a location for each hour, the error is defined as the average distance between the predicted and

observed locations across all hours and devices. The error statistics are reported in **Table *6-11***. Overall, the distance between predicted and observed trip origins/destinations is small (the median of errors is 0.141-0.399 miles), indicating locations generated by the model are adjacent to actual locations. However, there are some trips with extremely large errors (>400 miles), which may be caused by some low-frequency air travels that are difficult to predict. These long-distance travels would largely increase the mean and st.d. of errors measured by distance. However, since these trips only account for a small fraction of total trips and since the prediction of these low-frequency travels is fairly difficult, the performance of the current model is still acceptable, especially considering the low median of errors.

Table 6-11 Statistics of the distance errors between predicted and observed trips

Distance between predicted and observed trip starts (Miles)				
	Mean	Median	Max	St.d.
Workers (12,231)	3.028	0.149	481.170	15.455
Nonworkers (6,696)	6.982	0.322	492.382	32.631
Distance between predicted and observed trip ends (Miles)				
	Mean	Median	Max	St.d.
Workers (12,231)	3.114	0.141	481.170	15.574
Nonworkers (6,696)	7.115	0.399	492.382	32.783

6.4 Discussion

This section proposes a novel framework, the hierarchical activity-based framework (HABF), for simultaneously predicting the activity, departure time, and location of the next trip for each observed resident living in PG county. Overall, the proposed framework achieves acceptable accuracy in jointly predicting different tasks. Upstream, the IHTF outperforms all baselines in time and activity prediction, yielding

an 82.86% F1 Score for a 31-class activity classification problem and a 32.15% F1 Score for classifying non-stationary activities. Downstream, the time&activity-aware location generator also shows the power of generating reliable locations for activities based on individualized probability varying by calendar factors. An R^2 of 0.956 between the predicted and observed CBG-level OD volume is achieved. In addition, the median distance between all predicted and observed trip starts/ends is 0.141-0.399 miles, implying a high similarity regarding their spatial distributions. Predicted trip itineraries can be safely fed into activity-oriented agent-based traffic simulators to support traffic management, network planning, and policy assessment.

Compared with aggregated population flow forecasting, individual trip itinerary forecasting is more challenging due to the high randomness, complex heterogeneity, and rich information in individual-level spatiotemporal travel patterns. Careful trip preprocessing, comprehensive variable fusion, and hierarchical methodological frameworks are required to solve the complex problem stepwise for generating multi-task multi-step outputs. Although extensive trip preprocessing has been conducted in this section, high randomness in activity and trip patterns still remain when zooming in on individuals (**Figure 6-7**). The possible reasons include missing observations, misidentified trips, mislabeled activities, and intrinsic randomness in human mobility patterns. Hence, although a comprehensive deep-learning framework is built upon processed trips, the classification accuracy for the minority classes remains relatively low, even with efforts in adjusting their weight in the loss function. Future research should consider restricting the selection of training

datasets, decreasing the number of activity types, and separately predicting workers and nonworkers, to reduce the forecasting difficulty.

The location generator built in this study is fully probabilistic without involving deep-learning models. The reason is that simple deep learning methods and limited computational resources may not be feasible in handling large imbalanced classification sets and addressing cold start issues. However, all these challenges could be addressed if more complex deep learning models are well incorporated into the current probabilistic framework. Higher accuracy can be envisioned through a parameter learning process to minimize the loss function, while the loss function can either be set as the cross-entropy loss when the prediction target is POI or the distance error when the target is explicit location coordinates.

7 Chapter 7: Conclusion

7.1 Summary of key findings

7.1.1 Extracting travel demand from MDLD

This study introduces a pipeline for parsing raw MDLD to distill useful travel information such as individual trip rosters and aggregated multi-modal OD matrices. The pipeline broadly follows home&work identification, trip identification, mode imputation, population weighting, and result validation. This study demonstrates the reliability of this pipeline by comparing the MDLD-based nationwide travel demand with a wide range of surveys and documenting high consistency. Last, through a real-world application that quantifies nationwide human mobility changes during the COVID-19 pandemic, this study further demonstrates the pipeline's feasibility in timely quantifying large-scale human travel patterns.

7.1.2 Revisiting travel demand and underlying factors

To comprehensively understand the relations between MDLD-based travel demand and external factors such as socioeconomics, demographics, and land use, this study fits a set of explainable machine learning (EML) models and interpreted them via novel interpretation techniques. Various nonlinearities, threshold effects, and interaction effects are uncovered in relations between travel demand and external factors. Moreover, the extensive comparison across EML models and interpretations provides empirical evidence of their pros and cons, as well as their diverse sensitivity to different hyperparameters and data attributes:

1) Overall, EML models exhibit high accuracy in estimating travel demand. LightGBM outperforms all models in this study and executes with high efficiency. However, the model fairness issue does exist. Models present higher MAPE in CBGs with much lower or higher sampling rates.

2) Among feature importance, the impurity importance is the most reliable since it allows feature dependency and it is computationally efficient. Measured by impurity importance, POI count, total population, CBG area, and # accommodations and food stores, are the 4 most important features.

3) Among relation visualization methods, PDP suffers from irregular perturbations and long leading/tailing plateaus due to its assumption of feature independence and its sensitivity to outliers. ALE plots help to address these issues. The SHAP interaction plot further enhances the interpretation reliability and informativeness by focusing on heterogeneous interaction effects.

4) The most important features are captured well by all models, even by a single tree. However, those less important features may vary across models and show less robustness. Meanwhile, feature importance tends to shift from the most important features to inconsequential features as the tree ensemble grows more complex.

7.1.3 Population flow time series forecasting

This study introduces a comprehensive GCN-based framework, the Multi-graph Multi-head Adaptive Temporal Graph Convolutional Network (Multi-ATGCN), for citywide population inflow forecasting considering complex spatiotemporal dependency and heterogeneous external effects. By incorporating a variety of deep learning techniques and spatiotemporal information, Multi-ATGCN demonstrates

strong flexibility, high efficiency, and superior performance in multi-step multivariable time series forecasting. The main findings are summarized as follows:

1) Overall, Multi-ATGCN achieves state-of-the-art results on two real-world datasets, outperforming all baselines over different horizons. Compared with the best baseline, Multi-ATGCN yields a 5.1-6.4% reduction in MAE for 24-step prediction, Such an improvement is even more salient in data-sparse zones and long-horizon scenarios that are more difficult to predict. Although with high accuracy and a large number of parameters, the training speed of Multi-ATGCN is comparable to many state-of-the-art models due to its non-recursion design in the decoder.

2) The ablation study further demonstrates the importance of different components in improving the model performance. The fully-connected output layer that uses all hidden states exhibits the greatest contribution to model performance. The effect of GCN is evident as well, indicating the importance of enabling the information flow among interdependent zones. Other components, such as the closeness and period temporal heads, the auxiliary information, and the zone-based normalization, all significantly enhance the model performance.

3) The types of adjacency matrices would significantly influence the model performance. The multi-view approach achieves the best performance, followed by functional similarity and OD-based measures. Although the two self-adaptive methods perform slightly worse compared to pre-defined methods, their performances are still remarkable even without any given prior knowledge. Finally, the distance matrix only performs slightly better than the identity matrix, which implies that the distance may not explicitly capture the real graph structure.

