DEEP GENERATIVE MODELS FOR TRAJECTORY PREDICTION AND MOBILITY NETWORK FORECASTING

AMIRHOSSEIN NADIRI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING & COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

June 2025

Abstract

Predicting human mobility is essential for urban planning, traffic management, and epidemiology. This thesis tackles two intertwined challenges: accurately forecasting individual trajectories and inferring the resulting mobility network. First, we introduce TrajLearn, a Transformer-based deep generative model that treats trajectories as token sequences and employs spatially constrained beam search to predict each individuals's next k locations with high precision. Building on these forecasts, we present MobiNetForecast, which constructs and predicts the future topology of the mobility network by detecting when independently predicted trajectories intersect in space and time. Across large, real-world datasets, our unified framework achieves up to 40% relative gains in trajectory accuracy and up to 100× improvement in contact prediction over state-of-the-art baselines. These results demonstrate that combining advanced sequence modeling with explicit contact inference offers a powerful, scalable solution for dynamic mobility network forecasting.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisor, Manos Papagelis, for his unwavering guidance and invaluable support throughout my research journey. His insightful feedback, expert advice, and genuine commitment to our success have profoundly influenced this work, and I am deeply grateful for the opportunity to learn from him.

I am forever indebted to my parents and my sister, whose constant love have been my anchor during the most challenging times. Their belief in me and their sacrifices have made this journey possible, and their encouragement has been a continual source of inspiration.

I am also grateful to my examination committee members, Aijun An and Mehdi Nourinejad, for taking the time to read my thesis and provide feedback.

Finally, I'd like to recognize the importance of mental-health awareness for everyone working in research and beyond. To those who sometimes struggle with focus, motivation, or simply finding balance: *your dedication matters, and you are not alone.* I'm deeply grateful to my friend, Yashar, for his steady support and to my supervisor, Manos, for his patience. Thank you for reminding me that curiosity and care for one's own well-being can go hand in hand.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advances in location tracking technologies have enabled the generation and collection of massive amounts of mobility data—data that capture information about the location and time at which a moving object (such as a pedestrian, vehicle, drone, or animal) is observed. Most commonly, these data are captured in the form of trajectories. A trajectory represents the path traversed by a moving object within a predefined spatial area (i.e., a map) and typically consists of a chronological sequence of geographic points $p_1, p_2, \ldots, p_n$. Each geographic point p_i is expressed as a triplet (x, y, t), where x and y denote the latitude and longitude coordinates respectively, and t denotes the time at which the location was recorded [1]. In this way, trajectories constitute spatiotemporal data that embed both spatial and temporal information about the movement of objects.

Recent advances in artificial intelligence (AI) and deep learning have enabled more rapid integration of disparate data sources, facilitating rapid risk assessments and providing critical guidance for policymakers during health emergencies [2]. More generally, the aggregation of individual-level mobility information from devices such as mobile phones has the potential to substantially enhance the precision and effectiveness of mathematical models for infectious

disease spread [3, 4]. However, challenges in data sharing and access often arise—not only for health data but also for data on individual behaviors (such as human mobility data) that are frequently owned by companies and can be of commercial value [5].

1.1 Problem of Interest

As multiple objects move through the same space concurrently, they frequently come into close proximity and interact. At any given time t, the aggregation of all pair-wise interactions forms a *contact network*. By observing and aggregating these contact networks over an extended period $[0, T]$, one can derive a *mobility network* that encapsulates the patterns of interactions among objects over time.

In this work, we study human mobility through the lens of network science by systematically analyzing the connections formed between individuals as they traverse different spatial and temporal contexts. Specifically, we construct a *mobility network* where nodes represent individuals and an edge is established between two individuals when they are physically close to each other. Figure 1.1 provides an illustrative example: as individuals move along their trajectories, their encounters at discrete time steps are captured as network snapshots, and the sequence of these snapshots forms the dynamic mobility network. Our central goal is to predict the future states of this mobility network—that is, to forecast the interactions that will occur over a future time period $[T + 1, T + k]$.

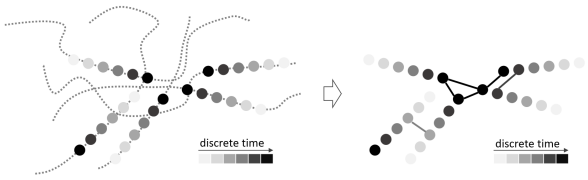


Figure 1.1: Illustrative example of the evolution of the system for five individuals over six discrete time steps: As individuals move along their trajectories (left), they come into contact with others in close proximity. The collection of these contacts at time t forms a proximity network. Over a time period $[0, T]$, the aggregation of such contacts results in a *mobility network* (right). Our goal is to predict the k future instances of the mobility network over the period $[T + 1, T + k]$.

1.2 The State of the Art and Limitations for Mobility Network Prediction

Various methods have been developed for learning and forecasting dynamic (or temporal) networks. Approaches based on temporal link prediction, including recurrent Graph Neural Networks (GNNs) and dynamic embedding methods [6, 7, 8, 9, 10], usually rely on historical interaction patterns to forecast future contacts. However, these models typically only capture the temporal evolution of historical links without adequately modeling the underlying mobility processes that generate these interactions.

Nevertheless, these models also exhibit limitations when applied to mobility networks:

- **Simplistic Treatment of Interactions:** Many temporal network models assume that future interactions can be predicted based solely on past link patterns, ignoring the underlying spatial movements that give rise to these interactions.

- **Lack of Contextual Information:** Most models do not incorporate rich contextual details such as the physical constraints of the urban environment or dynamic movement patterns, limiting their ability to accurately capture mobility nuances.

- **Overfitting and Limited Adaptability:** These models may overfit to the historical structure of the network and fail to adapt to rapid changes in mobility patterns, especially in rapidly changing urban contexts.

The common thread in these limitations is an over-reliance on historical interactions without sufficient modeling of the underlying mobility processes. However, mobility contact networks exhibit a fundamentally different characteristic: *first-time contacts dominate*. Two individuals who have never met before can quickly come into proximity—e.g., boarding the same bus, visiting the same retail store, or attending the same event. Because past link history is absent, dynamic link predictors struggle to assign a high probability to such *novel* edges. In our empirical analysis of three large-scale mobility datasets (Porto taxi, GeoLife and SFCO-3K), we observe that within a few future time steps, less than 5% of edges are *recurring* (previously observed), while over 90% of future edges are *new* (never seen in past windows). Figure 1.2 illustrates this phenomenon for H3@9 tessellation at 5-minute intervals: as we forecast $k = 1, \ldots, 10$ steps ahead, the fraction of recurring edges quickly drops below 5%, while the fraction of new edges remains above 90%.

Although these dynamic graph methods demonstrate good empirical performance in domains like social-media interaction, citation graphs, and user–item recommendation, *they typically rely on repeated patterns of edges*. In other words, the models learn from edges that appear at $t-1, t-2, \ldots$ to predict edges at t. If an edge (u, v) has never been observed in the past, the model cannot easily assign it a high score—unless it learns from some proxy (e.g. structural or attribute similarity) that u and v are likely to connect. In many real-world dynamic graphs, certain friendships or followings do reoccur. Such recurrence typically occurs in social-media interactions, email or messaging networks, citation graphs, and user–item recommendation systems, where existing or latent affinities drive repeated connections [11].

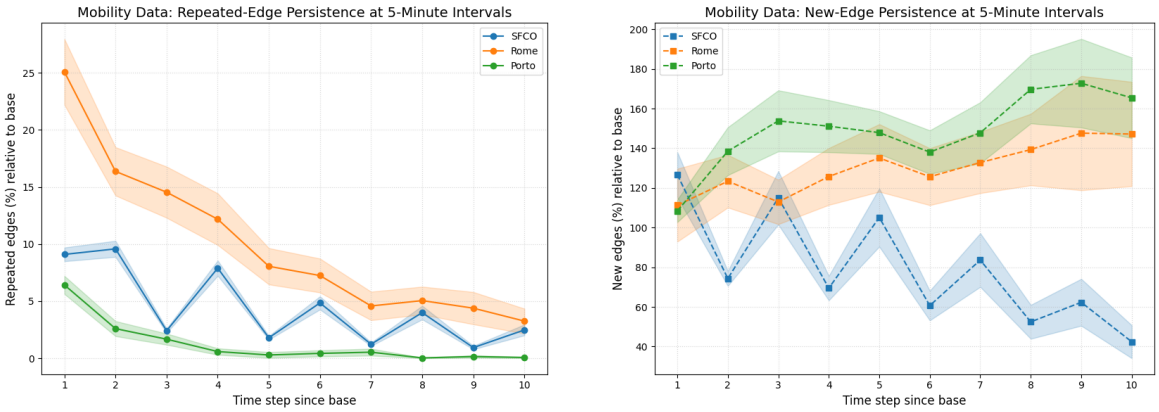


Figure 1.2: Edge persistence in mobility networks (H3@9, 5-minute intervals). **Left:** Fraction of future edges that are *recurring* (previously observed) as we step $k = 1, \ldots, 10$ windows ahead. **Right:** Fraction of *new* edges at each future window. Results are averaged over 100 random base snapshots (with 95% confidence intervals). Recurrence falls below 5% while new edges exceed 90%, highlighting the dominance of first-time contacts.

1.3 The State of the Art and Limitations for Trajectory Prediction

State-of-the-art models for trajectory prediction often rely on recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU). While these models have demonstrated impressive performance across a range of applications, traditional time-series forecasting methods, are ill-suited to our spatially-aware trajectory prediction task. Conventional approaches typically predict continuous scalar or vector values at fixed time intervals [12, 13], making them vulnerable to GPS noise and data sparsity when applied directly to raw mobility traces.

Moreover, standard forecasting models also lack inherent mechanisms to enforce spatial continuity or respect geographic constraints. Simply predicting future coordinate vectors fails to guarantee that a predicted path remains within connected regions of the road network or adheres to plausible movement patterns. Additionally, such models are prone to overfitting

on historical data, resulting in diminished generalizability when future trajectories deviate from past trends.

State-of-the-art trajectory prediction models, such as those based on recurrent neural networks (RNNs) (e.g., LSTM, GRU) and their variants, have achieved notable success. However, many of these approaches suffer from key limitations:

- **Fixed-Interval Continuous Predictions:** Off-the-shelf time-series methods forecast real-valued outputs at regular time steps and can lead to degraded performance under GPS noise and irregular sampling. [12, 13].
- **Insufficient Spatial Constraints:** Typical models ignore connectivity requirements between successive locations, allowing them to predict trajectories that violate real-world geographic or network constraints.
- **Overfitting to Historical Data:** Models may overfit on historical trajectories, leading to poor generalization on unseen data, especially when future movement patterns differ significantly from past trends.

1.4 The Proposed Approach

In our work, we propose a novel deep learning framework that diverges from conventional temporal network models by basing mobility network prediction on accurate trajectory forecasting. Our approach consists of two main steps:

(i) Trajectory Prediction: We leverage a Transformer-based model, TrajLearn, to predict the future trajectories of individuals from their current locations. By representing trajectories as sequences of blocks derived from a tessellated map, we

reduce data sparsity and capture the inherent spatial-temporal dynamics in a manner analogous to sequence prediction in natural language processing.

(i) Contact Inference and Network Construction: Instead of directly modeling the temporal evolution of interactions in a network, we infer contacts by identifying when two individuals, based on their independently forecasted trajectories, come into spatial proximity at the same time. These contacts are then aggregated over time to form a series of network snapshots that constitute the dynamic mobility network.

1.4.1 The Proposed Approach for Trajectory Prediction

Our trajectory prediction approach is centered on TrajLearn, a Transformer-based autoregressive model inspired by state-of-the-art natural language processing techniques (e.g., GPT [14]). In our framework:

- Each individual's trajectory is first converted from raw GPS coordinates into a sequence of spatiotemporal blocks using a higher-order mobility flow representation.
- The sequence of blocks is treated analogous to a sentence, where each block corresponds to a token.
- The Transformer model is then trained using teacher forcing to predict the next k blocks based on a given input sequence of length l.
- Constrained beam search is employed during inference to ensure that the predicted trajectories are spatially coherent, by restricting possible block transitions to immediate neighbors in the tessellation grid.

This prediction strategy allows us to forecast not only the future path of an individual but also the spatial context necessary for accurately inferring potential contacts between individuals.

1.4.2 The Proposed Approach for Mobility Network Prediction

Once individual trajectories are predicted, the next step is to utilize these predictions to detect contacts and construct the mobility network. Our methodology for mobility network prediction involves:

- **Temporal Mapping:** Each predicted trajectory is associated with discrete time steps. For each individual, the trajectory is represented as a sequence of block-time pairs (b, t) indicating which block is occupied at which time.

- **Contact Detection:** For any two individuals u and v, a contact is defined as occurring at time t if both are predicted to occupy the same block at that time. Mathematically, this is expressed as:

$$C_{uv}(t) = \begin{cases} 1, & \text{if } b_{u,t} = b_{v,t}, \\ 0, & \text{otherwise.} \end{cases}$$

- **Network Construction:** At each time step t, we construct a network snapshot $\mathcal{G}_t$ in which the nodes represent the objects present and the edges represent the contacts detected at that time. Over the entire prediction horizon $[T+1, T+k]$, the sequence of these snapshots forms the dynamic mobility network.

This trajectory-based approach to mobility network prediction has distinct advantages over traditional temporal network models. Rather than relying solely on historical interaction data, our method leverages physical movement prediction to infer contact formation, resulting in a more robust and context-aware model.

Mobility network prediction is an essential problem for understanding and managing real-world systems such as urban transportation[15], epidemic spread[2], and public safety[16]. By combining advanced trajectory prediction with robust contact detection and network construc-

tion, our approach provides a comprehensive solution that captures complex spatiotemporal interactions and dynamically evolving network structures. This research contributes to both the theoretical development and practical implementation of mobility forecasting systems and opens new avenues for future work in the domain.

1.5 Contributions

The major contributions of this thesis can be summarized as follows:

- **Contact Prediction Framework:** We propose MobiNetForecast, an integrated deep learning framework that bridges trajectory prediction and mobility network forecasting. Our framework departs from traditional temporal network models by incorporating explicit trajectory predictions to infer future contacts.

- **Transformer-Based Trajectory Prediction:** We develop TrajLearn, a decoder-only Transformer model that treats sequences of spatial blocks as tokens, leveraging state-of-the-art natural language processing methods for spatiotemporal sequence prediction.

- **Constrained Beam Search:** Our design includes a beam search module with spatial constraints, ensuring that the predicted trajectories are realistic and that the next predicted block is spatially adjacent. This module enhances the plausibility of contact predictions.

- **Rigorous Empirical Evaluation:** We conduct extensive experiments on multiple real-world mobility datasets, demonstrating that our proposed approach outperforms a range of baseline models in terms of prediction accuracy and robustness. Our analysis includes sensitivity studies and ablation experiments to underscore the effectiveness of individual components.

- **Hierarchical Mapping for Spatial Precision:** We introduce the concept of hierarchical maps, which employ mixed-resolution tessellation to refine spatial representations in high-density areas while remaining computationally efficient in low-density regions.

1.5.1 Published Work

This thesis is founded on the following published and submitted works:

- Nadiri, A., Li, J., Abuoda, G., & Papagelis, M. Mobility Network Forecasting: A Trajectory-based Contact Prediction Approach. *submitted*.
- Nadiri, A., Li, J., Faraji, A., Abuoda, G., & Papagelis, M. (2025, April). TrajLearn: Trajectory Prediction Learning using Deep Generative Models. *ACM Transactions on Spatial Algorithms and Systems*. [17]
- Faraji, A., Li, J., Alix, G., Alsaeed, M., Yanin, N., Nadiri, A., & Papagelis, M. (2023, December). Point2hex: Higher-order mobility flow data and resources. *In Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*. [18]

1.5.2 Reproducibility

In order to ensure reproducibility and adaptability, comprehensive access to source code and data has been publicly provided. Meticulous measures are undertaken to ensure that all prerequisites are explicitly detailed and accompanied by sequential instructions that facilitate the seamless setup and execution of the models. Furthermore, thoroughly documented model configurations are supplied for both the training and testing phases. In addition, clear and

detailed documentation of various configuration options is offered to enable the adaptation of the models for diverse scenarios and specialized applications.

MobiNetForecast GitHub repository:

- `https://github.com/amir-ni/MobiNetForecast`

TrajLearn GitHub repository:

- `https://github.com/amir-ni/Trajectory-prediction`

1.6 Thesis Organization

The remainder of this thesis is organized into seven chapters followed by appendices that address broader impacts and ethical implications. Chapter 2 presents a comprehensive literature review that covers spatiotemporal data mining, traditional and deep learning approaches for trajectory prediction, dynamic link prediction in temporal networks, and deep generative models. In Chapter 3, we lay the theoretical groundwork by introducing preliminary definitions and formally stating the research problems of trajectory prediction and mobility network forecasting. Chapter 4 then focuses on mobility data representation, where we discuss the challenges inherent in raw mobility data and explain how transforming trajectories into higher-order representations via block-based tessellation can mitigate sparsity while enhancing spatial coherence. Building on these foundations, Chapter 5 details our proposed methodology for trajectory prediction learning through the development of the TrajLearn model, including its Transformer-based architecture, training strategies, constrained beam search mechanism, and computational complexity analysis, and development of the MobiNetForecast, which uses the predicted trajectories to detect contacts and construct dynamic mobility networks. Chapter 6 shifts to the experimental evaluation of our approach, presenting detailed analyses and comparisons for both trajectory prediction and

mobility network forecasting components using diverse datasets and evaluation metrics. In Chapter 7, we introduce additional enhancements to our framework, such as methods for mapping predicted spatial block sequences back to GPS points and the use of hierarchical maps that adjust spatial resolution based on local data density. Finally, Chapter 8 concludes the thesis by summarizing our key findings, discussing the limitations of our current work, and outlining promising directions for future research.

Chapter 2

Literature Review

This chapter presents a comprehensive review of the literature related to mobility network forecasting, with a focus on trajectory prediction methods and the subsequent construction of mobility networks based on predicted contacts. The review is organized into four major sections. First, we discuss the broad field of spatiotemporal data mining, which lays the groundwork for understanding how massive trajectory and contact datasets are processed and analyzed. Next, we focus on trajectory prediction, detailing both traditional statistical approaches and advanced deep learning techniques. In this context, we review methods based on mobile data analysis as well as those employing computer vision. The following section covers link prediction in dynamic networks, where we describe both continuous-time and discrete-time frameworks that enable modeling of evolving interaction patterns. Finally, we delve into deep learning and deep generative models, which have emerged as powerful tools for modeling complex sequential and spatial phenomena. This literature review sets the stage for our research on forecasting mobility networks by predicting individual movements and inferring contact networks.