7.1.4 Individual trip itinerary forecasting

This study proposes a hierarchical activity-based framework (HABF) for simultaneously predicting the activity, departure time, and location of the next trip for each observed individual. An Interpretable Hierarchical Transformer (IHTF) is proposed to predict the hourly activity chain for each traveler, incorporating features from travelers, trips, and external environments. A location generator is then designed to generate locations based on predicted activity chains and historically visited places. The whole framework was trained and tested on a county-level dataset covering 2-month trips from over 18,000 devices. The main findings are summarized as follows:

1) Overall, the proposed framework achieves acceptable accuracy in jointly predicting different tasks. The IHTF outperforms all baselines in time and activity prediction, yielding an 82.86% F1 Score for a 31-class activity classification problem and a 32.15% F1 Score for classifying non-stationary activities. The location generator shows the power of generating reliable locations for activities. An R^2 of 0.956 between the predicted and observed CBG-level OD volume is achieved. In addition, the median distance between all predicted and observed trip starts/ends is 0.141-0.399 miles, implying a high similarity regarding their spatial distributions.

2) In the time and activity forecasting task, IHTF tends to be more "adventurous" in predicting non-stationary activities. This leads to a higher recall and F1 score but a relatively lower precision in predicting non-stationary classes. In addition, the model performance in predicting workers is significantly higher than that of nonworkers, manifesting in lower MAPE in hourly trip counts and higher precision, recall, and F1 score in predicting non-stationary activities for workers.

3) A well-designed loss function can significantly improve the model performance in handling imbalanced classification. Classical cross-entropy loss tends to label most of the activity as “Stationary”, leading to a much lower number of trips compared with observations. The weighted Focal Loss performs the best recall and F1 score across all loss functions in predicting non-stationary classes such as “Home->Work”, “Work->Home”, and “Others”, indicating it successfully helps the model to focus more on the minority classes that are more difficult to predict.

4) Although with a relatively simple design in location generation, the performance of the location generator is acceptable. The prediction shows high consistency with the observation, with a MAPE of 1.960-2.356% in the PMT and a MAPE of 5.179-11.024% in the trip distance. However, there are some trips with extremely large errors (>400 miles), which may be caused by some low-frequency air travels that are difficult to predict.

7.2 Future research directions

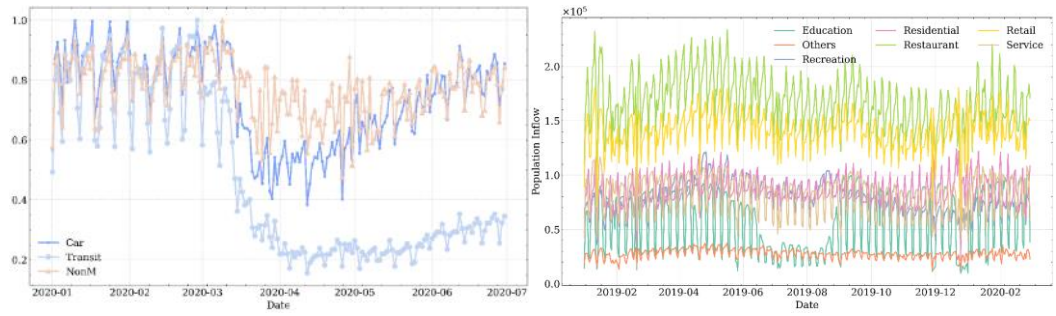
7.2.1 Augmenting model power in forecasting aggregated demand

Although extensive efforts have been made in this study in building comprehensive frameworks for multi-scale travel demand forecasting, limitations still exist and deserve further study. Follow-up research will expand the framework to support multi-task learning, OD forecasting, walk-forward validation, and transfer learning.

Multi-task learning: Multi-task learning is common in travel demand forecasting. For example, in four-step models, travel demand should be forecasted by travel modes; in activity-based models, travel demand should be forecasted by modes

and activities. **Figure 7-1** depicts travel demand time series by travel modes (a) and activities (b). One visible conclusion is that different types of travel demand may present very different time-varying patterns (see also **Figure 5-1** (d)). However, they also influence each other considering interactive mode choice behaviors and intertwined activity transition likelihood. Hence, expanding the current forecasting framework to multi-task learning is challenging but also worthwhile.

The Multi-ATGCN can serve as a reliable base framework for multi-task travel demand forecasting. However, some modules should be improved to address the new challenges. First, new adjacency matrices should be designed for each task since different tasks have different temporal patterns and spatial connections (**Figure 7-2**). The current Multi-ATGCN computes four adjacency matrices for each task. The increase in the number of tasks would easily lead to memory explosion, especially for large graphs. Hence, parameter sharing or graph partition strategies should be involved to mitigate the computational burden. Another challenge lies in the design of the loss function. An appropriate loss weighting strategy should be designed to achieve a globally best score while retaining acceptable accuracy for tasks that are more difficult to predict. Meanwhile, physical-guided learning can be involved to integrate some travel behavior theories into deep learning models. For example, the discrete choice model can be included in multi-modal forecasting, which would empower the model intelligence to respond reasonably to mode choice scenarios.



(a) (b)

Figure 7-1 Population flow time series by travel modes (a) and activities (b)

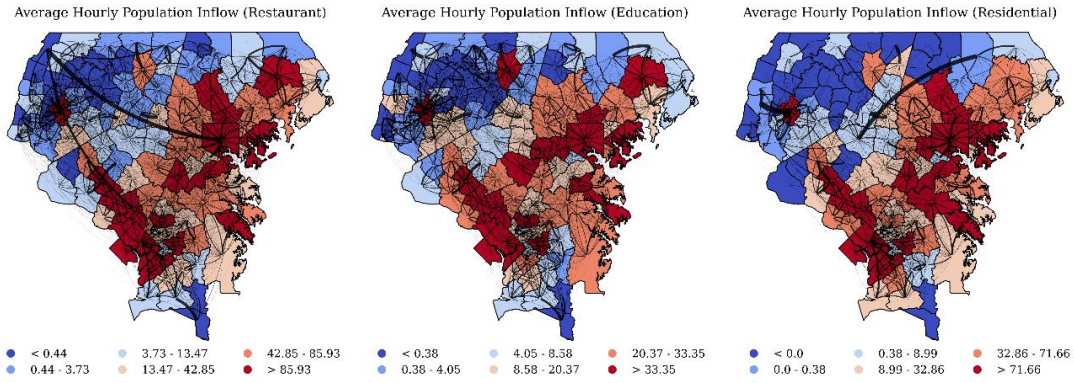


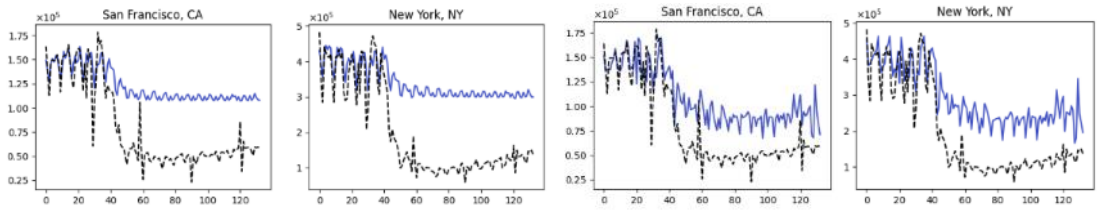
Figure 7-2 Spatial distribution of OD flow by activities

OD matrices forecasting: This study devotes considerable effort to population flow forecasting, while efforts on OD forecasting are limited. However, a powerful OD forecasting module is crucial to travel demand modeling since it generates the direct input for traffic assignment. OD forecasting is more challenging since it should jointly consider features of origins and destinations as well as their pairwise relations. Moreover, the high sparsity and the vast number of OD pairs, in particular in regions that contain thousands of TAZs, would be a huge challenge considering there are millions of OD pairs to be modeled. Another major problem that has not yet been well addressed is how to transform the line graph into a node graph since OD

forecasting is intended to predict edge-level time series while most of the current TGCN frameworks are designed to predict node-level signals.