2.1 Spatiotemporal Data Mining

Spatiotemporal data mining (STDM) is the process of extracting useful patterns, trends, and knowledge from data that vary over both space and time. This field has experienced rapid growth [19], due to the proliferation of urban sensing technologies, mobile computing, and pervasive sensor networks, all of which produce massive amounts of trajectory and contact data [16, 15, 20, 21]. Early studies in STDM predominantly focused on statistical models and classical data mining tasks [1] such as clustering [22], classification [23], frequent pattern mining [24], and anomaly detection [25].

2.1.1 Statistical and Classical Approaches

In the early 2000s, researchers employed statistical methods to mine spatiotemporal patterns. Clustering techniques were extensively used to identify groups of trajectories exhibiting similar movement characteristics. Lee et al. [26] introduced a partition-and-group-based framework that divides trajectories into line segments and clusters similar segments, thereby capturing underlying movement patterns. Yan et al. [27] extended this concept by integrating both geometric and semantic features, which allowed the model to encapsulate the spatial configuration and contextual meaning of mobility data. Other studies focused on road-network-based clustering where the inherent constraints of transportation networks are exploited to group trajectories based on shared road segments or traffic densities [28, 29].

Classification tasks in STDM have also been a significant research focus. For example, Zheng et al. [23] applied decision tree models to classify different transportation modes using GPS data, while Bolbol et al. [30] demonstrated the use of Support Vector Machines (SVMs) for similar classification tasks. These early methods often relied on hand-engineered features and assumed that movement patterns could be effectively captured using conventional

statistical measures.

Anomaly detection, another cornerstone of classical STDM, involves identifying trajectories or sub-trajectories that deviate significantly from the norm. Lee et al. [25] proposed a Partition-and-Detect framework, which first segments trajectories and then applies distance-based and density-based measures to detect outliers. Such techniques were particularly useful for applications like traffic accident risk analysis and urban safety monitoring [31].

2.1.2 Deep Learning Approaches

The advent of deep learning has dramatically transformed the field of spatiotemporal data mining. In recent years, deep neural architectures have been developed to automatically learn feature representations from raw data, thereby reducing the need for extensive feature engineering[32, 33]. Wang et al. [34] provide an extensive review of deep learning approaches in STDM, highlighting the shift from conventional statistical methods to neural networks that integrate spatial and temporal modeling.

Spatiotemporal Graph Neural Networks (STGNNs) are a prime example of this evolution. By representing urban networks or transportation systems as graphs and integrating temporal dynamics, STGNNs have enabled the modeling of complex relationships in large-scale urban data [35]. These methods combine the advantages of graph convolutional networks (GCNs) with recurrent neural networks (RNNs) or temporal convolutional layers to capture long-range dependencies. As a result, STGNNs have found applications in traffic forecasting, crowd flow prediction, and urban planning [36].

The integration of deep learning techniques has not only improved predictive accuracy but also enhanced the scalability of STDM methods. Techniques such as deep autoencoders [37] and adversarial networks [38] have been employed to learn robust representations of complex

spatiotemporal patterns. These advancements have laid the foundation for subsequent developments in trajectory prediction and mobility network forecasting.

2.2 Trajectory Prediction

Trajectory prediction is the task of forecasting the future movement of an object based on historical mobility data. Accurate trajectory prediction is crucial for a wide range of applications, including autonomous driving, urban planning, epidemic modeling, and, notably, mobility network forecasting. In the context of this thesis, predicting the next steps of individuals is a key precursor to constructing dynamic mobility networks based on predicted contacts.

2.2.1 Traditional Statistical Methods

Traditional trajectory prediction methods have largely been based on statistical models that leverage historical movement patterns. One of the earliest and most widely used techniques is the Markov model. Markov chains assume that the future state of an object depends solely on its current state. Ashbrook and Starner [39] and Song et al. [40] demonstrated that Markov-based predictors could effectively model user movement by estimating transition probabilities between locations. However, the inherent assumption of the Markov property—that the future is independent of the past given the present—limits the model's ability to capture long-term dependencies and periodic behaviors in human mobility.

Beyond Markov models, pattern-based methods have also been explored. Techniques based on frequent sequence mining, periodic pattern mining, and dynamic pattern modeling (such as the T-pattern model proposed by Monreale et al. [41]) extract recurring spatiotemporal patterns from trajectories. These methods often integrate contextual factors such as social

interactions and user preferences [42, 43], but they are generally constrained by their reliance on predefined rules and patterns.

Matrix factorization techniques have been applied to trajectory prediction as well. Approaches based on matrix factorization (e.g., Cheng et al. [44], Li et al. [45], Lian et al. [46]) decompose trajectory data into latent factors that capture underlying movement patterns. While these methods provide useful insights, they typically struggle with the sparsity and high dimensionality of mobility data.

2.2.2 Deep Learning Approaches

The limitations of traditional statistical methods have paved the way for the adoption of deep learning techniques in trajectory prediction. Deep learning models have the capacity to learn complex, non-linear representations from large-scale data, thereby capturing intricate spatial and temporal dependencies that are often present in human mobility.

RNN-based Models

Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs), have been widely applied to trajectory prediction tasks. Liu et al. [47] proposed ST-RNN, which extends traditional RNNs by incorporating time-specific and spatial-specific transition matrices to capture the temporal and spatial nuances of trajectory data. Similarly, Feng et al. [48] introduced DeepMove, an attentional recurrent network that leverages both the sequential order and long-term dependencies present in sparse trajectory datasets.

Other RNN-based approaches have been specifically designed to handle the sparsity and irregularity of trajectory data. For instance, Lian et al. [49] (GeoSAN), Yang et al. [50]

(Flashback), and Luo et al. [51] (STAN) proposed architectures that incorporate additional contextual information, such as historical user behaviors and periodicity, to enhance the prediction of the next point of interest (POI).

CNN-based Models

In recent years, Convolutional Neural Networks (CNNs) have also been employed for trajectory prediction. Unlike RNNs, CNNs offer the advantage of parallel processing and stable gradient flows, which can be beneficial when processing long sequences. Gao et al. [52] proposed a framework called VANext that uses CNNs to capture historical moving patterns. Their model applies an attention mechanism to identify the most similar historical trajectory, which is then used for predicting the next POI.

Miao et al. [53] developed an Attentive Convolutional Network (ACN) that treats embedded trajectories as images. Special convolutional filters are designed to capture high-order sequential patterns across lengthy trajectories, thereby overcoming the limitations imposed by sparse and noisy data. Additionally, Dang et al. [54] introduced GCDAN, a model that employs graph convolution and a dual-attention mechanism to mitigate the sparsity and inaccuracies inherent in trajectory data. These CNN-based approaches have demonstrated promising results, particularly in applications where spatial context is paramount.

Hybrid and Advanced Methods

To further improve trajectory prediction, researchers have proposed hybrid models that combine the strengths of various approaches. For instance, some studies integrate matrix factorization with deep neural networks to capture both latent factors and non-linear dynamics in trajectory data [55, 56, 57]. Other methods, such as the pretrained-based contrastive learning network (PreCLN) developed by Yan et al. [58], leverage auxiliary pre-training tasks

and user embeddings to refine predictions.

Moreover, deep learning methods have been extended to address specialized prediction tasks, such as predicting vehicle trajectories in response to rare events [59] or forecasting afternoon travel based on morning trajectories [60]. Amichi et al. [61] proposed a two-step prediction process: first predicting the purpose of visiting a location and then forecasting the next location, thus integrating semantic understanding with spatial prediction. Also, in the context of social sciences, models have been proposed to offer insights into human crowd behavior and group patterns of trajectories [62, 63, 64].

2.2.3 Application to Mobility Network Forecasting

The central focus of this thesis is on mobility network forecasting, which utilizes trajectory prediction to determine the next steps of individuals within a network. Once the future trajectories are predicted, the subsequent contacts among individuals are inferred, and a dynamic mobility network is constructed. This approach not only aids in predicting traffic flows and urban dynamics but also provides critical insights for applications such as epidemic modeling and transportation planning. In this context, the integration of deep learning methods—both RNN- and CNN-based models—plays a pivotal role in accurately forecasting individual movements, which form the basis for the evolving mobility network.

2.3 Link Prediction in Dynamic Networks

Link prediction is a fundamental problem in network analysis, involving the inference of missing or future connections between nodes. In the context of mobility network forecasting, link prediction is used to anticipate interactions between individuals based on their predicted trajectories. This section reviews the literature on dynamic network models, focusing on both

continuous-time and discrete-time frameworks.

2.3.1 Continuous-Time Dynamic Networks

Continuous-time dynamic networks (CTDNs) provide a natural framework for modeling interactions that occur at arbitrary time points. Unlike discrete-time models that rely on fixed time intervals, CTDNs capture the fine-grained temporal evolution of networks by considering timestamped events.

One of the pioneering approaches in this domain is the Continuous-Time Dynamic Network Embedding (CTDNE) model [65], which incorporates the timing of interactions into the node embedding process. By accounting for the precise moments at which links are formed, CTDNE can effectively capture the temporal dynamics of the network. Hierarchical Temporal Network Embedding (HTNE) [66] extends this idea by introducing a hierarchical structure that models multi-scale temporal dependencies, thereby enhancing the ability to predict both short-term and long-term interactions.

Temporal Graph Attention Networks (TGAT) [67] represent another significant advancement in modeling CTDNs. TGAT extends the graph attention mechanism to temporal graphs by applying self-attention to temporal edges. This approach dynamically weighs past interactions based on their temporal proximity to the prediction time, which is particularly useful in networks where recent interactions are more indicative of future behavior. Similarly, DyRep [68] leverages a continuous-time framework based on point processes (specifically Hawkes processes [69]) to model the excitation effects of past interactions, thereby capturing the self-reinforcing nature of link formation.

Other methods such as the Continuous-time Attention Walk (CAW) [70] and Temporal Random Walk Embedding for Dynamic Networks (TREND) [71] employ random walk

strategies guided by temporal attention mechanisms. These approaches learn node embeddings by simulating temporal walks that traverse the network, effectively capturing complex temporal dependencies and evolving network structures.

2.3.2 Discrete-Time Dynamic Networks

Discrete-time dynamic networks (DTDNs) simplify the analysis of network evolution by partitioning time into fixed intervals. In this framework, the network is represented as a sequence of static snapshots, each corresponding to a particular time period. This approach facilitates the application of traditional graph learning methods, but it may lose some of the granularity inherent in continuous interactions.

Early work on DTDNs focused on neighborhood similarity measures such as the Jaccard coefficient, Adamic-Adar index, and Preferential Attachment, which quantify the likelihood of link formation based on shared neighbors [72]. Graph summarization techniques were also employed to compress network information into lower-dimensional representations while preserving key structural properties [73].

Matrix factorization methods have been widely used for link prediction in DTDNs. Techniques such as Temporal Matrix Factorization (TMF) [74] and Collaborative Regularized Joint Matrix Factorization (CRJMF) [75] decompose the adjacency matrices of network snapshots into latent factors. These factors capture evolving patterns of node interactions over time, enabling the reconstruction of future network states. Extensions of these approaches, including Temporal Latent Space Inference (TLSI) [76], Multi-Level Joint Feature Embedding (MLjFE) [77], and Graph Regularized Non-negative Matrix Factorization (GrNMF) [78], incorporate additional constraints such as temporal smoothness and high-order proximity to further improve prediction accuracy.

Deep learning models have also been applied to DTDNs. Approaches like dyngraph2vec [79] and Deep Dynamic Network Embedding (DDNE) [80] combine RNNs with GCNs to jointly capture the temporal evolution and structural properties of the network. Moreover, models such as EvolveGCN [81] and Graph Convolutional Network with Generative Adversarial Network (GCN-GAN) [82] extend GCNs to dynamic settings by incorporating adversarial training and attention mechanisms.

In the specific context of mobility network forecasting, DTDN models are employed to infer the evolution of contact networks. By aggregating predicted interactions over discrete time intervals, these models can generate dynamic mobility networks that capture the temporal evolution of individual contacts. This is particularly valuable for applications such as epidemic modeling, where understanding the timing and structure of contacts is crucial.

2.4 Deep Learning and Deep Generative Models

Deep learning and deep generative models have revolutionized many fields by enabling the modeling of complex data distributions and generating realistic synthetic data. In the realm of trajectory prediction and mobility network forecasting, these models have shown great promise in capturing the uncertainty and multimodal nature of human mobility.

2.4.1 Deep Learning Architectures

Deep learning architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Graph Neural Networks (GNNs) have been extensively applied to spatiotemporal problems. As described earlier, CNNs and RNNs have been used to model sequential dependencies in trajectory data, while GNNs have been instrumental in representing the spatial relationships in road networks or contact graphs [83, 35].

More recently, attention mechanisms and transformer architectures have been introduced to address some of the limitations of traditional recurrent models. The Transformer model, introduced by Vaswani et al. [84], employs self-attention to process sequences in parallel, thereby effectively capturing long-range dependencies. In the context of trajectory prediction, treating sequences of spatial locations (e.g., hexagon-based tessellated maps) as language has proven to be a novel and promising approach [85]. This paradigm shift has opened the door to applying large language models (LLMs) such as GPT-2, PaLM [86], and LLaMA [87] to non-textual domains.

2.4.2 Generative Models

Generative models aim to learn the underlying probability distribution of data, enabling the generation of new, realistic samples. The most well-known generative models include Generative Adversarial Networks (GANs) [88], Variational Autoencoders (VAEs) [89], autoregressive models [90], flow-based models [91], and, more recently, diffusion models [92].

GANs have been successfully applied to generate realistic images, videos, and even trajectories. For example, TrafficGAN [38] uses an adversarial framework to predict traffic flow by integrating CNNs and LSTMs. In trajectory prediction, GAN-based models have been employed to simulate realistic movement paths while preserving the interaction fidelity among agents.

VAEs offer another approach by learning a latent representation of the data in a probabilistic framework. When applied to trajectory data, VAEs can model the inherent uncertainty and variability in human movement. Recent work has focused on integrating structured priors and temporal encoders into VAEs to improve the quality of generated trajectories [93, 94].

Autoregressive models, which generate data sequentially, have also been adapted for

trajectory prediction. These models predict each subsequent point in a trajectory based on the previously generated points, providing a natural framework for sequential data generation. The integration of attention mechanisms into autoregressive models has further enhanced their ability to capture long-range dependencies.

Diffusion models, an emerging class of generative models, iteratively refine random noise into coherent data samples. They have recently shown promise in generating high-fidelity sequences in spatiotemporal settings [92]. By modeling the reverse diffusion process, these models can capture the multi-modal distribution of future trajectories, which is particularly useful for applications where uncertainty is high.

Transformers

Transformer models [84] are known for their use in NLP tasks but can also generate sequences in other domains. Traditionally, sequence data was processed using RNNs [95], which suffered from limitations like vanishing gradients and sequential computation, slowing their training. The Transformer addressed these issues by employing a self-attention mechanism, which allows for parallelization and learning of long-range dependencies in the data. In context of this thesis, we treat historical trajectories as sequences of blocks on a tessellated map. These sequences are analogous to sequences of tokens in language models [14, 86, 87].

2.4.3 Limitations for Trajectory Prediction

While deep generative models have achieved significant success in various applications, traditional time series forecasting methods, even those leveraging transformer architectures, are not directly applicable to our trajectory prediction problem. This is primarily because conventional methods typically predict continuous scalar or vector values at fixed intervals [12,

13], whereas our approach abstracts raw trajectory data into sequences of discrete spatial units via map tessellation, effectively mitigating issues of GPS noise and data sparsity. Additionally, it has been demonstrated that even leveraging pre-trained LLM models does not necessarily yield superior performance on standard time series tasks [96], highlighting the need for a sophisticated design in case of using similar architecture for such tasks. Moreover, trajectory prediction requires not only forecasting future positions but also ensuring spatial continuity and adherence to real-world constraints, a challenge that standard forecasting models do not inherently address.

Chapter 3

Problems of Interest

In this chapter, we introduce the fundamental notations, definitions, and preliminary concepts that underpin our research on mobility network forecasting. In our work, we aim to predict the future states of mobility networks by first forecasting individual trajectories and then inferring contacts among moving entities. This chapter lays the groundwork by formally defining key concepts such as the geographic map, trajectories, partial trajectories, prediction horizons, contacts, and network structures. We subsequently present two interrelated problems: **Trajectory Prediction** and **Mobility Network Prediction**. Throughout this chapter, we use these definitions and notations to build a solid foundation for the methodologies proposed in later chapters.

3.1 Preliminary Definitions

Before delving into the formal problem definitions, it is necessary to establish and clarify the notations and core concepts used throughout this thesis. Table 3.1 summarizes the key symbols and their descriptions. In what follows, we describe the definitions required for next

Symbol	Description
$\mathcal{M}$	The map of a geographic area
$\mathcal{B}$	Set of hexagonal blocks forming a tessellation of $\mathcal{M}$
T	Trajectory sequence of spatiotemporal points
T_u	Trajectory of individual u
p_i	A spatiotemporal point with location and time, i.e., $p_i = (l_i, t_i)$
T^l	Trajectory history of length l
l	Input trajectory length
τ	Distance threshold for determining contacts
$d_{u,v}$	Physical distance between individuals u and v
G_t	Proximity network at discrete time step t
V_t	Set of vertices in the proximity network at time t
E_t	Set of edges (contacts) in the proximity network at time t
G	Mobility network aggregated over a set of time steps
k	Prediction horizon (number of future time steps to predict)

Table 3.1: Summary of key notations used throughout the thesis.

chapters.

Definition 3.1 (Map). A map $\mathcal{M}$ represents the administrative boundaries of a finite and continuous geographic area, such as a city or a region. Since the area under consideration is relatively small, the curvature of the Earth's surface can be neglected. Consequently, we approximate $\mathcal{M}$ as a finite two-dimensional Euclidean space $\mathbb{R}^2$. This simplification allows us to work with standard Euclidean geometry in subsequent analyses.