To address the aforementioned challenges, some improvements should be made based on the current Multi-ATGCN framework. For example, methods that can convert line-graph-based OD matrices to node graphs should be employed [226]. Adjacency matrices should be redesigned to consider pairwise relations between origins and destinations. Meanwhile, the sparsity issue in OD matrices can be addressed by SVD or by focusing on OD pairs with relatively higher volumes.

Walk-forward validation: In real-time time series forecasting, new data become available continually, including both targets and external variables. If parameters of deep learning methods remain unchanged after training, predictions tend to become error-prone since the model does not get the knowledge from new data. The prediction outcome becomes even more inaccurate if some unexpected interventions happened and entirely changed the data distribution. A more realistic approach is to retrain the model with actual data as it becomes available for further predictions and validate it simultaneously, aka walk-forward validation. **Figure 7-3** illustrates the prediction outcomes with or without walk-forward validation under the shock of COVID-19. As shown, if walk-forward validation is not employed, the model would keep generating the prediction based on knowledge from pre-pandemic patterns, and thus substantially deviate from the actual value.



(a) (b)

Figure 7-3 Prediction outcomes without (a)/with (b) walk-forward validation

This study does not include post-pandemic data; hence, missing walk-forward validation does not lead to significant performance degeneration. However, in the real-world application, it is important to consider the effects of external events on travel demand, and incorporating the walk-forward validation into model training would be important. One main challenge is that the training of deep learning models is time-consuming. It is not practical to retrain the whole model using all data every time new data become available. One solution is to use a portion of the historical data mixed with the new data for model parameter updating. The update frequency should also be carefully selected. A high frequency would lead to a high request for computational sources while a low frequency would lead to a delayed response to sudden intervention. Future studies should consider the trade-off between efficiency and accuracy to find the most appropriate portion and update frequency.

Transfer learning: The metropolitan areas have a large number of mobile devices that can generate the MDLD, providing sufficient samples for forecasting model training. However, there exist lots of data-sparse zones which lack sufficient observations. An important future direction is to store knowledge gained while learning from data-sufficient zones and apply it to data-sparse zones, i.e., transfer learning. Transfer learning has several benefits, including saving training time, better performance of neural networks, and not needing a lot of data. This is especially valuable in large-scale population flow forecasting since we can train the model using a portion of regions and apply the model to the entire population.

7.2.2 Enhancing model accuracy in individual trip forecasting

Due to the high randomness of individual-level spatiotemporal patterns, the forecasting accuracy of individual trip itineraries is below that of aggregated models and still needs further enhancement. For time and activity forecasting, the IHTF has already comprehensively considered various external effects and complex temporal nonlinearities, outperforming all baselines particularly in forecasting the minority classes. Hence, to further increase the model accuracy, more efforts should be made to improve trip preprocessing. The current training dataset is too random with lots of noise. Hence, the overall goal of trip preprocessing is to construct a more regular dataset for model training. For example, spatiotemporal similarity can be measured for each device based on its daily trips, and different models can be trained separately for devices with different spatiotemporal similarities to increase the model's robustness and accuracy. In addition, considering the differences in spatiotemporal patterns between workers and nonworkers, as well as weekdays and weekends, further studies should consider building separate models for each scenario to decrease the data complexity. Last, individual characteristics included in the current IHTF are relatively simple. Further studies can include more information that can be inferred from ACS based on the device’s home location.

The location generator built in this study is fully probabilistic without involving the parameter optimization process. Higher accuracy can be envisioned if combing the probabilistic model with deep learning models and minimizing the loss function through a parameter learning process. The IHTF, or any other methods that can handle sequence forecasting such as RNN and TCN, can be used as the base

framework. The loss function can either be set as the (weighted) cross-entropy loss when the prediction target is an identity of POI or as the distance between predicted locations and observed locations when the target is explicit location coordinates.

7.2.3 Constructing an end-to-end MDLD-based travel demand model

A complete travel demand model includes multi-modal OD estimation, future scenario forecasting, and traffic simulation. This study separately focuses on parts of the travel demand model, for example, parsing MDLD to derive OD matrices, forecasting aggregated population flow, and forecasting individual trip itineraries; However, it is unclear how all these modules should be seamlessly integrated into a holistic entity to ultimately replicate travel demand models at both aggregated and disaggregated levels using MDLD. Hence, one important future direction is to systematically incorporate these cutting-edge techniques together and demonstrate the efficiency, reliability, and portability of this advanced ensemble.

To achieve a complete travel demand model, one important part that is missing in this study is the traffic simulation. MDLD has rich, high-resolution travel information, which is compatible with almost all scales of traffic simulators (**Figure 7-4**). Future research will mesh forecasted multi-resolution travel demand with large-scale traffic simulation tools to obtain citywide traffic flow parameters. For example, forecasted OD matrices can be fed into macro simulators, and individual trip itineraries can be fed into agent-based simulators, to finally generate road traffic measures such as speed and volume, which can be used in comparisons with field observations collected from road detectors for holistic validation.

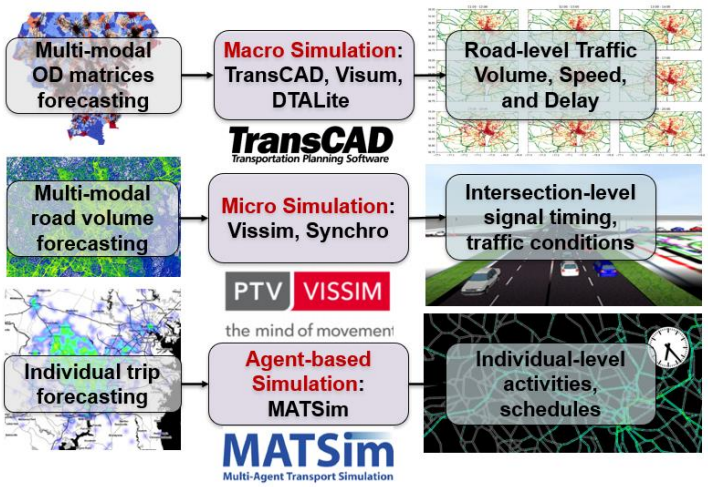


Figure 7-4 Conceptual framework of connecting MDLD with traffic simulators

The high forecasting accuracy and efficiency of deep learning models, along with continuously collected MDLD, enable modeling travel demand fully online. When deploying the MDLD-based travel demand model online, one main potential challenge is to meet the hourly update and execution frequency. The cloud computing services, such as Amazon Web Services (AWS) EMR and AWS Lambda, provide feasible solutions for guaranteeing computational efficiency. For example, all data storage, processing, and modeling can be finished on cloud services. The system will be scheduled hourly to fetch all related data and pass through the pre-trained deep learning models (Multi-ATGCN or HABF) for forecasting. Models will be trained and updated in a walk-validation manner by continuously including new data. Finally, outcomes should be fed into fast traffic simulator to generate citywide, road-level, time-dependent traffic volume and speed in the future.

References

1. Mitchell, R.B. and C. Rapkin, *Urban traffic*, in *Urban Traffic*. 1954, Columbia University Press.
2. Wardrop, J.G., *Road paper. some theoretical aspects of road traffic research.* Proceedings of the institution of civil engineers, 1952. **1**(3): p. 325-362.
3. Caceres, N., L. Romero, and F.G. Benitez, *Exploring strengths and weaknesses of mobility inference from mobile phone data vs. travel surveys.* Transportmetrica A: Transport Science, 2020. **16**(3): p. 574-601.
4. Milne, D. and D. Watling, *Big data and understanding change in the context of planning transport systems.* Journal of Transport Geography, 2019. **76**: p. 235-244.
5. Mladenovic, M. and A. Trifunovic, *The shortcomings of the conventional four step travel demand forecasting process.* Journal of Road and Traffic Engineering, 2014. **60**(1): p. 5-12.
6. Lee, R.J., I.N. Sener, and J.A. Mullins III, *An evaluation of emerging data collection technologies for travel demand modeling: from research to practice.* Transportation Letters, 2016. **8**(4): p. 181-193.
7. Chen, C., et al., *The promises of big data and small data for travel behavior (aka human mobility) analysis.* Transportation research part C: emerging technologies, 2016. **68**: p. 285-299.
8. Donnelly, R., *Advanced practices in travel forecasting*. Vol. 406. 2010: Transportation Research Board.
9. McNally, M.G. and C.R. Rindt, *The activity-based approach*, in *Handbook of transport modelling*. 2007, Emerald Group Publishing Limited.
10. Rasouli, S. and H. Timmermans, *Activity-based models of travel demand: promises, progress and prospects.* International Journal of Urban Sciences, 2014. **18**(1): p. 31-60.
11. Zhang, L., et al., *Feasibility and benefits of advanced four-step and activity-based travel demand models for Maryland*. 2011.
12. Wang, Z., S.Y. He, and Y. Leung, *Applying mobile phone data to travel behaviour research: A literature review.* Travel Behaviour and Society, 2018. **11**: p. 141-155.
13. Jiang, S., et al. *A review of urban computing for mobile phone traces: current methods, challenges and opportunities*. in *Proceedings of the 2nd ACM SIGKDD international workshop on Urban Computing*. 2013.
14. Nitsche, P., et al., *Supporting large-scale travel surveys with smartphones–A practical approach.* Transportation Research Part C: Emerging Technologies, 2014. **43**: p. 212-221.
15. Zhang, L., et al., *Interactive COVID-19 Mobility Impact and Social Distancing Analysis Platform.* Transportation Research Record, 2021: p. 03611981211043813.
16. Xiong, C., et al., *Mobile device data reveal the dynamics in a positive relationship between human mobility and COVID-19 infections.* Proceedings of the National Academy of Sciences, 2020. **117**(44): p. 27087-27089.

17. Alexander, L., et al., *Origin–destination trips by purpose and time of day inferred from mobile phone data.* Transportation research part c: emerging technologies, 2015. **58**: p. 240-250.
18. Iqbal, M.S., et al., *Development of origin–destination matrices using mobile phone call data.* Transportation Research Part C: Emerging Technologies, 2014. **40**: p. 63-74.
19. Wang, P., et al., *Understanding road usage patterns in urban areas.* Scientific reports, 2012. **2**(1): p. 1-6.
20. Çolak, S., et al., *Analyzing cell phone location data for urban travel: current methods, limitations, and opportunities.* Transportation Research Record, 2015. **2526**(1): p. 126-135.
21. Cervero, R. and K. Kockelman, *Travel demand and the 3Ds: Density, diversity, and design.* Transportation research part D: Transport and environment, 1997. **2**(3): p. 199-219.
22. Zheng, Y., et al., *Urban computing: concepts, methodologies, and applications.* ACM Transactions on Intelligent Systems and Technology (TIST), 2014. **5**(3): p. 1-55.
23. Barbosa, H., et al., *Human mobility: Models and applications.* Physics Reports, 2018. **734**: p. 1-74.
24. Yue, Y., et al., *Zooming into individuals to understand the aggregated: A review of trajectory-based travel behaviour studies.* Travel Behaviour and Society, 2014. **1**(2): p. 69-78.
25. Jiang, S., J. Ferreira, and M.C. Gonzalez, *Activity-based human mobility patterns inferred from mobile phone data: A case study of Singapore.* IEEE Transactions on Big Data, 2017. **3**(2): p. 208-219.
26. Sun, J., et al., *Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks.* IEEE Transactions on Knowledge and Data Engineering, 2020.
27. Lin, Z., et al. *Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.
28. Jin, W., et al. *Spatio-temporal recurrent convolutional networks for citywide short-term crowd flows prediction*. in *Proceedings of the 2nd International Conference on Compute and Data Analysis*. 2018.
29. Zhang, J., Y. Zheng, and D. Qi. *Deep spatio-temporal residual networks for citywide crowd flows prediction*. in *Thirty-first AAAI conference on artificial intelligence*. 2017.
30. Ke, J., et al., *Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network.* Transportation Research Part C: Emerging Technologies, 2021. **122**: p. 102858.
31. Shi, H., et al. *Predicting origin-destination flow via multi-perspective graph convolutional network*. in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 2020. IEEE.
32. Wang, Y., et al. *Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling*. in *Proceedings of the 25th*