Definition 3.2 (Trajectory). A trajectory denotes the movement of an individual or object over time. More formally, let $\mathcal{N} = \{u_1, u_2, \ldots, u_N\}$ be the set of individuals moving in the observation area $\mathcal{M}$ during a finite observation interval $[0, T]$. For any individual $u \in \mathcal{N}$, the trajectory T_u is defined as an ordered sequence of spatiotemporal points:

$$T_u = \{p_1, p_2, \ldots, p_n\} = \{(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)\},$$

where each p_i represents a distinct point in space at time t_i and l_i denotes the corresponding spatial coordinates (e.g., latitude and longitude).

Definition 3.3 (Partial Trajectory). Given a trajectory T_u for an individual u, a *partial trajectory* is a contiguous subsequence of T_u. Formally, given an initial index i and a length l, the partial trajectory is defined as:

$$T_u^l = \{p_i, p_{i+1}, \ldots, p_{i+l-1}\}.$$

This definition is useful for representing the observed segment of a trajectory that serves as input for prediction tasks.

Definition 3.4 (Trajectory History). The trajectory history of a specific length l is defined as the collection of all partial trajectories of that fixed length that have occurred in the past. This history, denoted as T^l, serves as a record for capturing recurrent movement patterns and temporal dependencies, which are critical for making accurate predictions of future positions.

Definition 3.5 (Prediction Horizon). The prediction horizon k specifies the number of future steps (or time intervals) that need to be predicted. In the context of trajectory forecasting, if a given partial trajectory T_u^l ends at time t, the prediction horizon k determines that the next k spatiotemporal points $p_{i+l}, p_{i+l+1}, \ldots, p_{i+l+k-1}$ should be estimated.

Definition 3.6 (Contact). As individuals move continuously through space, they may come close enough to each other, thereby forming a *contact*. Formally, a contact between two individuals u and v occurs at time t if the physical distance $d_{u,v}$ between them is less than or equal to a threshold τ, i.e.,

$$d_{u,v} \leq \tau.$$

This definition can be adjusted based on application-specific criteria; in our work, we focus on direct co-location within the same spatial block (see tessellation below) as the primary indicator of contact.

Definition 3.7 (Proximity Network). At any discrete time step t, a *proximity network* $G_t = (V_t, E_t)$ represents the instantaneous interactions among individuals. Here, the vertex set V_t corresponds to all individuals $u \in \mathcal{N}$, and the edge set E_t contains an edge (u, v) if and only if individuals u and v have a contact at time t (as defined above).

Definition 3.8 (Mobility Network). The *mobility network* is the dynamic graph formed by a sequence of proximity networks over the entire observation period $[0, T]$. Formally, if we denote the discrete time steps as $t = 1, 2, \ldots, T'$, then the mobility network $G = (V, E)$ is obtained from the collection $\{G_1, G_2, \ldots, G_{T'}\}$. This dynamic network captures the evolution of interactions (contacts) among individuals over time.

The above definitions enable us to transform raw spatiotemporal data (e.g., continuous GPS logs) into structured representations that are amenable to machine learning and network analysis. A particularly important step in this process is the spatial discretization of $\mathcal{M}$ via tessellation. In our study, we partition $\mathcal{M}$ into a set of hexagonal blocks $\mathcal{B} = \{b_1, b_2, \ldots, b_m\}$. This not only simplifies the representation of locations but also mitigates noise effects and facilitates the detection of contacts by defining co-location in terms of block membership.

Time discretization is similarly crucial. By mapping continuous timestamps $t \in [0, T]$ to discrete time steps, we obtain a sequence of snapshots. Each snapshot G_t represents a static proximity network, while the series $\{G_1, G_2, \ldots, G_{T'}\}$ naturally describes the dynamics of the system.

3.2 Problem Definitions

Building upon the aforementioned preliminaries, we now define the two primary problems that this thesis addresses. The first problem, **Trajectory Prediction**, involves forecasting future positions based on historical movement data. The second problem, **Mobility Network Prediction**, extends this task to the domain of dynamic networks by predicting future contacts (i.e., the edges in proximity networks).

3.2.1 Problem 1: Trajectory Prediction

Problem 1 (Trajectory Prediction). Given a map $\mathcal{M}$, a corresponding trajectory history T^l (i.e., a set of spatiotemporal point sequences of length l), a specific partial trajectory $T_i^l = \{p_{i_1}, p_{i_2}, \ldots, p_{i_l}\}$, and a prediction horizon $k > 0$, the objective is to predict the next k spatiotemporal points $p_{i_{l+1}}, p_{i_{l+2}}, \ldots, p_{i_{l+k}}$ of the given partial trajectory T_i^l.

Challenges in Trajectory Prediction:

- *Non-Stationarity:* Human mobility patterns can vary significantly over time, exhibiting sudden shifts due to external events or changes in routine. This non-stationary behavior challenges the assumption of consistent statistical properties over time.
- *Complex Temporal Dependencies:* Trajectories often encapsulate non-linear and complex temporal patterns that require sophisticated models to capture both short-term and long-term dependencies.
- *Sparsity and Noise:* Real-world trajectory data collected via mobile devices or GPS sensors can be sparse and subject to noise, which complicates the learning and prediction processes.

The successful resolution of the trajectory prediction problem is critical because it forms the basis for predicting future interactions among individuals in a mobility network.

3.2.2 Problem 2: Mobility Network Prediction

Problem 2 (Mobility Network Prediction). Given an observation area $\mathcal{M}$, a set of individuals or objects $\mathcal{N}$, and their trajectories $\{T_u\}_{u \in \mathcal{N}}$ recorded over the observation period $[0, T]$, along with a prediction horizon k, the goal is to predict the future k states of the mobility network. More concretely, we aim to forecast the future k proximity networks

$$G_{T'+1}, G_{T'+2}, \ldots, G_{T'+k},$$

where each network $G_t = (V_t, E_t)$ at time $t \in \{T' + 1, \ldots, T' + k\}$ represents the contacts among individuals at that future time step. Since the node set V_t generally remains constant (assuming all individuals continue to be present), the problem effectively reduces to predicting the edge sets E_t that capture the evolving interactions among individuals.

Challenges in Mobility Network Prediction:

- *Temporal Dynamics:* Accurately forecasting future contacts requires a deep understanding of the underlying temporal dynamics in individual trajectories. Dynamic changes in movement patterns need to be captured reliably.
- *Spatial Continuity and Realism:* The predicted interactions must be spatially coherent. That is, contacts should only occur when individuals are realistically close to one another based on the spatial discretization of $\mathcal{M}$.

In summary, mobility network prediction leverages the results of trajectory prediction to infer the future structure of interactions among individuals. By predicting when and where

individuals are likely to come into contact, we can construct dynamic models of social and spatial interactions that are critical for various real-world applications.

Chapter 4

Mobility Data Representation

In this chapter, we delve into the rationale behind using higher-order mobility data and explain the benefits of this data representation approach. We detail the transformation process from raw GPS trajectories to sequences of higher-order geometric elements (hexagons) that offer significant advantages in terms of sparsity reduction, compatibility with machine learning models, and enhanced generalization. In addition, we revise our original problem statements to incorporate higher-order trajectory representations, thereby aligning our forecasting tasks with this new paradigm. This chapter is organized into several sections: we begin with the motivation for using higher-order mobility representations, proceed to discuss how trajectories can be treated as sequences of geometric elements, describe the data generation pipeline, and conclude by revisiting the problem definitions in the context of higher-order mobility flow data.

4.1 Motivation

Mobility datasets available for research purposes are often inconsistent in terms of quality, documentation, and accessibility. Many datasets are discovered through ad-hoc searches and lack extensive metadata and reproducible generation processes. In light of these challenges, researchers frequently resort to generating synthetic mobility datasets; however, such datasets are not always published or well-documented, which impacts reproducibility across studies.

Furthermore, the raw mobility data in the form of GPS traces are inherently difficult to handle due to their continuous nature, sparsity, and the high volume of data required for meaningful statistical analysis. These challenges are amplified when attempting to use conventional machine learning models that are better suited to structured or discretized data. To address these issues, a novel representation of mobility data based on higher-order geometric elements has gained significant attention [97]. This approach involves tessellating a map with geometric primitives – in our case, hexagons – and representing trajectories as sequences of these hexagonal elements.

The advantages of this representation are manifold:

- **Reduced Sparsity:** By aggregating individual GPS points into fixed geometric blocks (hexagons), data sparsity is reduced. This aggregation increases the density of the dataset, enabling more effective learning even with limited raw data points.

- **Compatibility with Machine Learning Models:** The discrete and structured format of hexagon sequences aligns well with many popular machine learning models, particularly those based on sequence learning such as recurrent neural networks (RNNs) and transformer architectures.

- **Enhanced Generalization and Visual Analytics:** The abstract representation helps

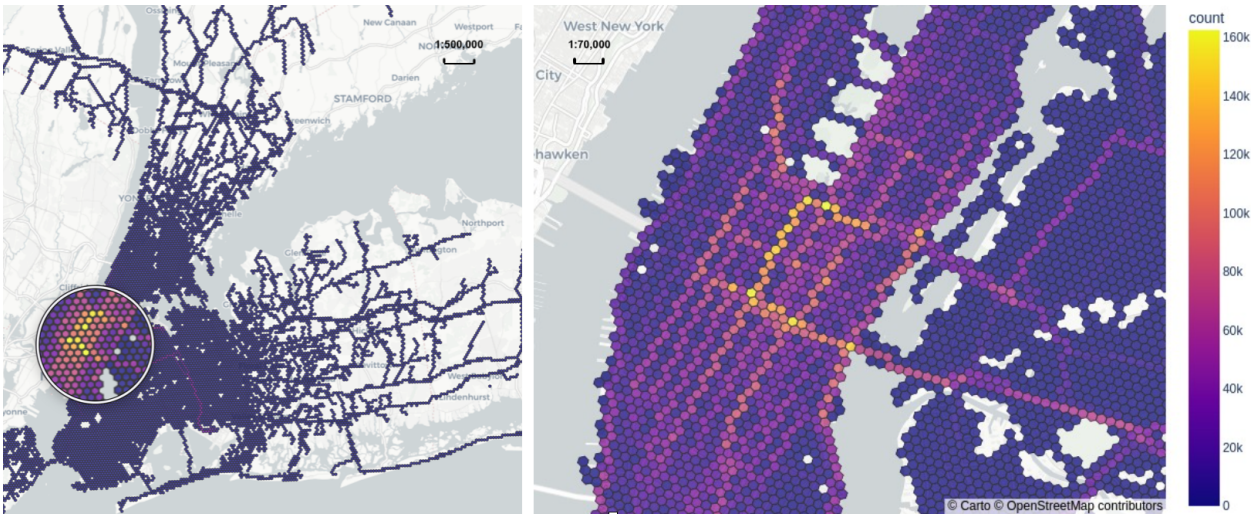


Figure 4.1: An example of hexagon-sequence trajectories on a tessellated map.

in reducing overfitting to noisy data and supports more efficient visual analytics. For instance, trajectories represented as sequences of hexagons can be easily visualized as heatmaps, providing immediate insight into mobility patterns.

Figure 4.1 illustrates taxi trajectories in New York City mapped as sequences of hexagons. This representation provides a visually and analytically coherent way to understand movement patterns within a complex urban environment.

4.2 Treating Trajectories as Sequences

Raw mobility data typically consists of a series of GPS points recording the continuous movement of individuals. However, such data is often sparse, noisy, and not immediately suitable for direct use in machine learning models. To address these shortcomings, it is necessary to transform the raw data into a discretized and structured representation. One effective approach is to abstract continuous trajectories into sequences of higher-order elements, such as hexagons derived from a tessellated map. This transformation not only simplifies

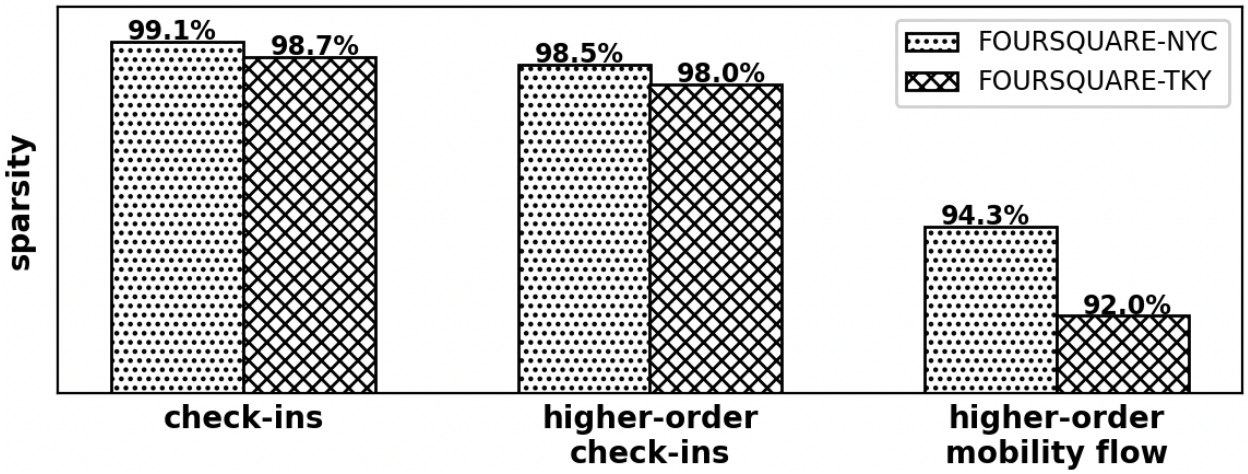


Figure 4.2: Impact of higher-order mobility flow abstraction on sparsity.

analysis but also makes the data compatible with sequence learning models.

In this framework, each GPS point is first aligned with the underlying road network using map-matching techniques. Once the points are accurately aligned, their continuous coordinates are mapped onto corresponding hexagonal blocks that form a tessellation of the observation map. As a result, the continuous trajectory is converted into a sequence of discrete hexagon identifiers, representing the trajectory in step-wise fashion.

Figure 4.2 demonstrates the impact of this higher-order abstraction on data sparsity. Benchmark datasets, such as Foursquare-Nyc and Foursquare-Tky [98], exhibit notable improvements in density after the transformation. For example, converting raw check-ins into hexagon sequences can lead to a reduction in sparsity by approximately 0.7% (a 60% increase in density). When focusing on higher-order mobility flow representations, sparsity may decrease by around 5%, resulting in more than a 500% increase in density. Such improvements facilitate more robust statistical analysis and enhance the performance of downstream machine learning models.

The benefits of treating trajectories as sequences of blocks extend beyond sparsity reduction.

The structured nature of these sequences inherently provides a natural ordering that captures the dynamic evolution of movement patterns. This temporal framework is particularly advantageous for sequence learning models, such as recurrent neural networks (RNNs) and transformer architectures, which are designed to process sequential inputs. By converting raw GPS data into block sequences, we retain the essential spatial-temporal information while filtering out high-frequency noise and minor fluctuations inherent in the raw data.

4.2.1 Benefits of Hexagon-Based Tessellation

Hexagon-based tessellation offers several distinct advantages over more conventional partitioning methods such as square grids:

- **Uniform Neighbor Relationships:** Each hexagon has exactly six neighbors, all equidistant from its center. This uniformity simplifies the definition of proximity and ensures that spatial relationships are isotropic, thereby reducing directional biases.
- **Closer Approximation to Circular Regions:** Hexagons better approximate circular shapes compared to squares. A unit-area hexagon has a perimeter of approximately 3.722, whereas a square of the same area has a perimeter of 4. This lower perimeter-to-area ratio minimizes edge effects in spatial analyses.
- **Simplified Nearest Neighborhood Definition:** The uniform structure of hexagons eliminates ambiguities inherent in grid-based tessellations, where neighbors may be categorized as orthogonal or diagonal with differing distances. With hexagons, all neighbors are defined consistently, streamlining proximity computations.
- **Scalability and Resolution Flexibility:** Tessellation can be performed at varying resolutions by adjusting the hexagon size. A finer resolution (smaller hexagons) yields a

more detailed representation of trajectories but requires higher computational cost; conversely, coarser resolutions facilitate faster processing while providing a broader overview of movement patterns.

These characteristics make hexagon-based representations particularly well-suited for use in modern machine learning models. In transformer architectures, for example, each hexagon can be treated as a discrete token. The uniform geometry of hexagons ensures that transitions between tokens reflect consistent spatial relationships, thereby enhancing both interpretability and predictive performance in sequence-to-sequence tasks.

In summary, by treating trajectories as sequences of hexagonal blocks, we address several critical challenges:

- The transformation reduces sparsity and increases data density, which is essential for effective model training.
- The structured sequence inherently provides a temporal framework that reflects the dynamic evolution of mobility patterns.
- The advantages of hexagon-based tessellation—uniform neighborhood relationships, accurate spatial approximations, and scalability—enhance model compatibility and predictive performance.

4.3 Preliminaries and Definitions

In this section, we formalize the key concepts used in our higher-order mobility representation framework. These definitions closely mirror those introduced in Chapter 3, with added emphasis on the higher-order representation of trajectories.

Definition 4.1 (Map). See the definition from Chapter 3.

Definition 4.2 (GPS Trace Points). A GPS trace point is defined as a triplet $s = (o, t, \langle x, y \rangle)$, where o represents the moving object, t is the timestamp, and $\langle x, y \rangle$ denotes the geocoordinate tuple (longitude and latitude). The set of all trace points is denoted by S.

Definition 4.3 (Trajectory). See the definition from Chapter 3.

Definition 4.4 (Map Tessellation). Let $\mathcal{B} = \{b_1, b_2, ..., b_n\}$ be a set of disjoint blocks that entirely tessellate the map $\mathcal{M}$, forming a regular tiling. In our study, we choose hexagons as the polygonal blocks. The terms "hexagons" and "blocks" are used interchangeably. Hexagon-based tessellation offers several advantages over traditional square grid tessellations, including more uniform neighbor relationships and reduced edge effects [99]. Notably, the tessellation can be performed at different levels of resolution by defining various hexagon sizes—the smaller the hexagon, the higher the resolution.

Definition 4.5 (Higher-order Trajectory). Given a trajectory $P = p_1 p_2 \dots p_n$, and noting that each point p_i is located within a unique block $b_i \in \mathcal{B}$ of the tessellated map $\mathcal{M}$, the trajectory can be translated into a sequence of blocks. This sequence is called a *higher-order trajectory*. The benefit of this representation is that it discretizes the continuous space into a series of ordered steps, which can then be processed using sequence-based learning algorithms.