ACM SIGKDD international conference on knowledge discovery & data mining. 2019.
33. Liu, Q., et al. *Predicting the next location: A recurrent model with spatial and temporal contexts*. in *Thirtieth AAAI conference on artificial intelligence*. 2016.
34. Yao, D., et al. *Serm: A recurrent model for next location prediction in semantic trajectories*. in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017.
35. Luca, M., et al., *A survey on deep learning for human mobility.* ACM Computing Surveys (CSUR), 2021. **55**(1): p. 1-44.
36. Toch, E., et al., *Analyzing large-scale human mobility data: a survey of machine learning methods and applications.* Knowledge and Information Systems, 2019. **58**(3): p. 501-523.
37. Ewing, R. and R. Cervero, *Travel and the built environment: A meta-analysis.* Journal of the American planning association, 2010. **76**(3): p. 265-294.
38. Ewing, R. and R. Cervero, *Travel and the built environment: a synthesis.* Transportation research record, 2001. **1780**(1): p. 87-114.
39. Handy, S., X. Cao, and P. Mokhtarian, *Correlation or causality between the built environment and travel behavior? Evidence from Northern California.* Transportation Research Part D: Transport and Environment, 2005. **10**(6): p. 427-444.
40. Wang, T., S. Hu, and Y. Jiang, *Predicting shared-car use and examining nonlinear effects using gradient boosting regression trees.* International Journal of Sustainable Transportation, 2021: p. 1-15.
41. Galster, G.C., *Nonlinear and threshold effects related to neighborhood: Implications for planning and policy.* Journal of Planning Literature, 2018. **33**(4): p. 492-508.
42. Zhang, W., et al., *Nonlinear effect of accessibility on car ownership in Beijing: Pedestrian-scale neighborhood planning.* Transportation research part D: transport and environment, 2020. **86**: p. 102445.
43. Jadon, S. *A survey of loss functions for semantic segmentation*. in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. 2020. IEEE.
44. Wu, R., et al., *Location prediction on trajectory data: A review.* Big data mining and analytics, 2018. **1**(2): p. 108-127.
45. McNally, M.G., *The four-step model*, in *Handbook of transport modelling*. 2007, Emerald Group Publishing Limited.
46. Johnston, R.A., *The urban transportation planning process.* The geography of urban transportation, 2004. **3**: p. 115-140.
47. de Dios Ortúzar, J. and L.G. Willumsen, *Modelling transport*. 2011: John wiley & sons.
48. Hensher, D.A. and K.J. Button, *Handbook of transport modelling*. 2000.
49. Cao, X.J., P.L. Mokhtarian, and S.L. Handy, *The relationship between the built environment and nonwork travel: A case study of Northern California.* Transportation Research Part A: Policy and Practice, 2009. **43**(5): p. 548-559.

50. Boarnet, M. and R. Crane, *The influence of land use on travel behavior: specification and estimation strategies.* Transportation Research Part A: Policy and Practice, 2001. **35**(9): p. 823-845.
51. Handy, S.L., et al., *How the built environment affects physical activity: views from urban planning.* American journal of preventive medicine, 2002. **23**(2): p. 64-73.
52. Cao, X., P.L. Mokhtarian, and S.L. Handy, *Do changes in neighborhood characteristics lead to changes in travel behavior? A structural equations modeling approach.* Transportation, 2007. **34**(5): p. 535-556.
53. Hu, S., et al., *Promoting carsharing attractiveness and efficiency: An exploratory analysis.* Transportation Research Part D: Transport and Environment, 2018. **65**: p. 229-243.
54. Hu, S., et al., *Examining spatiotemporal changing patterns of bike-sharing usage during COVID-19 pandemic.* Journal of transport geography, 2021. **91**: p. 102997.
55. Wang, T., S. Hu, and Y. Jiang, *Predicting shared-car use and examining nonlinear effects using gradient boosting regression trees.* International Journal of Sustainable Transportation, 2021. **15**(12): p. 893-907.
56. Hu, S., et al., *Modeling Usage Frequencies and Vehicle Preferences in a Large-Scale Electric Vehicle Sharing System.* IEEE Intelligent Transportation Systems Magazine, 2020.
57. Levin, M.W. and S.D. Boyles, *Effects of autonomous vehicle ownership on trip, mode, and route choice.* Transportation Research Record, 2015. **2493**(1): p. 29-38.
58. Zhang, Q., et al., *Mode choice between autonomous vehicles and manually-driven vehicles: An experimental study of information and reward.* Transportation Research Part A: Policy and Practice, 2022. **157**: p. 24-39.
59. Ding, C., et al., *Non-linear associations between zonal built environment attributes and transit commuting mode choice accounting for spatial heterogeneity.* Transportation Research Part A: Policy and Practice, 2021. **148**: p. 22-35.
60. Yang, J., J. Cao, and Y. Zhou, *Elaborating non-linear associations and synergies of subway access and land uses with urban vitality in Shenzhen.* Transportation Research Part A: Policy and Practice, 2021. **144**: p. 74-88.
61. Hu, S., et al., *Exploring the effect of battery capacity on electric vehicle sharing programs using a simulation approach.* Transportation Research Part D: Transport and Environment, 2019. **77**: p. 164-177.
62. Hu, S., et al., *Examining factors associated with bike-and-ride (BnR) activities around metro stations in large-scale dockless bikesharing systems.* Journal of Transport Geography, 2022. **98**: p. 103271.
63. Hu, S. and P. Chen, *Who left riding transit? Examining socioeconomic disparities in the impact of COVID-19 on ridership.* Transportation Research Part D: Transport and Environment, 2021. **90**: p. 102654.
64. Hu, S., et al., *A big-data driven approach to analyzing and modeling human mobility trend under non-pharmaceutical interventions during COVID-19*

pandemic. Transportation Research Part C: Emerging Technologies, 2021. **124**: p. 102955.
65. Chen, P., et al., *Estimating traffic volume for local streets with imbalanced data.* Transportation research record, 2019. **2673**(3): p. 598-610.
66. Zhu, L., et al., *Big data analytics in intelligent transportation systems: A survey*. IEEE Transactions on Intelligent Transportation Systems, 2018. **20**(1): p. 383-398.
67. Vij, A. and K. Shankari, *When is big data big enough? Implications of using GPS-based surveys for travel demand analysis.* Transportation Research Part C: Emerging Technologies, 2015. **56**: p. 446-462.
68. Wang, F. and C. Chen, *On data processing required to derive mobility patterns from passively-generated mobile phone data.* Transportation Research Part C: Emerging Technologies, 2018. **87**: p. 58-74.
69. Widhalm, P., et al., *Discovering urban activity patterns in cell phone data.* Transportation, 2015. **42**(4): p. 597-623.
70. Yang, M., et al., *A data-driven travel mode share estimation framework based on mobile device location data.* Transportation, 2021: p. 1-45.
71. Toole, J.L., et al., *The path most traveled: Travel demand estimation using big data resources.* Transportation Research Part C: Emerging Technologies, 2015. **58**: p. 162-177.
72. Jia, J.S., et al., *Population flow drives spatio-temporal distribution of COVID-19 in China.* Nature, 2020: p. 1-5.
73. Lenormand, M., et al., *Cross-checking different sources of mobility information.* PloS one, 2014. **9**(8): p. e105184.
74. Zhang, Y., et al. *Daily OD matrix estimation using cellular probe data*. in *89th Annual meeting transportation research board*. 2010.
75. Hariharan, R. and K. Toyama. *Project Lachesis: parsing and modeling location histories*. in *International Conference on Geographic Information Science*. 2004. Springer.
76. Zheng, Y. and X. Xie, *Learning travel recommendations from user-generated GPS traces.* ACM Transactions on Intelligent Systems and Technology (TIST), 2011. **2**(1): p. 1-29.
77. Yuan, J., Y. Zheng, and X. Xie. *Discovering regions of different functions in a city using human mobility and POIs*. in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012.
78. Huang, H., Y. Cheng, and R. Weibel, *Transport mode detection based on mobile phone network data: A systematic review.* Transportation Research Part C: Emerging Technologies, 2019. **101**: p. 297-312.
79. Wu, L., B. Yang, and P. Jing, *Travel mode detection based on GPS raw data collected by smartphones: a systematic review of the existing methodologies.* Information, 2016. **7**(4): p. 67.
80. Feng, T. and H.J. Timmermans, *Transportation mode recognition using GPS and accelerometer data.* Transportation Research Part C: Emerging Technologies, 2013. **37**: p. 118-130.