Definition 4.6 (Higher-order Mobility Flow). Given a higher-order trajectory $P = \{b_1, b_2, \dots, b_m\}$, we can further describe the movement by considering not only the blocks where the object is observed, but also the transitions between blocks. This leads to the concept of a *higher-order mobility flow*, represented as:

$$P = \{b_1,\ \mathcal{B}_{[1,2]},\ b_2,\ \mathcal{B}_{[2,3]},\ \dots,\ \mathcal{B}_{[m-1,m]},\ b_m\},$$

where each $\mathcal{B}_{[i,i+1]} \subset \mathcal{B}$ denotes the sequence of blocks traversed while moving from b_i to b_{i+1}.

By representing trajectories as sequences of hexagons, we implicitly assume that objects

move in a discrete, step-wise manner. This transformation aligns well with existing machine learning models and facilitates further analysis such as trajectory prediction and mobility network construction.

4.4 The Data Generation Pipeline

Transforming raw trajectory data into higher-order mobility flow representations involves several stages, each critical to ensuring the quality and usability of the transformed data. In this section, we describe the complete pipeline for generating higher-order mobility flow data.

4.4.1 Input Data

The pipeline begins with raw trajectory datasets provided in the form of GPS trace points. An example input file might have the following structure:

```
taxi_id, date_time, longitude, latitude
1, 2008-02-02 15:36:08, 116.51172, 39.92123
1, 2008-02-02 15:46:08, 116.51135, 39.93883
```

Each line in the dataset represents an observation of a moving object (e.g., a taxi) at a specific time and geographic location.

4.4.2 Map-Matching

Raw GPS data are inherently noisy and may deviate from actual road networks due to errors and sampling frequency. To mitigate these issues, the map-matching process is applied to align the raw GPS points with the underlying road network. Popular routing systems, such as the Open Source Routing Machine (OSRM)[1], are often employed to find the most likely paths

[1] https://project-osrm.org/docs/v5.24.0/api/#route-service

between consecutive GPS points. By concatenating these paths, we obtain a map-matched trajectory that reflects the true route taken by the object.

4.4.3 Computational Geometry and Tessellation

Once the trajectories have been map-matched, computational geometry techniques are applied to convert these paths into sequences of higher-order geometric elements. The steps include:

1. **Tessellation:** The map $\mathcal{M}$ is tessellated into a set of hexagonal blocks $\mathcal{B}$ (see Definition 4.4). The choice of hexagons provides uniformity in neighboring relationships and minimizes edge effects.

2. **Mapping Points to Blocks:** Each map-matched GPS point is then associated with the corresponding hexagon by converting its latitude and longitude coordinates into the tessellated coordinate system. This operation is performed in constant time using specialized coordinate system conversions [100].

3. **Formation of Higher-Order Trajectories:** With all points mapped to hexagons, the original continuous trajectory is converted into a sequence of blocks. This sequence is our higher-order trajectory (Definition 4.5). Optionally, information about the intermediate blocks traversed between two observed blocks may be retained to construct a higher-order mobility flow (Definition 4.6).

4.4.4 Pipeline Overview

Figure 4.3 provides a schematic overview of the data generation pipeline, illustrating the transformation from raw GPS traces to higher-order trajectory representations in the form of hexagon sequences.

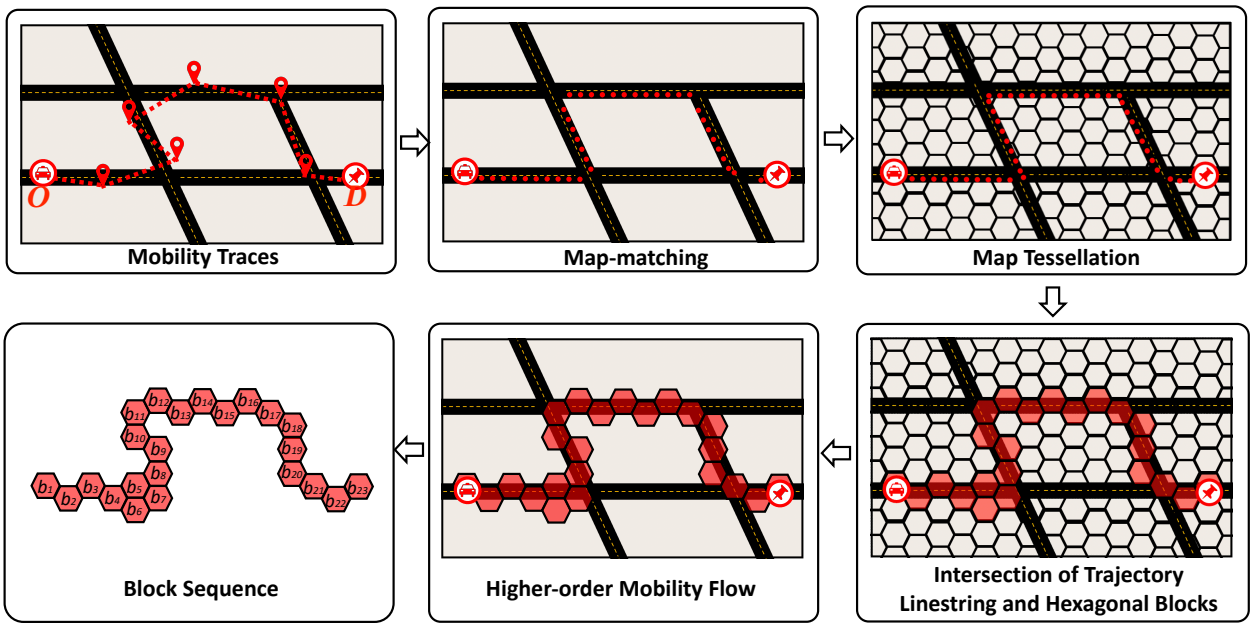


Figure 4.3: Construction pipeline for higher-order mobility flow data.

Figure 4.4 further demonstrates examples of map tessellation at different resolutions, highlighting how varying hexagon sizes influence the level of detail in the representation.

This pipeline, though conceptually straightforward, is computationally intensive due to the involvement of map-matching and computational geometry tasks. However, the benefits of using a higher-order representation—such as reduced sparsity and increased compatibility with machine learning models—are well worth the additional processing overhead.

4.5 Problem Statements (Revisited)

With the higher-order mobility data representation in place, we now revisit the original problem definitions to reflect this abstraction. Rather than predicting continuous GPS coordinates, the objective is now to forecast discrete sequences of hexagon blocks.

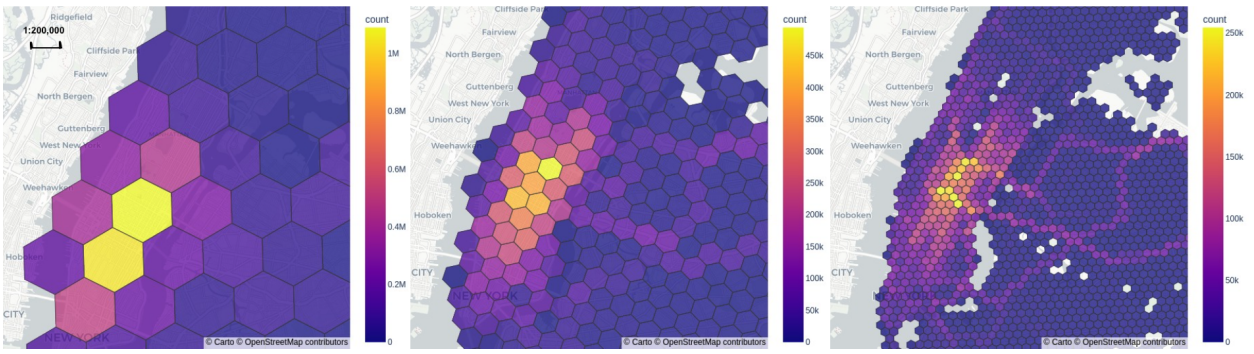


Figure 4.4: An example of varying resolutions – from lower to higher resolution, from left to right.

4.5.1 Problem 1 (Revisited): Higher-Order Trajectory Prediction

Problem 3 (Higher-Order Trajectory Prediction). Given a map $\mathcal{M}$, a corresponding trajectory history T^l represented as a set of block sequences, a partial trajectory $T_i^l = b_{i_1} b_{i_2} \dots b_{i_l}$ (where each b_i is a hexagon block), and a prediction horizon $k > 0$, the objective is to predict the next k blocks $b_{i_{l+1}}, b_{i_{l+2}}, \dots, b_{i_{l+k}}$ that will form the continuation of the partial trajectory T_i^l.

This formulation leverages the higher-order representation, enabling a more structured and effective prediction model. Moreover, using hexagon sequences capitalizes on the reduced sparsity and the uniform spatial relationships provided by hexagon-based tessellation.

4.5.2 Problem 2 (Revisited): Higher-Order Mobility Network Prediction

Using the higher-order abstraction, we redefine the concept of a *contact*. In this context, a contact between two individuals u and v occurs when they are simultaneously present in the same hexagon (or in adjacent hexagons, if a relaxed definition is adopted) at the same time step. The aggregation of such contacts over discrete time intervals forms the proximity

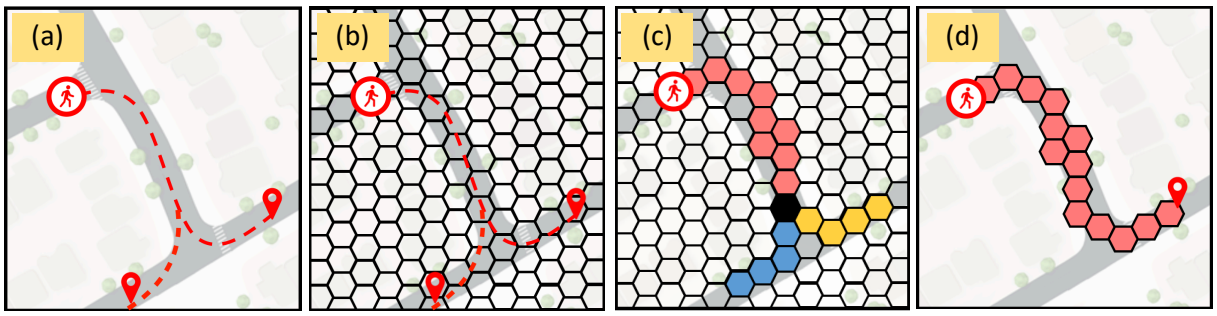


Figure 4.5: Illustrative example of the trajectory prediction problem using higher-order mobility flow representations (hexagons); (*a*) there are two potential trajectories for the pedestrian, (*b*) trajectories are represented on a hexagon-based tessellated map, (*c*) given the historical data (red) and the current location (black), two trajectories are predicted (blue and orange), (*d*) the actual trajectory followed.

networks, which in turn constitute the dynamic mobility network.

Problem 4 (Higher-Order Mobility Network Prediction). Given an observation area (map) $\mathcal{M}$, a set of objects $\mathcal{N}$, and their higher-order trajectories P_u, where each P_u is represented as a sequence of hexagon blocks $P_u = b_{i_1} b_{i_2} \dots b_{i_l}$ for $u \in \mathcal{N}$ observed during the interval $[0, T]$, along with a prediction horizon k, the task is to predict the future k states of the mobility network. Equivalently, the goal is to forecast the future k proximity networks $G_t = (V_t, E_t)$ for $t \in [T+1, T+k]$, where an edge exists between two individuals if they have a contact in the same hexagon at time t.

The enhanced representation of trajectories as hexagon sequences not only improves the granularity and accuracy of trajectory prediction but also supports more robust and realistic mobility network forecasting. The trade-off between resolution and computational efficiency is controlled by the size of the hexagons—the higher the resolution (smaller hexagons), the more detailed the prediction, albeit with increased computational cost.

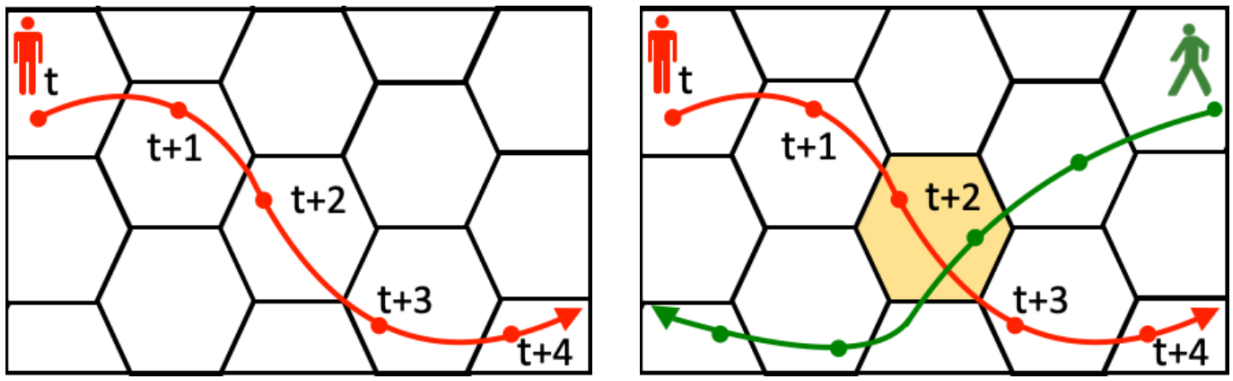


Figure 4.6: In the context of higher-order mobility flow, a contact occurs when two objects are located within the same hexagon (yellow) at the same discrete time step (time $t + 2$).

Chapter 5

Proposed Methodology

In this chapter, we present the comprehensive methodology behind our work, which integrates advanced trajectory prediction with dynamic network construction to forecast future mobility networks. Our approach is divided into two main components. First, we describe our deep learning framework for predicting the future paths of individual objects, building on a Transformer-based architecture that treats trajectories as sequences of discrete spatial tokens. Then, we explain how these trajectory predictions are used to detect contacts—by mapping them onto synchronized, time-stamped spatial blocks—and to construct dynamic mobility network snapshots. Together, these components allow us to forecast not only the future locations of moving objects but also the evolving patterns of interactions among them.

5.1 Trajectory Prediction Learning

The first cornerstone of our methodology is a robust trajectory prediction module that leverages the power of Transformer-based models to capture the complex spatiotemporal dependencies underlying individual movement patterns. In our TrajLearn framework,

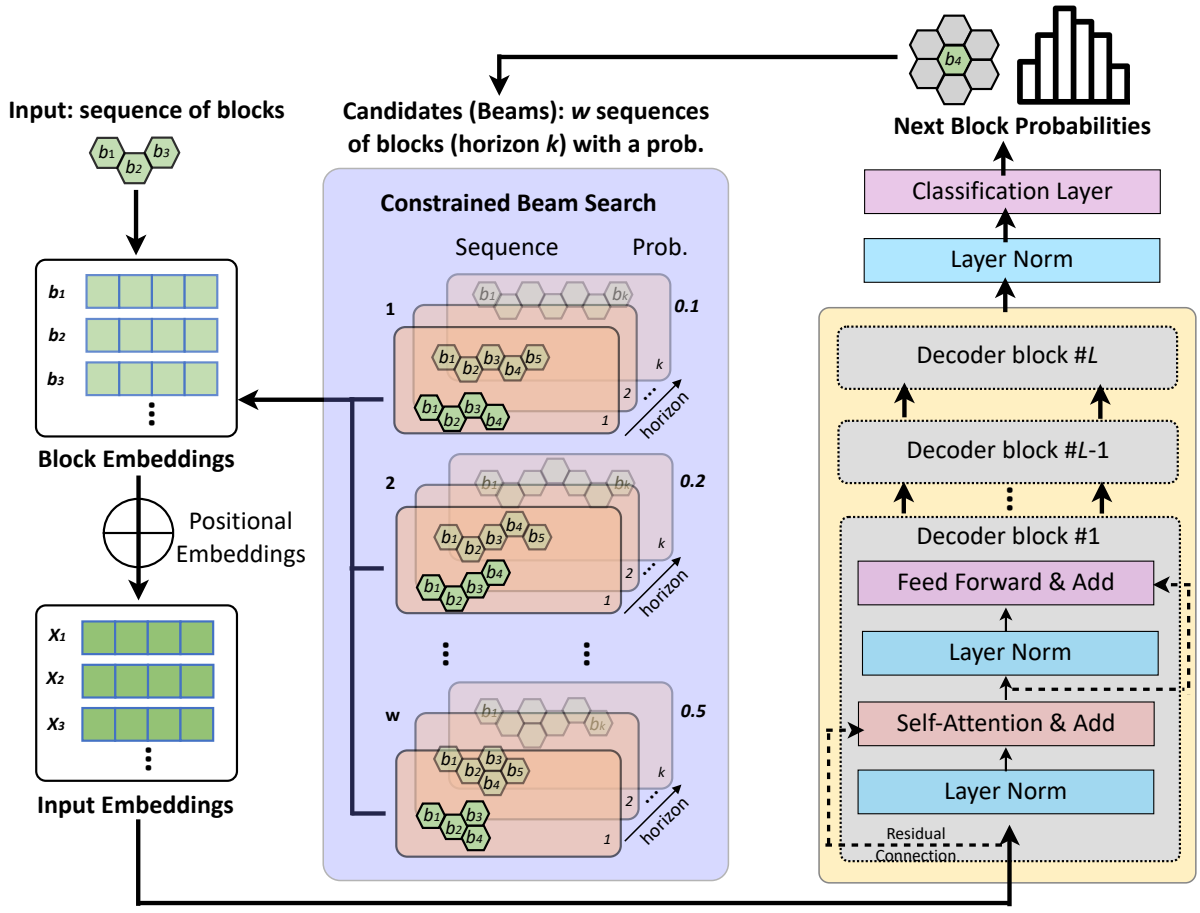


Figure 5.1: TRAJLEARN's high-level architecture.

trajectories are represented as sequences of hexagon blocks obtained via a higher-order mobility flow representation, effectively transforming spatial paths into a sequence prediction problem akin to language modeling. Our discussion is organized around three main aspects: (i) the design of TRAJLEARN which leverages the Transformer architecture to capture the intricate dependencies present in trajectory data, (ii) the training methodology employed to enable accurate trajectory prediction, and (iii) the beam search strategy with constraints that efficiently explores multiple plausible future trajectories. In addition, we analyze the computational complexity of our model in Section 5.1.4.