81. Li, L., et al., *Coupled application of generative adversarial networks and conventional neural networks for travel mode detection using GPS data.* Transportation Research Part A: Policy and Practice, 2020. **136**: p. 282-292.
82. Xiao, G., Z. Juan, and C. Zhang, *Travel mode detection based on GPS track data and Bayesian networks.* Computers, environment and urban systems, 2015. **54**: p. 14-22.
83. Wang, B., L. Gao, and Z. Juan, *Travel mode detection using GPS data and socioeconomic attributes based on a random forest classifier.* IEEE Transactions on Intelligent Transportation Systems, 2017. **19**(5): p. 1547-1558.
84. Sadeghian, P., J. Håkansson, and X. Zhao, *Review and evaluation of methods in transport mode detection based on GPS tracking data.* Journal of Traffic and Transportation Engineering (English Edition), 2021. **8**(4): p. 467-482.
85. Tedjopurnomo, D.A., et al., *A survey on modern deep neural network for traffic prediction: Trends, methods and challenges.* IEEE Transactions on Knowledge and Data Engineering, 2020.
86. Jiang, W. and J. Luo, *Graph neural network for traffic forecasting: A survey.* arXiv preprint arXiv:2101.11174, 2021.
87. Manibardo, E.L., I. Laña, and J. Del Ser, *Deep learning for road traffic forecasting: Does it make a difference?* IEEE Transactions on Intelligent Transportation Systems, 2021.
88. Van Wee, B. and S. Handy, *Key research themes on urban space, scale, and sustainable urban mobility.* International journal of sustainable transportation, 2016. **10**(1): p. 18-24.
89. Holtzclaw, J., et al., *Location efficiency: Neighborhood and socio-economic characteristics determine auto ownership and use-studies in Chicago, Los Angeles and San Francisco.* Transportation planning and technology, 2002. **25**(1): p. 1-27.
90. Zhang, Y., et al., *Exploring the impact of built environment factors on the use of public bikes at bike stations: Case study in Zhongshan, China.* Journal of transport geography, 2017. **58**: p. 59-70.
91. Wood, S.N., *Generalized additive models: an introduction with R*. 2006: chapman and hall/CRC.
92. Wali, B., et al., *Developing policy thresholds for objectively measured environmental features to support active travel.* Transportation research part D: transport and environment, 2021. **90**: p. 102678.
93. Tao, T. and J. Cao, *Exploring the interaction effect of poverty concentration and transit service on highway traffic during the COVID-19 lockdown.* Journal of Transport and Land Use, 2021. **14**(1): p. 1149-1164.
94. Friedman, J.H., *Greedy function approximation: a gradient boosting machine.* Annals of statistics, 2001: p. 1189-1232.
95. Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.
96. Ding, C., X.J. Cao, and P. Næss, *Applying gradient boosting decision trees to examine non-linear effects of the built environment on driving distance in Oslo.* Transportation Research Part A: Policy and Practice, 2018. **110**: p. 107-117.

97. Baumgarte, F., et al., *Revealing influences on carsharing users' trip distance in small urban areas.* Transportation Research Part D: Transport and Environment, 2022. **105**: p. 103252.
98. Cheng, L., et al., *Applying a random forest method approach to model travel mode choice behavior.* Travel behaviour and society, 2019. **14**: p. 1-10.
99. Yang, C., M. Chen, and Q. Yuan, *The application of XGBoost and SHAP to examining the factors in freight truck-related crashes: An exploratory analysis.* Accident Analysis & Prevention, 2021. **158**: p. 106153.
100. Shao, Q., et al., *Threshold and moderating effects of land use on metro ridership in Shenzhen: Implications for TOD planning.* Journal of Transport Geography, 2020. **89**: p. 102878.
101. Tao, T., et al., *Exploring the nonlinear relationship between the built environment and active travel in the twin cities.* Journal of Planning Education and Research, 2020: p. 0739456X20915765.
102. Cheng, L., et al., *Examining non-linear built environment effects on elderly's walking: A random forest approach.* Transportation research part D: transport and environment, 2020. **88**: p. 102552.
103. Molnar, C., G. Casalicchio, and B. Bischl. *Interpretable machine learning–a brief history, state-of-the-art and challenges*. in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2020. Springer.
104. Doshi-Velez, F. and B. Kim, *Towards a rigorous science of interpretable machine learning.* arXiv preprint arXiv:1702.08608, 2017.
105. Molnar, C., *Interpretable machine learning*. 2020: Lulu. com.
106. Fisher, A., C. Rudin, and F. Dominici, *All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously.* J. Mach. Learn. Res., 2019. **20**(177): p. 1-81.
107. Goldstein, A., et al., *Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation.* journal of Computational and Graphical Statistics, 2015. **24**(1): p. 44-65.
108. Ribeiro, M.T., S. Singh, and C. Guestrin. *" Why should i trust you?" Explaining the predictions of any classifier*. in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016.
109. Lundberg, S.M. and S.-I. Lee, *A unified approach to interpreting model predictions.* Advances in neural information processing systems, 2017. **30**.
110. Vlahogianni, E.I., M.G. Karlaftis, and J.C. Golias, *Short-term traffic forecasting: Where we are and where we're going.* Transportation Research Part C: Emerging Technologies, 2014. **43**: p. 3-19.
111. Karlaftis, M.G. and E.I. Vlahogianni, *Statistical methods versus neural networks in transportation research: Differences, similarities and some insights.* Transportation Research Part C: Emerging Technologies, 2011. **19**(3): p. 387-399.

112. Ye, J., et al., *How to build a graph-based deep learning architecture in traffic domain: A survey.* IEEE Transactions on Intelligent Transportation Systems, 2020.
113. Box, G.E., et al., *Time series analysis: forecasting and control*. 2015: John Wiley & Sons.
114. Lee, S. and D.B. Fambro, *Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting.* Transportation Research Record, 1999. **1678**(1): p. 179-188.
115. Williams, B.M. and L.A. Hoel, *Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results.* Journal of transportation engineering, 2003. **129**(6): p. 664-672.
116. Zhang, Y. and Y. Xie, *Forecasting of short-term freeway volume with v-support vector machines.* Transportation Research Record, 2007. **2024**(1): p. 92-99.
117. Cai, P., et al., *A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting.* Transportation Research Part C: Emerging Technologies, 2016. **62**: p. 21-34.
118. Wang, Y., et al., *Enhancing transportation systems via deep learning: A survey.* Transportation research part C: emerging technologies, 2019. **99**: p. 144-163.
119. Connor, J.T., R.D. Martin, and L.E. Atlas, *Recurrent neural networks and robust time series prediction.* IEEE transactions on neural networks, 1994. **5**(2): p. 240-254.
120. Cho, K., et al., *Learning phrase representations using RNN encoder-decoder for statistical machine translation.* arXiv preprint arXiv:1406.1078, 2014.
121. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
122. Li, Y. and H. Cao, *Prediction for tourism flow based on LSTM neural network.* Procedia Computer Science, 2018. **129**: p. 277-283.
123. Xu, C., J. Ji, and P. Liu, *The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets.* Transportation research part C: emerging technologies, 2018. **95**: p. 47-60.
124. Lin, Z., *Recurrent neural network models of human mobility*. 2018: University of California, Berkeley.
125. Yao, H., et al. *Deep multi-view spatial-temporal network for taxi demand prediction*. in *Proceedings of the AAAI conference on artificial intelligence*. 2018.
126. Bahdanau, D., K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate.* arXiv preprint arXiv:1409.0473, 2014.
127. Vaswani, A., et al. *Attention is all you need*. in *Advances in neural information processing systems*. 2017.
128. Dauphin, Y.N., et al. *Language modeling with gated convolutional networks*. in *International conference on machine learning*. 2017. PMLR.
129. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