5.1.1 The TrajLearn Model

To tackle the challenging task of trajectory prediction in complex mobility networks, we propose TrajLearn, a model that is built on the Transformer architecture [84]. Although the Transformer was originally designed for natural language processing tasks, it is highly effective at modeling sequential data in various domains. In our approach, we draw a close analogy between language and trajectories by regarding a `token` (or word) as equivalent to a `hexagon ID` (or block) in a tessellated map. Just as language models learn dependencies between words in a sentence, our model learns dependencies between hexagons in a trajectory. The sequence of hexagons (obtained via our higher-order mobility flow representation) forms the backbone of our prediction task.

Figure 5.1 illustrates the high-level architecture of TrajLearn. Our model is a decoder-only Transformer comprising L layers, each featuring A causal self-attention heads and a hidden state dimension of H. Unlike the original Transformer architecture that employs sinusoidal positional encodings, we use learned positional embeddings. These embeddings are capable of adapting to the complex patterns present in trajectory data. The input to our Transformer model is formulated as follows:

$$h_0 = BW_e + W_p \tag{5.1}$$

where $B = (b_1, b_2, \ldots, b_l)$ represents the input sequence of hexagon blocks (i.e., the higher-order mobility flow), W_e is the block embedding matrix, and W_p is the learned positional embedding matrix. Furthermore, unlike in the original Transformer, Layer normalization was moved to the input of each sub-block, and an additional Layer normalization was added after the final self-attention block. Each layer $j \in \{1, 2, \ldots, L\}$ of the Transformer processes the hidden state from the previous layer using a residual connection, causal multi-head

self-attention, and a position-wise feed-forward network. The hidden state at layer j is computed as:

$$h'_j = h_{j-1} + \texttt{Self-Attention}(\texttt{LayerNorm}(h_{j-1})), \tag{5.2}$$

$$h_j = h'_j + \texttt{FeedForward}(\texttt{LayerNorm}(h'_j)) \tag{5.3}$$

where $\texttt{LayerNorm}(\cdot)$, $\texttt{Self-Attention}(\cdot)$, and $\texttt{FeedForward}(\cdot)$ denote layer normalization, the causal multihead self-attention operation, and the position-wise feed-forward network, respectively. For the `LayerNorm`, the module utilizes a modified version of L2 regularization proposed in [101] on all non-bias or gain weights. As an activation function, we opted for the Gaussian Error Linear Unit (`GELU`) [102], which is chosen due to its performance in NLP tasks and its ability to alleviate the vanishing gradient problem, allowing for a more effective learning process. `GELU` is defined as:

$$\texttt{GELU}(x) = x \cdot P(X \leq x) \tag{5.4}$$

where $X \sim N(0, 1)$ follows the standard normal distribution. In implementation, this is approximated by:

$$0.5x \left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{5.5}$$

The self-attention mechanism is central to our model. In causal self-attention, each token in the sequence is allowed to attend only to tokens that precede it. This causality constraint is crucial for autoregressive sequence generation. Formally, the self-attention operation is

defined as:

$$\texttt{Self-Attention}(E) = \texttt{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \mathbf{M}\right)V,$$
$$\text{with} \quad \mathbf{M}_{i,j} = \begin{cases} -\infty, & \text{if } j > i, \\ 0, & \text{if } j \leq i, \end{cases} \qquad Q = EW_Q,\ K = EW_K,\ V = EW_V, \tag{5.6}$$

where Q, K and V are matrices representing the `queries`, `keys` and `values`, respectively, W_Q, W_K, and W_V represent the respective learnable weight parameter matrices, and d_k is the dimensionality of the `keys`. The matrix $\mathbf{M}$ is used to mask out future positions in the sequence, so the output of the self-attention layer for each position depends only on tokens to its left, ensuring causality in the attention mechanism. This mechanism allows the model to focus on different parts of the input sequence when generating the output. The output of the last layer is fed into layer normalization, followed by a linear projection and a `softmax` activation that predicts the next block in the trajectory based on the probabilities of all possible next blocks:

$$P(b_{l+1} \mid B) = \texttt{softmax}\left(\texttt{FeedForward}(\texttt{LayerNorm}(h_L))\right) \tag{5.7}$$

This output represents the conditional probability of the next block given the observed sequence, enabling the model to generate future trajectories in an autoregressive manner.

Although we followed a decoder-only transformer architecture, our choice is motivated by the autoregressive nature of trajectory prediction, which requires the sequential generation of future spatial tokens conditioned solely on past trajectory data. A decoder-only transformer directly models the conditional distribution of each future hexagonal block given the preceding sequence, thereby simplifying both training and inference. This architecture also reduces

computational overhead and enables seamless integration with our constrained beam search mechanism (subsection 5.1.3) to enforce spatial continuity between adjacent blocks. While encoder-decoder architectures may be beneficial for tasks involving more complex input-output mappings, our focus on the trajectory prediction task, renders the decoder-only approach particularly efficient. Nonetheless, our approach for trajectory prediction is flexible, and **other architectures used in large language models could be used**. Thus, any advancements in language models are applicable and can benefit our approach with minimal effort.

5.1.2 Model Training

Training the TrajLearn model requires careful consideration of both the trajectory data representation and the optimization strategies employed. In our framework, trajectories are represented as sequences of hexagon blocks. To signal the end of a trajectory, we introduce a special token, `<End of Trajectory>` (or `<EOT>`). This token indicates that the trajectory has concluded, mirroring the role of the `<EOS>` token in language modeling tasks. The introduction of the `<EOT>` token is critical—it not only allows the model to learn when trajectories naturally come to an end, but also helps to enforce spatial continuity by encouraging transitions between adjacent hexagons rather than random jumps.

Once all trajectories are transformed into sequences of hexagon blocks, we generate partial trajectories for training. For each trajectory, we consider segments of length $l + k$, where the first l blocks serve as input and the subsequent k blocks represent the target for prediction. Here, $l \geq 1$ is the input length parameter, and $k \geq 1$ is the prediction horizon. During training, the model uses the first l blocks of a partial trajectory to predict the remaining blocks, continuing the process until the `<EOT>` token is encountered or the prediction horizon

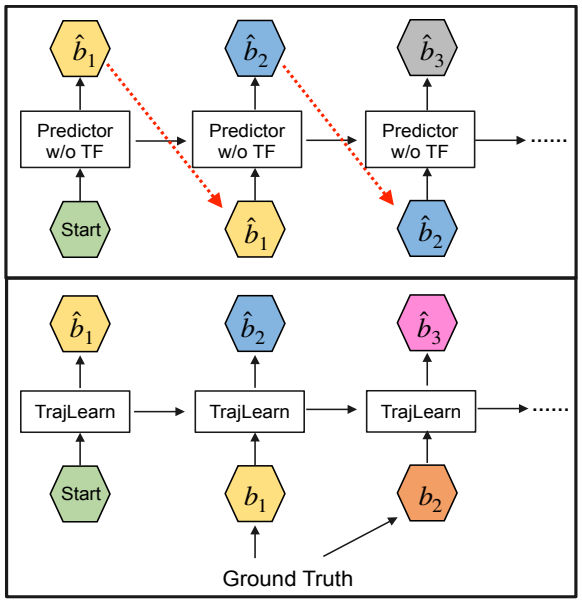


Figure 5.2: Model training with (bottom) and without (top) teacher forcing.

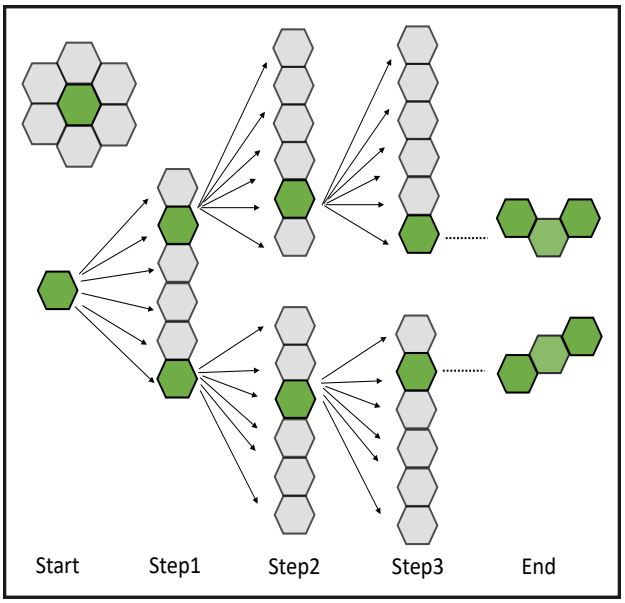


Figure 5.3: Beam search example where $w = 2$ and $k = 3$.

is reached.

Our training procedure incorporates the technique of *teacher forcing*. With teacher forcing, during each training step the model's next input is not the predicted token from the previous step, but rather the ground truth token. This strategy helps mitigate the compounding errors that can arise in autoregressive prediction, thereby stabilizing and accelerating the convergence process. Figure 5.2 visually contrasts the training process with and without teacher forcing, illustrating how the ground truth is injected into the sequence during training. It is important to note that teacher forcing is used exclusively during training and is not applied during inference, where the model must rely on its own predictions to generate future tokens.

5.1.3 Beam Search with Constraints

To optimize the trajectory prediction process and improve the performance, we incorporate *beam search with constraints*. Beam search is a heuristic search algorithm that explores the most promising trajectory paths. It maintains a set of candidate sequences of blocks and generates new candidate sequences at each step by expanding the current best candidates. By guiding the search process and focusing on the most likely trajectory paths, beam search improves prediction performance and increases the quality of the predicted trajectories. Figure 4.5 shows an example of exploring two paths. Our model's beam search involves the following stages:

- **Initialization.** The algorithm begins with the last visited block of the current trajectory as its initial state. It selects a set of candidate next blocks using the output probabilities provided by the model's classification layer.

- **Beam Expansion.** Each candidate in the beam is expanded by one block, generating a new set of candidate blocks for the next step. The expansion process is guided by the spatial relationships between the blocks, allowing expansion only to geographically adjacent blocks based on the hexagon-based map tessellation, The probabilities for each candidate in the beam are updated based on their cumulative probabilities as follows:

$$P(b_{i_1} \dots b_{i_n}) = P(b_{i_1} \dots b_{i_{n-1}}) \times P(b_{i_n} | b_{i_1} \dots b_{i_{n-1}}) \tag{5.8}$$

- **Beam Pruning.** After expansion, the beam is pruned to a certain beam width w, representing the most likely (or top) candidates for expansion based on their cumulative probabilities.

- **Termination.** The algorithm continues iterating through the beam expansion and pruning

stages until it either reaches the desired prediction horizon or all candidates encounter an <EOT>.

Figure 5.3 depicts an illustrative example of the beam search with constraints with beam width $w = 2$ and prediction horizon $k = 3$.

Ensuring the Validity of Predicted Trajectories

To preserve *spatial continuity* in the predicted trajectories, we introduce constraints to the beam search algorithm. Formally, for a current block $b_i \in \mathcal{B}$, let $\Gamma_{\mathcal{M}}(b_i)$ denote the set of its adjacent blocks in $\mathcal{M}$. Then, at each step, the path can only expand from b_i towards one of its adjacent blocks $b_j \in \Gamma_{\mathcal{M}}(b_i)$. By restricting the model to only consider adjacent blocks during the next block prediction task, we ensure *the validity* of the predicted trajectories, as they are always guided to follow spatially connected paths. This constraint further enhances the overall prediction *accuracy* by preventing the inclusion of non-adjacent blocks in the next block prediction task.

5.1.4 Computational Complexity

An important aspect of our model is understanding its computational and memory complexity, particularly as it relates to real-time inference. The TrajLearn relies on a Transformer architecture, the main operations of which include self-attention and feed-forward neural networks. During inference, we incorporate a beam search component. Below, we analyze the computational complexity of our model.

Self-Attention. The computational complexity of the self-attention mechanism is $O(n^2 \cdot d)$, where n is the sequence length and d is the dimension of the representation. Self-attention involves comparing each element in the sequence (like a word in a sentence) with every other

element. This requires $n \times n$ comparisons or attention scores to be computed, leading to a quadratic complexity in terms of sequence length, and then, for each comparison, the model computes attention scores based on token representations, which are vectors of size d. This computation involves these vectors and hence introduces a factor of d in the complexity.

Beam Search. During the inference time, the beam search algorithm is utilized with a complexity of $O(w \cdot \beta \cdot k)$. Here, w is the beam width, k is the prediction horizon (depth of the search tree), and β is the branching factor for each beam. At each level of the tree, the algorithm examines w nodes, and for each of these nodes, it considers up to β child nodes. However, only the best w among these $w \cdot \beta$ nodes are retained for the next level. Therefore, the number of nodes evaluated at each level is proportional to $w \cdot \beta$, and this process repeats for k levels, which is our prediction horizon.

Hexagonal Tessellation. Mapping a GPS coordinate to its corresponding hexagon is performed in constant time, i.e., $O(1)$, due to efficient coordinate-to-hexagon conversion methods. This ensures that the transformation from raw trajectory data to a hexagonal representation introduces minimal computational overhead. While the map-matching process, which aligns raw GPS data with the appropriate road network or spatial features, can be computationally intensive when processing large-scale data for training and testing, it is executed as part of an offline preprocessing pipeline. This separation guarantees that these heavy-lifting tasks do not impact the speed of the online inference process. Moreover, for real-time trajectory inference in practical applications, incremental map-matching techniques can be employed to update the trajectory representation dynamically as new data arrives, without significant overhead. Consequently, although the offline preprocessing stage may be resource-intensive, it is decoupled from real-time operations, ensuring that TrajLearn remains well-suited for online or real-time applications.

Table 6.4 compares the computational complexity, memory complexity, and the number of trainable parameters for TrajLearn and several baseline models. These baselines include conventional approaches such as Markov Chains (MC), LSTM-based models, GRU-based models, and state-of-the-art methods like DeepMove and Flashback++. Our analysis demonstrates that TrajLearn maintains competitive complexity while delivering state-of-the-art performance.

5.2 Mobility Network Prediction

Building upon our trajectory prediction capabilities, the second major component of our methodology focuses on constructing dynamic mobility networks from the forecasted trajectories. In this module, each predicted trajectory is aligned with discrete time steps, forming a sequence of block-time pairs that allow for the precise identification of contacts between moving objects. A contact is defined to occur when two objects are predicted to occupy the same hexagon at the same time step. By aggregating these contacts across the prediction horizon, we generate a series of network snapshots that collectively represent the evolving mobility network. This process transforms individual movement forecasts into a comprehensive, time-varying graph that encapsulates the dynamics of human interactions, offering critical insights for applications such as epidemic modeling, urban planning, and intelligent transportation systems.

5.2.1 The MobiNetForecast Framework

Our approach, which we refer to as the MobiNetForecast framework, leverages the strengths of our Transformer-based trajectory prediction model from the previous chapter. By employing a structured higher-order data representation (i.e., sequences of hexagon blocks)

for trajectories, we are able to predict not only the future paths of individuals but also to deduce their potential points of contact by synchronizing the predicted trajectories in time. The proposed framework integrates both spatial and temporal information, thereby enabling accurate contact tracing and effective construction of a dynamic mobility network. An overview of this framework is presented in Figure 5.4.

In order to predict the mobility network, our framework integrates three key modules:

(i) **Trajectory Prediction Module**: This module uses our previously introduced TRAJLEARN model to forecast future trajectory segments (i.e., the next k hexagon blocks for each object) based on historical data.

(ii) **Contact Detection Module**: Based on the predicted trajectories, this module identifies contacts between objects by matching them to the same hexagon at corresponding time steps.

(iii) **Mobility Network Construction Module**: Finally, the detected contacts are used to construct time-varying network snapshots that represent the mobility network.

Trajectory Prediction Module

The trajectory prediction model, as described in Section 5.1, forms the foundational component of our framework. Briefly, each object's historical trajectory, represented as a sequence of hexagon blocks, is passed to our Transformer-based decoder architecture. The model then predicts a continuation of the sequence over a prediction horizon of k time steps. In our framework, this model outputs a probability distribution over the set of hexagon blocks for the next predicted block. By sampling from these distributions (or by selecting the most

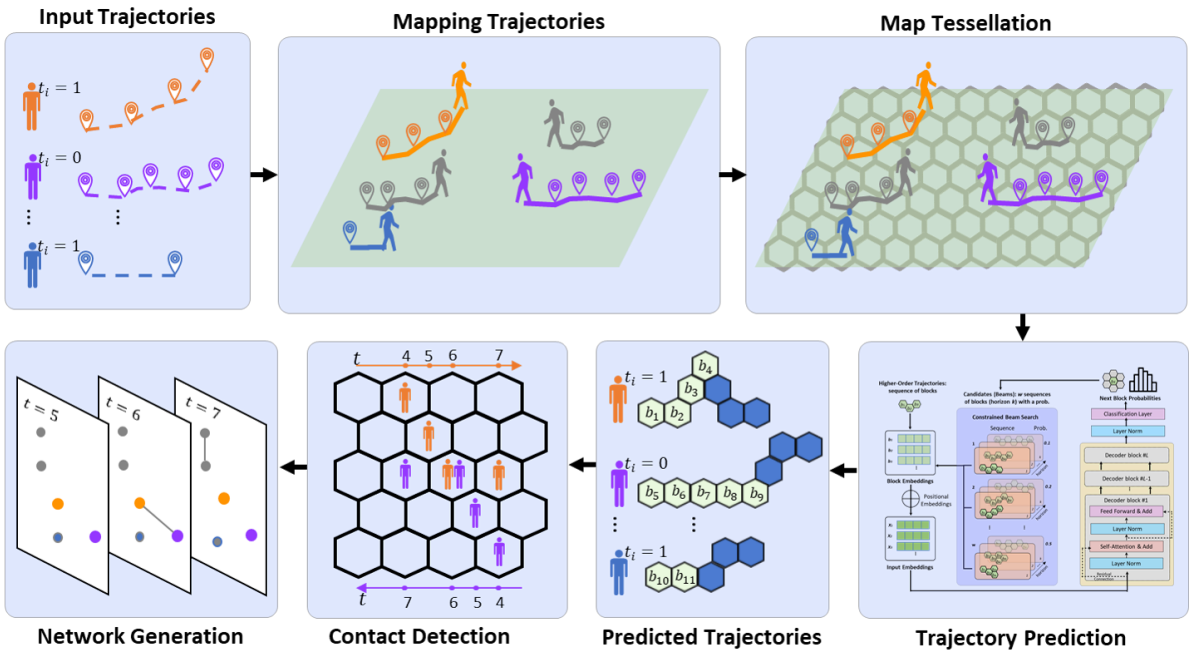


Figure 5.4: MobiNetForecast's High-Level Architecture.

probable block), the model produces a predicted trajectory:

$$\mathbf{T}^{u}_{\text{pred}} = \{b_{u,T+1}, b_{u,T+2}, \ldots, b_{u,T+k}\}.$$

The predictions are generated autoregressively, and the use of constrained beam search (detailed in Section 5.1) ensures that the sequence is spatially coherent.

Contact Detection Module

Contact prediction is the process of determining when and where interactions occur between moving objects based on their predicted trajectories. As described in previously, each object's trajectory is modeled as a sequence of hexagon blocks. In our framework, every hexagon block is associated not only with spatial information but also with a discrete time step. This temporal mapping is essential because contacts can only be determined if two or more objects

occupy the same spatial block at the same time.

Let each trajectory of an object u be represented in its higher-order form as:

$$\mathbf{T}_i^u = \{b_{i_1}, b_{i_2}, \ldots, b_{i_l}\},$$

where each block b_{i_j} corresponds to a spatial region derived from our hexagon-based tessellation of the map $\mathcal{M}$. To synchronize trajectories temporally, we associate each block with its time step. Assuming that object u's trajectory starts at time t_u, we assign the time mapping such that:

$$t_{u_j} = t_u + j - 1, \quad j = 1, 2, \ldots, l,$$

resulting in a representation as a sequence of block-time pairs:

$$\mathbf{T}^u = \{(b_{i_1}, t_{u_1}), (b_{i_2}, t_{u_2}), \ldots, (b_{i_l}, t_{u_l})\}.$$

This mapping enables us to handle trajectories that begin at different times and ensures that contact detection is performed consistently across objects.

Once every predicted trajectory is associated with its corresponding time steps, we can formalize the notion of contact. For any two objects u and v, a contact at time step t is defined to occur if they both occupy the same hexagon block at time t. Mathematically, we define a contact indicator function $C_{uv}(t)$ as follows:

$$C_{uv}(t) = \begin{cases} 1, & \text{if } b_{u_t} = b_{v_t}, \\ 0, & \text{otherwise,} \end{cases}$$

where b_{u_t} and b_{v_t} denote the hexagon blocks occupied by u and v at time t, respectively.

The set of all contacts at a given time step t is then collected as:

$$\mathcal{C}_t = \{(u, v) \mid C_{uv}(t) = 1 \text{ and } u \neq v\}.$$

This procedure is repeated over all time steps in the prediction horizon, ensuring that any simultaneous occupancy of the same hexagon by two or more objects is captured as a contact.

Mobility Network Construction Module

After identifying the contacts at each time step, the next step is to construct the corresponding mobility network. For each time step t, we create a network snapshot $\mathcal{G}_t = (\mathcal{N}_t, \mathcal{E}_t)$, where:

- The set of nodes, $\mathcal{N}_t$, includes all objects for which we have a predicted (or actual) position at time t. Formally:

$$\mathcal{N}_t = \{u \mid u \text{ appears in } \mathbf{T}^u \text{ at time } t\}.$$

- **Edges:** The edges $\mathcal{E}_t$ are derived directly from the contacts detected at time t. For every pair of objects (u, v) that form a contact at t, an edge is introduced:

$$\mathcal{E}_t = \{(u, v) \mid (u, v) \in \mathcal{C}_t\}.$$

By repeating this construction for every time step within the prediction horizon (e.g., from $t = T + 1$ to $t = T + k$), we can assemble a dynamic mobility network that evolves over time:

$$\mathcal{G} = \{\mathcal{G}_{T+1}, \mathcal{G}_{T+2}, \ldots, \mathcal{G}_{T+k}\}.$$

This sequence captures the evolution of contacts among the objects and forms the predicted mobility network over the forecast horizon.

5.2.2 Model Training

While the trajectory prediction module of our framework was detailed in Section 5.1, its training is equally critical to accurate mobility network forecasting. The training process consists of the following core steps:

Data Preparation

All trajectories from the datasets are converted into higher-order mobility flow data, where each trajectory is represented as a sequence of hexagon block IDs with associated time steps. Partial trajectories of varying lengths are generated to create training instances. For example, for a given trajectory sequence of length $l + k$, the first l blocks are used as input and the subsequent k blocks serve as the target sequence.

Loss Function and Teacher Forcing

We train our Transformer-based trajectory prediction model using a cross-entropy loss function that compares the predicted probability distribution against the one-hot encoded ground truth for each future block:

$$\mathcal{L} = -\sum_{t=1}^{k} \sum_{c=1}^{|\mathcal{B}|} y_{t,c} \log(\hat{y}_{t,c}),$$

where $|\mathcal{B}|$ denotes the number of hexagonal blocks, $y_{t,c}$ is the indicator function for the correct block, and $\hat{y}_{t,c}$ is the predicted probability for block c at time step t. The training employs teacher forcing, where at each time step the model's next input is replaced by the ground truth block from the training data. This method helps to stabilize training and accelerates convergence.

Incorporating Spatial Constraints

During training, the model is also conditioned on spatial continuity. Specifically, when generating the next block prediction, the model is guided to preferentially assign higher probabilities to blocks that are spatially adjacent to the current block. This is achieved implicitly through the data representation and explicitly via constrained beam search during inference (discussed in Section 5.1.3).

5.2.3 Computational Complexity

The overall computational complexity of our mobility network prediction framework is dictated by the following components:

Trajectory Prediction

The Transformer-based trajectory prediction module has a computational complexity of $O(N \cdot n^2 \cdot d)$ per sequence, where N is the number of layers, n is the sequence length, and d is the embedding dimension. This component is the most computationally expensive but benefits from GPU acceleration and highly optimized matrix multiplication routines provided by modern deep learning libraries.

Contact Detection

For M objects and a prediction horizon of k time steps, the naive approach of comparing each object with every other object yields a worst-case complexity of $O(M^2 \cdot k)$. In practice, spatial indexing and efficient data structures help reduce this complexity, as the number of potential contacts is substantially lower than M^2 in sparse environments.

Mobility Network Construction

Constructing the network snapshot for each time step is a linear pass over the set of detected contacts, with complexity proportional to the number of contacts C_t at time t. Overall, constructing all k snapshots has a complexity of $O\left(\sum_{t=T+1}^{T+k} C_t\right)$.

Beam Search

During inference, the constrained beam search algorithm has complexity $O(w \cdot \beta \cdot k)$, where w is the beam width, β is the average branching factor (i.e., the number of candidate blocks considered per time step), and k is the prediction horizon. Since w and β are typically small constants, the additional complexity added by beam search is modest compared to the trajectory prediction module.

By decoupling the trajectory prediction task from direct graph-based modeling of dynamic networks, our framework maintains scalability even for large-scale mobility networks with many objects and long prediction horizons.

Chapter 6

Experimental Evaluation

In this chapter, we present a thorough experimental evaluation of our proposed methods, examining both the trajectory prediction model (TrajLearn) and the integrated mobility network forecasting framework (MobiNetForecast). Our evaluation is designed to rigorously assess the performance, robustness, and scalability of our approach across multiple real-world datasets, spatial resolutions, and various parameter settings. We detail our experimental configuration, datasets, baseline comparisons, and evaluation metrics, and we provide comprehensive analyses—including parameter sensitivity studies, ablation experiments, and interpretability assessments—to demonstrate the strengths and limitations of our framework.

6.1 Evaluating TrajLearn

In this section, we focus on the evaluation of our trajectory prediction model, TrajLearn. This module is the foundation of our overall framework, as it forecasts future sequences of hexagon blocks that represent individual trajectories. We assess TrajLearn using

metrics such as ACCURACY@N and BLEU scores, comparing its performance against traditional and state-of-the-art baseline methods. Our experiments analyze the impact of various factors—including input sequence length, prediction horizon, and beam search configurations—on the model's predictive accuracy. The results not only illustrate the effectiveness of our Transformer-based architecture in capturing complex spatiotemporal dependencies but also highlight the benefits of our higher-order mobility representation and constrained inference techniques. We address five main research questions:

(Q1) Model Accuracy Performance. What is the accuracy performance of TRAJLEARN against sensible baselines?

(Q2) Parameter Sensitivity Analysis. How does the performance of our model vary with different input trajectory length l and prediction horizon k?

(Q3) Beam Search Analysis. How is the performance of our model affected by varying values of the beam width w?

(Q4) Map Resolution Analysis. How does the performance of the model change with varying levels of map tessellation?

(Q5) Ablation Study. How does beam search with constraint and model architecture hyperparameters impact the model's performance?

6.1.1 Experimental Configuration

Computational Environment. We conducted experiments on a server equipped with an NVIDIA RTX A6000 graphics card and 320GB of system memory. The model was developed in Python 3 and trained and deployed using the PyTorch 1.13 framework.

Map Tessellation and Resolutions. For tessellating a map, we utilized the `H3` geo-indexing system[2], which partitions the world into hexagonal cells of varying resolutions. We opted for `H3`'s resolutions 7, 8, and 9 for our experiments. Table 6.1 reports on the hexagon's *edge length* (km) and surface area (km^2).

Training Parameters. We train our model using the AdamW optimizer, with an initial learning rate of 5×10^{-3}, learning decay until reaching 5×10^{-7}, batch size of 64, and a dropout ratio of 0.1.

6.1.2 Datasets

In the experiments, we employ higher-order mobility flow data representations of three popular real-world trajectory datasets [18]. We briefly describe the semantics of each dataset (prefixed by "HO-" to indicate higher-order) and summarize their statistics in Table 6.2.

- **HO-PORTO** [103]: This dataset consists of recorded mobility traces of taxis operating in Porto, Portugal.
- **HO-ROME** [104]: This dataset consists of recorded mobility traces of taxis operating in Rome, Italy.
- **HO-GEOLIFE** [105, 106, 107]: This dataset consists of recorded mobility traces of individuals, covering a total distance of ∼1.2 million Km.

After transforming each dataset into higher-order mobility flows, we split the trajectory dataset, ordered by the start times of the trajectories, into training, validation, and test sets using a 70%, 10%, and 20% ratio, respectively. By partitioning the dataset into contiguous time intervals based on these start times, we eliminate the need to train and test over different

[2]H3 Geo-indexing Library. `https://h3geo.org/`

Resolution	Hex Edge Length (Km)	Hex Area (Km^2)
Hex@7	1.406	5.161
Hex@8	0.531	0.737
Hex@9	0.201	0.105
Hex@10	0.076	0.015

Table 6.1: Properties of hexagons of different resolutions.

Dataset	#Entities	Observation Period	Res	#Block	#Trajectory	Avg. Length
Ho-Rome	315	02/01/14 – 03/02/14	7	172	5,678	72.75
			8	875	5,837	260.17
			9	4,231	5,854	689.75
Ho-Porto	442	07/01/13 – 06/30/14	7	3,491	45,186	25.55
			8	12,998	397,367	24.07
			9	45,633	1,151,544	35.33
Ho-GeoLife	52	04/01/07 – 10/31/11	7	1,878	1,556	117.58
			8	6,360	1,830	219.28
			9	21,270	1,964	525.72

Table 6.2: Statistics of higher-order mobility flow datasets used in TrajLearn evaluation.

random splits and report variance. This method ensures our evaluation reflects how the data naturally progresses over time, avoiding any randomness from dividing the data arbitrarily.

Training Data. TrajLearn is trained using the Higher-order mobility datasets at various resolutions (7, 8, and 9), as discussed in Chapter 4. To ensure data quality, we excluded trajectories consisting of fewer than 15 hexagonal blocks. This exclusion was necessary because the experiment setup requires a minimum of $l = 10$ historical data points to forecast subsequent $k = 5$ points. Consequently, the number of trajectories in our training dataset differs from the original datasets. Table 6.2 presents the statistics for the training datasets across these resolutions.

6.1.3 Baseline Methods

The datasets discussed in this research are commonly used in trajectory prediction research, but comparing results across studies has become challenging due to several reasons: (i) some

prior research has introduced additional metadata, which was not initially present in the original dataset, such as POI data [108] or weather data [109]. These additions can favor certain models over others, irrespective of their architecture or training methods; (*ii*) different researchers employ various preprocessing pipelines, leading to significant differences in the generated data. This can involve excluding trajectories with GPS measurement errors [110, 109] or outliers [111, 109], which often represent the most challenging trajectories to predict. Therefore, to assess the performance of our model, we have selected prime models from existing literature that do not depend on additional meta information to serve as baselines. Each model represents a different approach to trajectory prediction.

MC [112]. A Markov Chain (MC) is a commonly used model for sequence prediction. It considers each location as a state and makes predictions based on a transition matrix between these states.

LSTM [113]. Long Short-Term Memory (LSTM) is a type of Recurrent Neural Network (RNN) specifically designed to capture long-term sequential dependencies, which are essential for mobility prediction tasks. In our experiments, we explored a range of hyperparameters, including: `embedding size` (64, 100, 200, 300, 500), `hidden size` (64, 128, 200), `number of layers` (1 or 2), and `dropout rate` of 0.2 for the middle layer in a 2-layer LSTM. We report the best results.

LSTM-ATTN [114]. LSTM with attention is a variant of LSTM that integrates an attention mechanism that allows the model to focus on specific parts of the sequence when making predictions. We applied the same parameter search space that we used for the LSTM.

GRU [115]. Gated Recurrent Units (GRU) is a variant of RNN designed to address the vanishing gradient problem of RNNs and can be applied to sequence prediction tasks. We used the same parameters range for this model as for the LSTM.

DeepMove [48]. DeepMove is a state-of-the-art method combining a multi-modal recurrent network and a historical attention mechanism to capture both spatial and temporal dependencies.

Flashback++ [116]. Flashback++ is a follow-up work by the authors of Flashback [50]. It is a system that relies on RNN and uses sparse semantic trajectory modeling to predict the next location by looking for similar trajectories in terms of temporal characteristics. A grid search was conducted on the hidden dimension values $\{10, 32, 64, 128, 256\}$, and the optimal value was selected based on the results.

Baselines Implementation. For the baselines, we implemented the MC, LSTM, LSTM-ATTN, and GRU models ourselves. For the DeepMove model, we used the implementation provided by [117][3], and for Flashback++ we relied on the implementation by [116][4], both with adjustments to support hexagonal-based spatial data in our experiments. Computational complexity, memory complexity, and the number of trainable parameters comparison of baseline models and TrajLearn is reported in Table 6.4.

6.1.4 Evaluation Metrics

To evaluate the performance of our model against the baselines, we consider and adopt the following well-established metrics for evaluating sequence prediction tasks of trained language models.

Accuracy@N [↑]. This metric assesses how often the correct sequence appears within the top-N ranked predictions made by the model. Given a set of sequences P in the test dataset, with a sequence denoted as s, the actual label of s as true(s), and the set of top N

[3] `https://github.com/LibCity/Bigscity-LibCity`
[4] `https://github.com/Pursue1221/FlashbackPlusPlus`

Dataset	Model	Resolution 7				Resolution 8				Resolution 9			
		Acc@1	Acc@3	Acc@5	BLEU	Acc@1	Acc@3	Acc@5	BLEU	Acc@1	Acc@3	Acc@5	BLEU
Ho-Porto	MC	0.2232	0.2460	0.2483	0.2443	0.2131	0.2390	0.2426	0.2359	0.2239	0.2511	0.2561	0.2490
	LSTM	0.4360	0.5070	0.5325	0.4836	0.3268	0.4435	0.4888	0.3778	0.3744	0.5168	0.5626	0.4132
	LSTM-ATTN	0.4383	0.5112	0.5387	0.4891	0.3233	0.4377	0.4781	0.3719	0.3675	0.5086	0.5556	0.4075
	GRU	0.3140	0.3741	0.4089	0.3581	0.2010	0.2771	0.3067	0.2411	0.2236	0.3074	0.3364	0.2588
	DeepMove*	0.0802	0.2038	0.3450	0.1516	0.1272	0.1994	0.2482	0.1725	0.2463	0.3139	0.3497	0.2858
	Flashback++*	0.4439	0.5015	0.5254	0.4929	0.3320	0.4599	0.4867	0.4066	0.3993	0.5316	0.5809	0.4314
	TrajLearn (ours)	**0.4507**	**0.5285**	**0.5648**	**0.5108**	**0.4244**	**0.5638**	**0.6138**	**0.4860**	**0.4785**	**0.6555**	**0.7111**	**0.5255**
	Improvement (%)	**1.53**	**5.38**	**7.50**	**3.63**	**27.84**	**22.59**	**26.11**	**19.53**	**19.81**	**23.30**	**22.42**	**21.80**
Ho-Rome	MC	0.0440	0.0591	0.0643	0.0685	0.1335	0.1482	0.1512	0.1504	0.1459	0.1699	0.1726	0.1686
	LSTM	0.2284	0.2919	0.3195	0.2566	0.3191	0.4152	0.4536	0.349	0.3664	0.5020	0.5527	0.3977
	LSTM-ATTN	0.2264	0.2892	0.3170	0.2550	0.3164	0.4133	0.4508	0.3462	0.3663	0.5016	0.5522	0.3972
	GRU	0.2132	0.2656	0.2934	0.2392	0.2244	0.2806	0.3064	0.2480	0.1636	0.2420	0.2801	0.1932
	DeepMove	0.2644	0.3594	0.3940	0.2966	0.3529	0.4140	0.4367	0.3815	OOM	OOM	OOM	OOM
	Flashback++	0.2448	0.3086	0.3386	0.2658	0.3364	0.4292	0.4666	0.3634	0.3860	0.5269	0.5821	0.4225
	TrajLearn (ours)	**0.2924**	**0.3700**	**0.4046**	**0.3279**	**0.3953**	**0.5149**	**0.5631**	**0.4317**	**0.4515**	**0.6085**	**0.6704**	**0.4886**
	Improvement (%)	**10.60**	**2.95**	**2.69**	**10.56**	**17.50**	**19.95**	**20.70**	**18.80**	**16.96**	**15.51**	**15.15**	**15.63**
Ho-GeoLife	MC	0.1045	0.1083	0.1093	0.1113	0.0743	0.0848	0.0858	0.0864	0.0665	0.0882	0.0899	0.0857
	LSTM	0.3837	0.4710	0.5009	0.3996	0.3632	0.4325	0.4631	0.3787	0.3909	0.4898	0.5203	0.4166
	LSTM-ATTN	0.4208	0.4840	0.5109	0.4376	0.4081	0.4742	0.5048	0.4271	0.3892	0.4809	0.5136	0.4142
	GRU	0.3051	0.3544	0.3995	0.3183	0.2187	0.3135	0.3656	0.2391	0.1070	0.1811	0.2437	0.1308
	DeepMove	0.4212	0.5928	0.6679	0.4765	0.3598	0.5136	0.6255	0.3778	OOM	OOM	OOM	OOM
	Flashback++	0.3907	0.4755	0.5072	0.4072	0.3911	0.4420	0.4885	0.3995	0.4144	0.5154	0.5301	0.4311
	TrajLearn (ours)	**0.6008**	**0.6683**	**0.7028**	**0.6235**	**0.5303**	**0.6082**	**0.6427**	**0.5565**	**0.4266**	**0.5247**	**0.5589**	**0.4545**
	Improvement (%)	**42.60**	**12.75**	**5.23**	**30.82**	**35.60**	**37.63**	**31.56**	**39.30**	**2.94**	**1.80**	**8.44**	**5.43**

Table 6.3: TrajLearn's prediction performance against five baseline methods, for varying evaluation metric and resolution, over three benchmark datasets. We fix input length $l = 10$ & prediction horizon $k = 5$. The bold/underlined numbers indicate the best/second best method, respectively. Improvement (%) reports the relative improvement of our model over the strongest baseline. (*Experiments with DeepMove on Ho-Porto and Flashback++ on {Ho-Porto, resolution 9} were conducted on 30,000 randomly sampled trajectories due to their limited efficiency and scalability on large datasets.)

predictions for s as $\text{Top}_N(s)$, it is defined as:

$$\text{Accuracy@N} = \frac{|\{s \in P \mid \text{true}(s) \in \text{Top}_N(s)\}|}{|P|} \tag{6.1}$$

This metric evaluates a model's ability to include the true label in its top N predictions, summarizing performance at various precision levels. We report Accuracy@1, Accuracy@3, and Accuracy@5.