130. Kalchbrenner, N., et al., *Neural machine translation in linear time.* arXiv preprint arXiv:1610.10099, 2016.
131. Van Den Oord, A., et al., *WaveNet: A generative model for raw audio.* SSW, 2016. **125**: p. 2.
132. Bai, S., J.Z. Kolter, and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.* arXiv preprint arXiv:1803.01271, 2018.
133. Yu, B., H. Yin, and Z. Zhu, *Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting.* arXiv preprint arXiv:1709.04875, 2017.
134. Wu, Z., et al., *Graph wavenet for deep spatial-temporal graph modeling.* arXiv preprint arXiv:1906.00121, 2019.
135. Wu, Z., et al. *Connecting the dots: Multivariate time series forecasting with graph neural networks*. in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020.
136. Cui, Z., et al., *Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting.* IEEE Transactions on Intelligent Transportation Systems, 2019. **21**(11): p. 4883-4894.
137. Du, B., et al., *Deep irregular convolutional residual LSTM for urban traffic passenger flows prediction.* IEEE Transactions on Intelligent Transportation Systems, 2019. **21**(3): p. 972-985.
138. Li, Y., et al., *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.* arXiv preprint arXiv:1707.01926, 2017.
139. Wu, Z., et al., *A comprehensive survey on graph neural networks.* IEEE transactions on neural networks and learning systems, 2020. **32**(1): p. 4-24.
140. Bruna, J., et al., *Spectral networks and locally connected networks on graphs.* arXiv preprint arXiv:1312.6203, 2013.
141. Henaff, M., J. Bruna, and Y. LeCun, *Deep convolutional networks on graph-structured data.* arXiv preprint arXiv:1506.05163, 2015.
142. Defferrard, M., X. Bresson, and P. Vandergheynst, *Convolutional neural networks on graphs with fast localized spectral filtering.* Advances in neural information processing systems, 2016. **29**.
143. Kipf, T.N. and M. Welling, *Semi-supervised classification with graph convolutional networks.* arXiv preprint arXiv:1609.02907, 2016.
144. Atwood, J. and D. Towsley, *Diffusion-convolutional neural networks.* Advances in neural information processing systems, 2016. **29**.
145. Gilmer, J., et al. *Neural message passing for quantum chemistry*. in *International conference on machine learning*. 2017. PMLR.
146. Hamilton, W., Z. Ying, and J. Leskovec, *Inductive representation learning on large graphs.* Advances in neural information processing systems, 2017. **30**.
147. Velickovic, P., et al., *Graph attention networks.* stat, 2017. **1050**: p. 20.
148. Liu, Z., et al. *Geniepath: Graph neural networks with adaptive receptive paths*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.

149. Geng, X., et al. *Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.
150. Yu, B., et al., *3d graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting.* arXiv preprint arXiv:1903.00919, 2019.
151. Li, R., et al. *Adaptive graph convolutional neural networks*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018.
152. Chai, D., L. Wang, and Q. Yang. *Bike flow prediction with multi-graph convolutional networks*. in *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*. 2018.
153. Wu, Y. and H. Tan, *Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework.* arXiv preprint arXiv:1612.01022, 2016.
154. Yu, R., et al. *Deep learning: A generic approach for extreme condition traffic forecasting*. in *Proceedings of the 2017 SIAM international Conference on Data Mining*. 2017. SIAM.
155. Cheng, X., et al. *Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting*. in *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018. IEEE.
156. Cui, Z., et al., *Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction.* arXiv preprint arXiv:1801.02143, 2018.
157. Wu, Y., et al., *A hybrid deep learning based traffic flow prediction method and its understanding.* Transportation Research Part C: Emerging Technologies, 2018. **90**: p. 166-180.
158. Lim, B., et al., *Temporal fusion transformers for interpretable multi-horizon time series forecasting.* International Journal of Forecasting, 2021. **37**(4): p. 1748-1764.
159. Chiang, W.-L., et al. *Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks*. in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019.
160. Mallick, T., et al., *Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting.* Transportation Research Record, 2020. **2674**(9): p. 473-488.
161. Gambs, S., M.-O. Killijian, and M.N. del Prado Cortez. *Next place prediction using mobility markov chains*. in *Proceedings of the first workshop on measurement, privacy, and mobility*. 2012.
162. Rendle, S., C. Freudenthaler, and L. Schmidt-Thieme. *Factorizing personalized markov chains for next-basket recommendation*. in *Proceedings of the 19th international conference on World wide web*. 2010.
163. Bahadori, M.T., Q.R. Yu, and Y. Liu, *Fast multivariate spatio-temporal analysis via low rank tensor learning.* Advances in neural information processing systems, 2014. **27**.

164. Cheng, C., et al. *Fused matrix factorization with geographical and social influence in location-based social networks*. in *Proceedings of the AAAI conference on artificial intelligence*. 2012.
165. Feng, J., et al. *Deepmove: Predicting human mobility with attentional recurrent networks*. in *Proceedings of the 2018 world wide web conference*. 2018.
166. Du, N., et al. *Recurrent marked temporal point processes: Embedding event history to vector*. in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016.
167. Chen, Y., et al. *Context-aware deep model for joint mobility and time prediction*. in *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020.
168. Lin, Z., et al., *Deep generative models of urban mobility.* IEEE Transactions on Intelligent Transportation Systems, 2017.
169. Yan, Z., et al., *Semantic trajectories: Mobility data computation and annotation.* ACM Transactions on Intelligent Systems and Technology (TIST), 2013. **4**(3): p. 1-38.
170. Ying, J.J.-C., W.-C. Lee, and V.S. Tseng, *Mining geographic-temporal-semantic patterns in trajectories for location prediction.* ACM Transactions on Intelligent Systems and Technology (TIST), 2014. **5**(1): p. 1-33.
171. Ying, J.J.-C., et al. *Semantic trajectory mining for location prediction*. in *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. 2011.
172. Chen, T. and C. Guestrin. *Xgboost: A scalable tree boosting system*. in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.
173. MTI. *Next Generation National Household Travel Survey (NHTS): National and Pooled Fund Origin Destination Data*. 2022.
174. NHTS. *NextGen NHTS National OD Data*. 2022.
175. Zhang, L., et al., *Interactive covid-19 mobility impact and social distancing analysis platform.* Transportation Research Record: p. 03611981211043813.
176. Wikipedia. *Geohash*. 0222.
177. Wang, F., et al., *Extracting trips from multi-sourced data for mobility pattern analysis: An app-based data example.* Transportation Research Part C: Emerging Technologies, 2019. **105**: p. 183-202.
178. Xiao, L., et al., *Nonlinear and synergistic effects of TOD on urban vibrancy: Applying local explanations for gradient boosting decision tree.* Sustainable Cities and Society, 2021. **72**: p. 103063.
179. Hu, S., et al., *Examining spatiotemporal evolution of racial/ethnic disparities in human mobility and COVID-19 health outcomes: Evidence from the contiguous United States.* Sustainable cities and society, 2022. **76**: p. 103506.
180. MIT. *MIT election data*. 2021 [cited 2022.
181. SafeGraph. *SafeGraph Data for Academics*. 2020 [cited 2021.
182. Apley, D.W. and J. Zhu, *Visualizing the effects of predictor variables in black box supervised learning models.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2020. **82**(4): p. 1059-1086.