BLEU score [↑]. This is a standard metric for sequence prediction tasks in NLP, which measures the quality of predicted sequences by comparing them with the ground truth sequences. In our context, we had to adapt the BLEU score to assess predicted trajectories.

Model	Time Complexity	Memory Complexity	# Parameters
MC	$O(1)$	$O(1)$	$\approx 10K$
LSTM	$O(T \cdot H^2)$	$O(T \cdot H)$	$\approx 6.1M$
LSTM-ATTN	$O(T \cdot H^2 + T^2 \cdot H)$	$O(T^2 \cdot H)$	$\approx 6.1M$
GRU	$O(T \cdot H^2)$	$O(T \cdot H)$	$\approx 5.1M$
DeepMove	$O(T \cdot H^2 + T^2 \cdot H)$	$O(T^2 \cdot H)$	$\approx 8.4M$
Flashback++	$O(T \cdot H^2)$	$O(T \cdot H)$	$\approx 6.5M$
TrajLearn	$O(T \cdot H^2 + T^2 \cdot H)$	$O(T^2 \cdot H)$	$\approx 7.3M$

Table 6.4: Computational complexity, memory complexity, and number of trainable parameters of baseline models and TrajLearn for a single inference. Here, T is the sequence length, and H is the hidden size. The trainable parameter count is reported for models trained on Ho-GeoLife, res=7.

We do so by quantifying the similarity between the predicted trajectories (sequences of blocks) and the actual trajectories by comparing the overlap of n-grams (or n-blocks) – contiguous sequences of n blocks along the trajectories – between them, incorporating a brevity penalty for overly short predicted trajectories. For a given trajectory, an n-block consists of a specific sequence of blocks (e.g., a turn at an intersection followed by a straight path and another turn), analogous to a combination of words forming a meaningful sentence. Formally, the BLEU score in our context is given by:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{T} w_n \log p_n\right)$$

where p_n is the ratio of number of n-blocks matches to the total number of n-blocks in the predicted trajectories, w_n are weights that sum to 1 ($\sum_{n=1}^{T} w_n = 1$), and T is the maximum order of n-block considered. The brevity penalty BP is defined as follows:

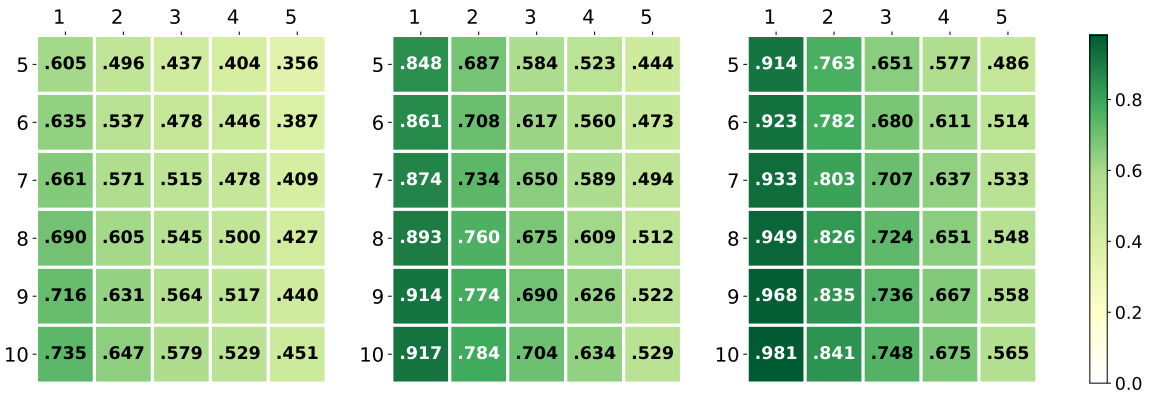
$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{6.2}$$

where c, r are the lengths of the predicted and ground truth sequences, respectively. In our experiments, we adhere to NLP best practices and consider up to 4-blocks ($T = 4$) and uniform weights.

6.1.5 Results and Discussion

(Q1) Model Accuracy Performance. We compare the performance of TrajLearn against the baselines employing different metrics and varying resolutions over three real-world trajectory datasets. We fix the input trajectory length ($l = 10$) and prediction horizon ($k = 5$). Table 6.3 shows the numerical results, where for each metric, the **bold** and underlined numbers correspond to the best and second-best performing model, respectively. A few key observations can be made: (i) TrajLearn demonstrates a remarkable performance by consistently securing one of the top two spots and, in all instances outperforming all competitors by a large margin (see % improvement), (ii) TrajLearn's accuracy is improving as the N of the Accuracy@N is increasing, where N represents the number of top predictions considered. This is due to the *teacher forcing* technique involved in the training stage (see Section 5.1.2), which provides supervision and corrects the prediction for any subsequent step, effectively allowing TrajLearn to learn.

Note that DeepMove encounters out-of-memory (OOM) issues with large datasets, specifically Ho-Porto at all resolutions, Ho-Rome at resolution 9, and Ho-GeoLife at resolution 9. To address this, we conducted experiments on randomly sampled trajectories (∼30,000) for each dataset, similar to the practice in Flashback++ [116]. However, for Ho-Rome and Ho-GeoLife at resolution 9, the sample sizes were too small to achieve satisfactory performance. The main reason for this is the method's approach to training the prediction model, which heavily depends on sparsely available POI check-in datasets.



(a) Accuracy@1 (b) Accuracy@3 (c) Accuracy@5

Figure 6.1: TrajLearn's accuracy for varying prediction horizon k (horizontal) & input length l (vertical) on Ho-Porto, res=7.

Upon aligning their methodology with our continuous path approach (hexagonal traversal), the volume of "check-ins" surges substantially, leading to OOM errors. Despite these OOM errors hindering some experiments, their impact on the overall conclusions is minimal, as the remaining results demonstrate our approach's generalizability and practicality. Our model preprocesses and trains on data batches, eliminating the need to load the entire dataset into memory. This reduces the memory footprint and enhances efficiency when processing large datasets.

(Q2) Parameter Sensitivity Analysis. In this experiment, we investigate the influence of varying input trajectory length ($5 \leq l \leq 10$) and prediction horizon ($1 \leq k \leq 5$) on the accuracy of TrajLearn at different precision levels. Figure 6.1 presents the results for Ho-Porto with resolution 7. A few key observations can be made: (i) the longer the trajectory history l, the higher the prediction accuracy; (ii) the shorter the prediction horizon k, the higher the accuracy prediction. These trends are logical, as predicting a far horizon with limited information as input becomes increasingly challenging. Similar to **(Q1)**, the accuracy is improving as the N of the Accuracy@N is increasing.

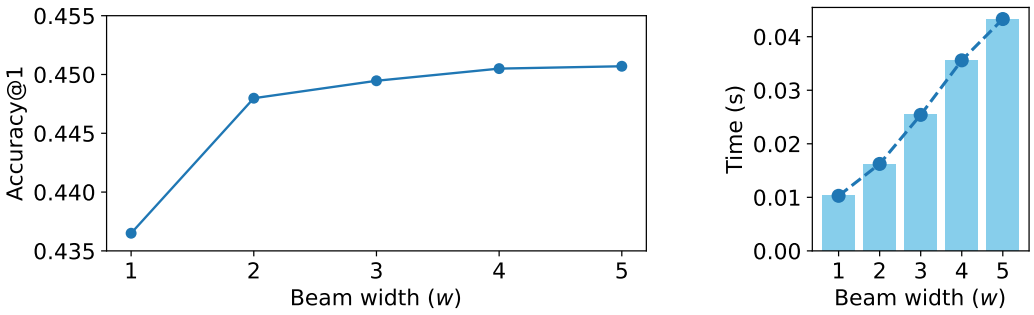


Figure 6.2: Impact of beam width w on TRAJLEARN's accuracy (left) and inference time (right) on HO-PORTO, res=7 and batch size of 64

.

(Q3) Beam Search Analysis. In this experiment, we investigate the tradeoff between *the accuracy performance* of TRAJLEARN and its *running cost during inference*, as a result of varying values of the beam width w ranging from 1 to 5. Figure 6.2 shows the results for ACCURACY@1 (left) and inference time averaged per batch (right) on HO-PORTO with resolution 7. The findings suggest that augmenting the beam width typically boosts the model's performance as anticipated. However, this enhancement becomes less apparent as the number of beams increases, implying diminishing returns. The running cost during inference increases linearly with the beam width, as expected. Consequently, we set the beam width to $w = 5$ (and not larger) for the rest of our experiments.

(Q4) Map Resolution Analysis. In this experiment, we investigate the impact of a map's resolution on TRAJLEARN's performance. Recall that a lower (higher) resolution means larger (smaller) hexagons. Depending on how the data is collected (e.g., vehicles or individuals) and the specific domain application, the different resolutions offer a trade-off between computational efficiency and accuracy. For illustration purposes, Figure 6.3 presents the results for varying resolutions on the HO-PORTO dataset. A few observations can be made: (i) the smaller the resolution, the slower the rate with which the accuracy decreases

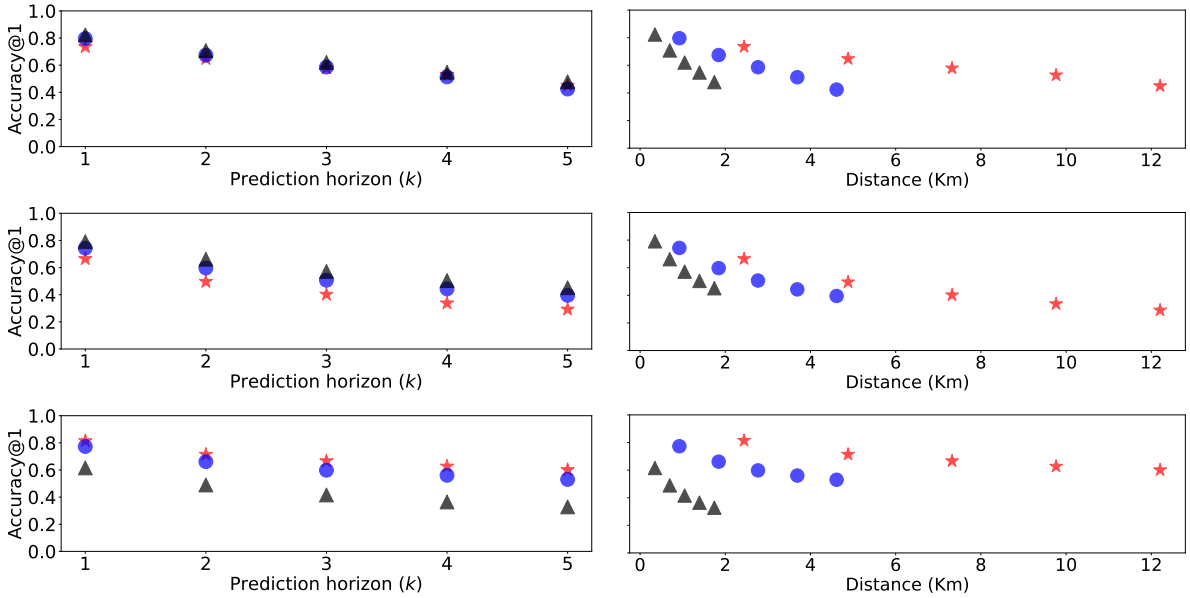


Figure 6.3: TRAJLEARN's accuracy for varying higher-order mobility flow resolutions (7: ⋆, 8: ●, 9: ▲) on HO-PORTO (top), HO-ROME (middle), HO-GEOLIFE (bottom). We report ACCURACY@1 as a factor of the prediction horizon k (left) and the actual distance traveled (right).

as the prediction horizon increases; (ii) the smaller the resolution, the slower the rate with which the accuracy decreases, as the distance traveled increases.

(Q5) Ablation Study. In this experiment, we investigate the impact of certain components of TRAJLEARN's neural architecture. We selectively remove the beam search module and report the *accuracy performance change*. Table 6.5 shows the results, which indicate that the accuracy drops when removing the beam search. In addition, we perform a hyperparameter analysis to gauge the impact of various parameters on accuracy: *embedding dimension*, *number of decoder layers*, and *number of attention heads*. These tests were carried out using the HO-GEOLIFE dataset at resolution 7. Figure 6.4 presents the results. From our observations, TRAJLEARN's performance benefits from an increase in the number of layers, as it allows for a more comprehensive processing of dependencies. Likewise, enhancing the

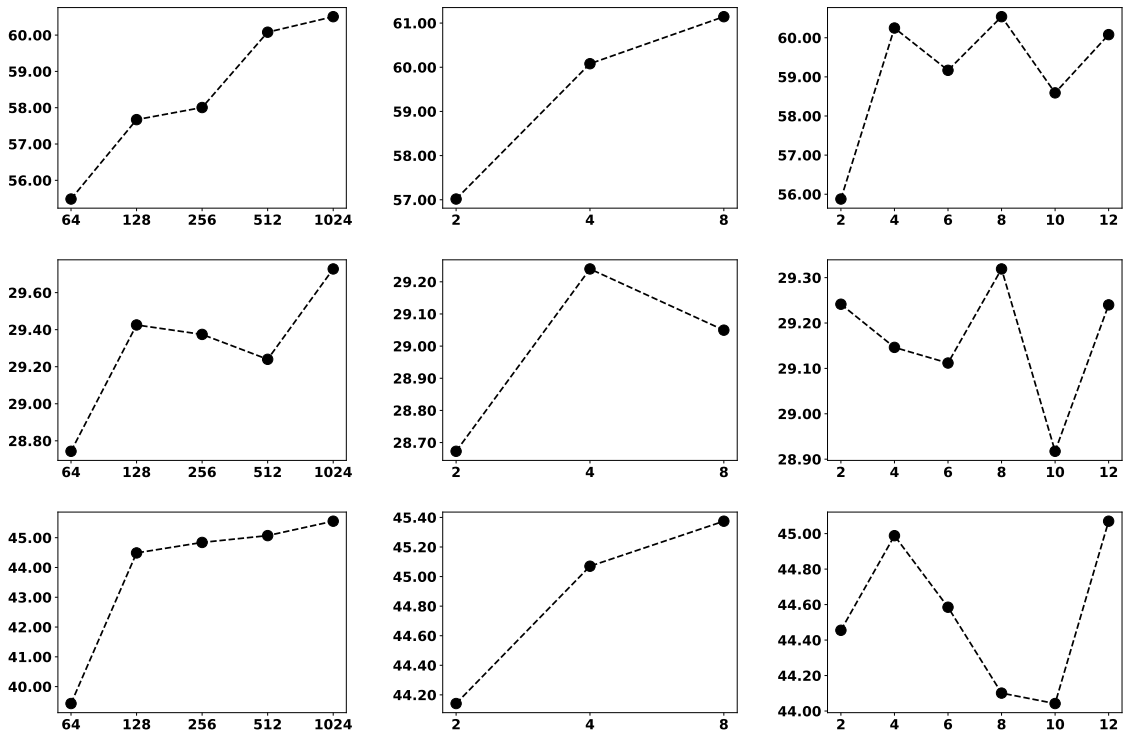


Figure 6.4: TrajLearn's Accuracy@1 over datasets Ho-GeoLife (top), Ho-Rome (middle), and Ho-Porto (bottom) with resolution 7 for varying embedding vector size (left), number of attention heads (middle), and number of Transformer layers (right).

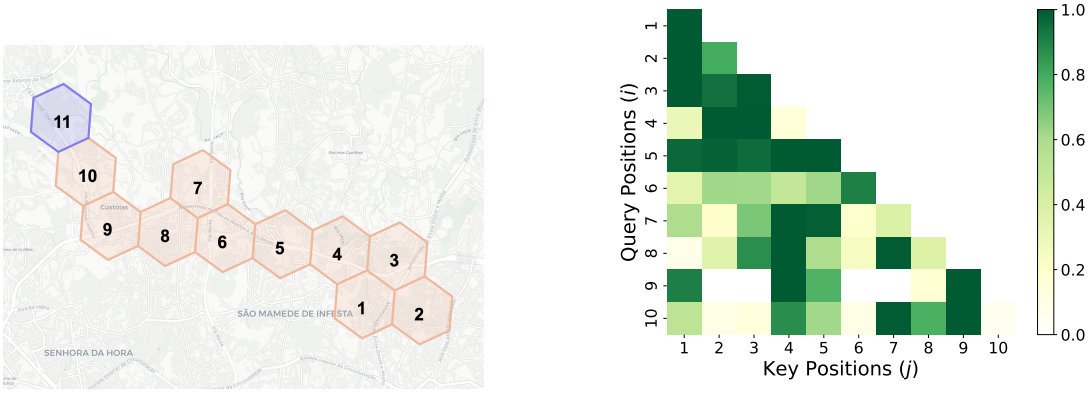
number of attention heads results in a better outcome. Additionally, increasing the embedding dimension enhances performance up to a certain threshold, likely due to the constrained input length.