183. Ke, G., et al., *Lightgbm: A highly efficient gradient boosting decision tree.* Advances in neural information processing systems, 2017. **30**: p. 3146-3154.
184. Prokhorenkova, L., et al., *CatBoost: unbiased boosting with categorical features.* Advances in neural information processing systems, 2018. **31**.
185. Friedman, J., T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent.* Journal of statistical software, 2010. **33**(1): p. 1.
186. Tibshirani, R., *Regression shrinkage and selection via the lasso.* Journal of the Royal Statistical Society: Series B (Methodological), 1996. **58**(1): p. 267-288.
187. Zou, H. and T. Hastie, *Regularization and variable selection via the elastic net.* Journal of the royal statistical society: series B (statistical methodology), 2005. **67**(2): p. 301-320.
188. Efron, B., et al., *Least angle regression.* The Annals of statistics, 2004. **32**(2): p. 407-499.
189. Breiman, L., et al., *Classification and regression trees*. 2017: Routledge.
190. Dietterich, T.G. *Ensemble methods in machine learning*. in *International workshop on multiple classifier systems*. 2000. Springer.
191. Geurts, P., D. Ernst, and L. Wehenkel, *Extremely randomized trees.* Machine learning, 2006. **63**(1): p. 3-42.
192. Louppe, G., *Understanding random forests: From theory to practice.* arXiv preprint arXiv:1407.7502, 2014.
193. Lundberg, S.M., G.G. Erion, and S.-I. Lee, *Consistent individualized feature attribution for tree ensembles.* arXiv preprint arXiv:1802.03888, 2018.
194. Lundberg, S.M., et al., *From local explanations to global understanding with explainable AI for trees.* Nature machine intelligence, 2020. **2**(1): p. 56-67.
195. Li, L., et al., *Hyperband: A novel bandit-based approach to hyperparameter optimization.* The Journal of Machine Learning Research, 2017. **18**(1): p. 6765-6816.
196. Pedregosa, F., et al., *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 2011. **12**: p. 2825-2830.
197. Ali, M., *PyCaret: An open source, low-code machine learning library in Python.* PyCaret version, 2020. **2**.
198. Akiba, T., et al. *Optuna: A next-generation hyperparameter optimization framework*. in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019.
199. Klaise, J., et al., *Alibi Explain: Algorithms for Explaining Machine Learning Models.* J. Mach. Learn. Res., 2021. **22**: p. 181:1-181:7.
200. Xie, Y., et al., *Fairness by "Where": A Statistically-Robust and Model-Agnostic Bi-Level Learning Framework.* 2022.
201. Bai, L., et al., *Adaptive graph convolutional recurrent network for traffic forecasting.* Advances in neural information processing systems, 2020. **33**: p. 17804-17815.
202. Hu, S. and C. Xiong, *High-Dimensional Population Flow Time Series Forecasting Via an Interpretable Hierarchical Transformer.* Available at SSRN 4049754, 2022.

203. Salinas, D., et al., *DeepAR: Probabilistic forecasting with autoregressive recurrent networks.* International Journal of Forecasting, 2020. **36**(3): p. 1181-1191.
204. Guo, S., et al. *Attention based spatial-temporal graph convolutional networks for traffic flow forecasting*. in *Proceedings of the AAAI conference on artificial intelligence*. 2019.
205. Li, Y., et al. *Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting*. in *International Conference on Learning Representations*. 2018.
206. Simonovsky, M. and N. Komodakis. *Dynamic edge-conditioned filters in convolutional neural networks on graphs*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
207. Karpathy, A. and L. Fei-Fei. *Deep visual-semantic alignments for generating image descriptions*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.
208. Cho, K., et al., *On the properties of neural machine translation: Encoder-decoder approaches.* arXiv preprint arXiv:1409.1259, 2014.
209. Zheng, C., et al. *Gman: A graph multi-attention network for traffic prediction*. in *Proceedings of the AAAI conference on artificial intelligence*. 2020.
210. Grover, A. and J. Leskovec. *node2vec: Scalable feature learning for networks*. in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016.
211. Fang, Z., et al. *Spatial-temporal graph ode networks for traffic flow forecasting*. in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021.
212. Choi, J., et al. *Graph neural controlled differential equations for traffic forecasting*. in *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022.
213. Li, L., et al., *A system for massively parallel hyperparameter tuning.* Proceedings of Machine Learning and Systems, 2020. **2**: p. 230-246.
214. Wang, J., et al. *Libcity: An open library for traffic prediction*. in *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 2021.
215. Ester, M., et al. *A density-based algorithm for discovering clusters in large spatial databases with noise*. in *kdd*. 1996.
216. Choi, E., et al., *Retain: An interpretable predictive model for healthcare using reverse time attention mechanism.* arXiv preprint arXiv:1608.05745, 2016.
217. Lim, B., et al., *Temporal fusion transformers for interpretable multi-horizon time series forecasting.* International Journal of Forecasting, 2021.
218. Li, S., et al., *Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting.* Advances in Neural Information Processing Systems, 2019. **32**: p. 5243-5253.
219. Guo, T., T. Lin, and N. Antulov-Fantulin. *Exploring interpretable lstm neural networks over multi-variable data*. in *International conference on machine learning*. 2019. PMLR.

220. Lin, T.-Y., et al. *Focal loss for dense object detection*. in *Proceedings of the IEEE international conference on computer vision*. 2017.
221. Sudre, C.H., et al., *Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations*, in *Deep learning in medical image analysis and multimodal learning for clinical decision support*. 2017, Springer. p. 240-248.
222. Salehi, S.S.M., D. Erdogmus, and A. Gholipour. *Tversky loss function for image segmentation using 3D fully convolutional deep networks*. in *International workshop on machine learning in medical imaging*. 2017. Springer.
223. Milletari, F., N. Navab, and S.-A. Ahmadi. *V-net: Fully convolutional neural networks for volumetric medical image segmentation*. in *2016 fourth international conference on 3D vision (3DV)*. 2016. Ieee.
224. Sutskever, I., O. Vinyals, and Q.V. Le. *Sequence to sequence learning with neural networks*. in *Advances in neural information processing systems*. 2014.
225. Gonzalez, M.C., C.A. Hidalgo, and A.-L. Barabasi, *Understanding individual human mobility patterns.* nature, 2008. **453**(7196): p. 779-782.
226. Makarov, N., *Development of a Deep Learning Surrogate for the Four-Step Transportation Model.* 2021.