Discussion. Beyond addressing the five experimental evaluation questions, we now delve deeper into the underlying reasons for TrajLearn's performance improvements. Unlike

Dataset	Accuracy@1	Accuracy@1 W/O Beam	Change (%)
Ho-Porto@7	0.4507	0.4365	-3.15
Ho-Porto@8	0.4244	0.4064	-4.24
Ho-Porto@9	0.4785	0.4677	-2.26

Table 6.5: Impact of removing the beam search on TrajLearn's accuracy.

conventional autoregressive approaches, such as RNN-based models (e.g., LSTM and GRU), which often suffer from vanishing gradients and limited long-range context, TrajLearn leverages a transformer-based architecture that models the joint distribution of future trajectories, thereby effectively capturing complex higher-order mobility flows. The inherent self-attention mechanism of transformers enables the model to learn intricate spatial-temporal dependencies over extended sequences, seamlessly integrating cues from both local (recent) contexts and long-range dependencies that are critical for accurate prediction, thus yielding more informed and precise outcomes. Furthermore, the model's generative formulation facilitates the exploration of multiple plausible future paths in a coherent and probabilistic manner, enhancing its ability to account for the inherent uncertainty and multimodality in movement data. Moreover, although hexagonal tessellation is common across all evaluated models, its superior ability to capture subtle spatial dependencies—thanks to its uniform neighborhood structure—provides a richer and more consistent representation of the movement space. This enhanced spatial representation is particularly well-exploited by a powerful model like the transformer, which can discern nuanced relationships both between adjacent spatial units and across larger regions. Additionally, the employment of teacher forcing during training guides the model with ground-truth sequences, thereby promoting more robust learning of complex spatial-temporal patterns. Finally, the integration of a constrained beam search mechanism during inference enables TrajLearn to explore multiple plausible future paths in a coherent and probabilistic manner, ensuring that predicted trajectories adhere to spatial continuity constraints. This approach effectively accounts for the inherent uncertainty and multimodality in movement data, ultimately generating realistic and feasible paths. Collectively, these design choices empower TrajLearn to handle complex higher-order mobility flows, leading to performance improvements of up to approximately 40% over baseline methods, as demonstrated in our experiments.



(a) An example trajectory. (b) Aggregated heatmap.

Figure 6.5: An example that shows how TrajLearn predicts the future path of a trajectory. (**a**) Given as input the sequence of hexagons 1-10, the model predicts the hexagon 11. (**b**) The heatmap representing the aggregated attention weights across all 8 heads.

6.1.6 Interpretability Study

Understanding how deep models make predictions is crucial, especially when the models are used in safety-critical applications or systems with significant real-world impact. To gain insights into how TrajLearn makes decisions, we perform an interpretability study.

Given that TrajLearn is a Transformer-based architecture, in this section, we aim to analyze the predictions using weights of self-attention for interpretation [118], specifically, we use self-attention weights in the last decoder block. The attention mechanism assigns a weight to each input token (hexagon block) indicating its relevance to the prediction of the output token. In causal self-attention, each token only attends to tokens in its left context, making it possible to trace the influence of past trajectory segments on the prediction of the next block. By examining the attention weights, we can see which tokens in the input sequence were deemed most *relevant* or *important* when predicting a particular output token. This can provide insights into how the model understands and uses context.

Figure 6.6 presents heatmaps for all 8 attention heads when predicting the 11th hexagon

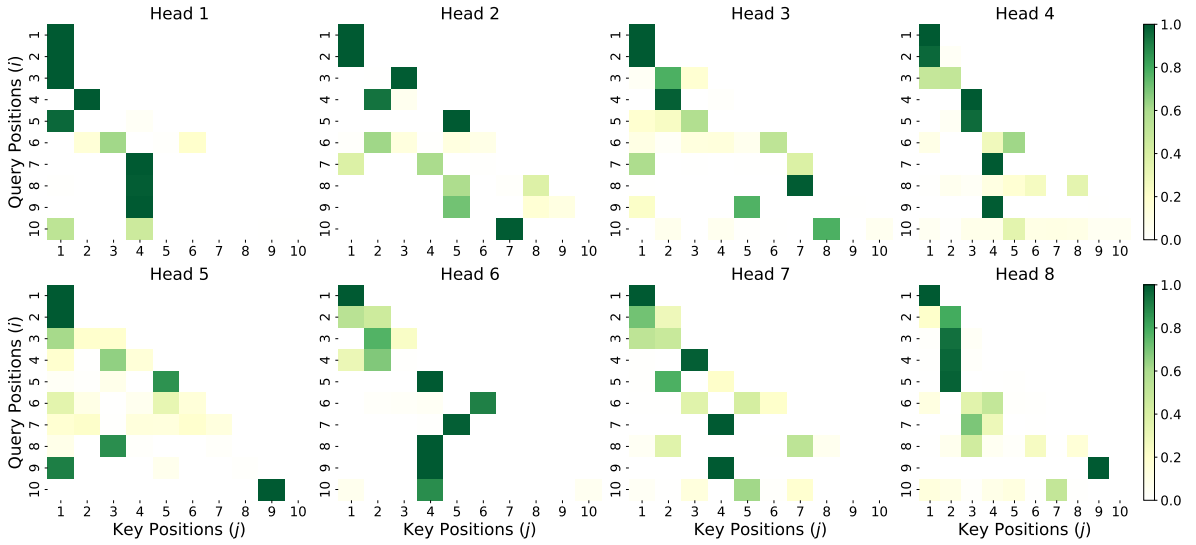


Figure 6.6: The heatmaps of the attention weights of all 8 attention heads when predicting hexagon 11 from Figure 6.5a.

in a trajectory, with the input consisting of hexagons 1 through 10. Each heatmap shows the distribution of attention scores (softmax of the multiplication of *query* and *key* in the self-attention layer in the last decoder) across the input sequence. Notably, the variation across different heads indicates that the model captures multiple types of relationships: some heads may focus on immediate context (e.g., the last few blocks), while others may capture longer-range dependencies.

To obtain a consolidated view, we aggregate the attention weights across all heads using the max function. By taking the maximum across heads, any strong signal that *any* head assigns is preserved, yielding a consolidated view to pinpoint exactly those input tokens that exert the strongest influence on each prediction, revealing which features (e.g., local versus long-range transitions) are most critical to the model's trajectory forecasts. Figure 6.5 shows the aggregated attention weights, which highlight the most influential positions in the sequence for the prediction task. For instance, certain positions—such as hexagon 4—consistently receive high attention, underscoring their importance in defining the movement pattern. As

depicted in Figure 6.5a, hexagon 4 represents a turn on the highway that significantly could influence the prediction of the next block (11) based on recent hexagons. Moreover, there is evidence of significant attention between non-adjacent positions, indicating that the model identified some long-distance relationships. These visualizations help in validating that the model leverages both local and global spatial-temporal patterns effectively.

6.2 Evaluating MobiNetForecast

This section is dedicated to the evaluation of MobiNetForecast, our comprehensive framework for forecasting mobility networks. This evaluation builds on the trajectory predictions generated by TrajLearn to detect contacts between moving objects and to construct dynamic network snapshots over time. We measure the performance of the mobility network predictions using metrics such as precision, recall, and F1-Score. Our experiments compare the performance of MobiNetForecast against baseline models for temporal link prediction, and we examine how map resolution influence overall performance. The evaluation demonstrates that our trajectory-based approach to network forecasting effectively captures the evolution of interactions in dynamic environments, outperforming conventional methods that rely solely on historical interaction data. We address three key research questions:

(Q1) Framework Performance. How does MobiNetForecast compare against established baseline methods in predicting dynamic networks?

(Q2) Model Performance. How does TrajLearn used in MobiNetForecast compare against using other trajectory prediction methods in predicting dynamic networks?

Table 6.6: Statistics of higher-order mobility flow datasets used in MobiNetForecast evaluation.

Dataset	# Objects	# Trajectories	# Contacts@10
GeoLife	182	17,621	124K
SFCO-3K	3000	90,000	1.4M

(Q3) Map Resolution Analysis. How does varying the resolution of map tessellation (i.e., the size of hexagons) impact the accuracy of contact detection and network construction?

6.2.1 Experimental Configuration

Computational Environment. The experiments were conducted on a high-performance server equipped with an NVIDIA RTX A6000 graphics card. All models were implemented in Python 3 using the PyTorch 2.3 framework.

Map Tessellation and Resolutions. We tessellated the observation area using the `H3` geo-indexing system. Our experiments focused on H3 resolutions 8, 9, and 10. The properties of these resolutions are listed in Table 6.1.

Training Parameters. The trajectory prediction model within MobiNetForecast was trained with the AdamW optimizer, an initial learning rate of 1×10^{-4} decayed to 5×10^{-9}, a batch size of 256, and no dropout.

6.2.2 Datasets

We evaluated our framework on a real-world and a large-scale synthetic trajectory datasets. The semantics of each dataset and their respective statistics are detailed below and summarized in Table 6.6.

- **GeoLife** [105, 106, 107]: This dataset contains real-world mobility traces from

individuals covering approximately 1.2 million kilometers.

- **SFCO-3K** [119]: A simulated dataset representing trajectories of 3,000 agents in the San Francisco region over a period of 15 months. Due to scale of the dataset being very large, only the first month data is used in this evaluation.

After preprocessing, including transforming trajectories into hexagon sequences with discrete time steps and generating contacts, we divided each dataset into training (67.5%), validation (7.5%), and test (25%) sets while preserving temporal order.

6.2.3 Baseline Methods

To benchmark the performance of MobiNetForecast, we compared it against a range of baselines that encompass both trajectory prediction-based and temporal link prediction-based methods:

Trajectory Prediction-Based Models: These models focus on forecasting the future trajectory of individuals based on their historical movement patterns. They serve as direct comparisons to our trajectory prediction component.

- **MC** [112]: See Section 6.1.3.
- **LSTM** [113]: See Section 6.1.3.
- **LSTM-ATTN** [114]: See Section 6.1.3.
- **GRU** [115]: See Section 6.1.3.
- **Flashback++** [120]: See Section 6.1.3.

Temporal Link Prediction-Based Models: These models predict future interactions within the mobility network by analyzing historical spatiotemporal contact data, without explicitly modeling individual trajectories.

- **EvolveGCN** [10]: is a framework that dynamically updates Graph Neural Networks (GNNs) using a Recurrent Neural Network (RNN) to capture the evolving nature of these networks.
- **ROLAND** [121]: leverages GNNs for dynamic graph learning, focusing on modeling the temporal evolution of the network.
- **DyTed** [122]: is a representation learning framework designed for discrete-time dynamic graphs that separates time-invariant and time-varying node representations through a temporal-clips contrastive learning task. In this evaluation, we use the LSTMGCN version of DyTed.
- **AGCRN** [123]: builds on the standard GCN by incorporating two specialized modules. The first is a node-adaptive parameter learning module that factorizes the parameters to produce node-specific adjustments. The second is a data-adaptive graph generation module that automatically uncovers the interdependencies among different traffic series.

Baselines Implementation. Trajectory prediction-based models are implemented as in Section 6.1.3, and then the contact prediction module and mobility network construction module as defined in MobiNetForecast are used to generate the mobility network. For the ROLAND model, the official implementation provided by [121][5], and for DyTed the official implementation provided by [122][6] is used. In case of AGCRN and EvolveGCN,

[5] https://github.com/snap-stanford/roland
[6] https://github.com/Kaike-Zhang/DyTed

the implementation provided by [124][7] is used.

6.2.4 Evaluation Metrics

The following metrics are used to evaluate the performance of our mobility network prediction framework:

Recall [↑]. Measures the proportion of actual contacts (true positives) correctly predicted. It assesses the model's ability to capture relevant interactions within the mobility network.

$$\frac{|\{(u,v) \mid \hat{\mathbf{A}}_t(u,v) = 1 \text{ and } \mathbf{A}_t(u,v) = 1\}|}{|\{(u,v) \mid \hat{\mathbf{A}}_t(u,v) = 1\}|},$$

where $\mathbf{A}_t$ is the ground truth adjacency matrix of the mobility network and $\hat{\mathbf{A}}_t$ is the adjacency matrix of the predicted mobility network.

Precision [↑]. Quantifies the proportion of correctly predicted contacts among all predicted contacts. High precision indicates that the model's predictions are accurate, minimizing false positives.

$$\frac{|\{(u,v) \mid \hat{\mathbf{A}}_t(u,v) = 1 \text{ and } \mathbf{A}_t(u,v) = 1\}|}{|\{(u,v) \mid \mathbf{A}_t(u,v) = 1\}|}$$

F1-Score [↑]. The harmonic mean of precision and recall, providing a balanced measure of the model's accuracy, especially in scenarios with imbalanced class distributions. It captures both the completeness and the exactness of the contact predictions.

$$\text{F1} \;=\; 2 \;\times\; \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

[7] `https://github.com/pyg-team/pytorch_geometric`

6.2.5 Results and Discussion

(Q1) Framework Performance. Across both real-world and synthetic datasets, MobiNetForecast consistently outperforms the strongest baselines in all three evaluation metrics—precision, recall, and F1-score—at every tested map resolution (8, 9, and 10). For example, on the San Francisco dataset at resolution 8, the best trajectory-prediction baseline (Flashback++) achieved an F1 of 0.160, and the best graph-prediction baseline (ROLAND) reached an F1 of 0.158, whereas MobiNetForecast attained an F1 of 0.186, marking an +16.4% improvement over Flashback++ and +17.7% over ROLAND (Table 6.7). Similar gains are observed at higher resolutions: at resolution 10, MobiNetForecast achieves an F1 of 0.068 (versus Flashback++'s 0.059 and ROLAND's 0.053), representing improvements of +15.3% and +28.3%, respectively. On GeoLife, the gains are even more pronounced—at resolution 8, the framework's F1 of 0.315 surpasses Flashback++ (0.261) and AGCRN (0.194) by +20.7% and +62.4%, respectively. These results demonstrate that leveraging trajectory predictions to drive contact inference and network construction yields markedly stronger performance than methods relying solely on historical contacts or pure trajectory forecasting.

(Q2) Model Performance. To isolate the impact of our chosen trajectory predictor, we compared MobiNetForecast when using TrajLearn against the same framework powered by alternative trajectory prediction models (MC, LSTM, LSTM-ATTN, GRU, Flashback++). As summarized in Table 6.8, across both datasets and all resolutions, replacing TrajLearn with any other predictor yields lower recall, precision, and F1. For instance, on San Francisco at resolution 8, TrajLearn enables recall of 0.289 and precision of 0.137, whereas Flashback++ yields recall 0.261 and precision 0.115—thus, TrajLearn contributes an +8.4% gain in recall and +16.1% gain in precision over the strongest trajectory-

Table 6.7: MobiNetForecast's prediction performance against nine baseline methods across two datasets and three map resolutions. The bold numbers indicate the best performing method, while underlined numbers represent the second best.

Dataset	Model	Resolution 8			Resolution 9			Resolution 10		
		Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
San Francisco	MC	0.023	0.046	0.031	0.015	0.083	0.025	0.008	0.063	0.014
	LSTM	0.097	0.231	0.136	0.065	0.187	0.096	0.026	0.146	0.044
	LSTM-ATTN	0.103	0.245	0.145	0.072	0.193	0.105	0.030	0.158	0.044
	GRU	0.118	0.240	0.158	0.075	0.193	0.108	0.032	0.162	0.053
	Flashback++	0.115	0.261	0.160	0.073	0.200	0.100	0.035	0.187	0.059
	EvolveGCN	0.0002	0.247	0.0003	0.0002	0.187	0.0003	0.0001	0.119	0.0002
	Roland	0.0006	0.277	0.001	0.0006	0.150	0.001	0.0004	0.127	0.0008
	DyTed	0.001	0.271	0.002	0.001	0.220	0.002	0.0008	0.181	0.002
	AGCRN	0.0006	0.193	0.001	0.0005	0.140	0.0009	0.0004	0.130	0.0008
	MobiNetForecast	**0.137**	**0.289**	**0.186**	**0.086**	**0.226**	**0.125**	**0.040**	**0.208**	**0.068**
GeoLife	MC	0.013	0.049	0.020	0.010	0.051	0.023	0.008	0.047	0.013
	LSTM	0.143	0.482	0.221	0.132	0.514	0.210	0.137	0.506	0.216
	LSTM-ATTN	0.151	0.546	0.237	0.143	0.531	0.225	0.141	0.523	0.222
	GRU	0.157	0.510	0.240	0.153	0.549	0.239	0.141	0.572	0.227
	Flashback++	0.171	0.553	0.261	0.165	0.586	0.258	0.160	0.535	0.250
	EvolveGCN	0.039	0.207	0.066	0.054	0.214	0.086	0.068	0.221	0.104
	Roland	0.084	0.456	0.143	0.127	0.470	0.201	0.135	0.480	0.210
	DyTed	0.114	0.513	0.187	0.163	0.528	0.250	0.154	0.518	0.238
	AGCRN	0.125	0.437	0.194	0.121	0.455	0.191	0.125	0.479	0.199
	MobiNetForecast	**0.206**	**0.667**	**0.315**	**0.199**	**0.706**	**0.311**	**0.189**	**0.659**	**0.294**

prediction baseline. On GeoLife at resolution 9, TrajLearn drives a recall of 0.706 (versus Flashback++'s 0.586) and precision of 0.199 (versus 0.165), translating to improvements of +20.5% and +20.6%, respectively. These improvements confirm that the superior sequential modeling of TrajLearn is a key factor in the overall success of the framework.

(Q3) Map Resolution Analysis. We examined the influence of hexagon size (resolutions 8–10) on contact-prediction performance. Figure 6.7 presents these results. On the San Francisco data, finer tessellation (higher resolution) leads to decreases in both precision and recall: recall drops from 0.289 (res 8) to 0.226 (res 9) and 0.208 (res 10), while precision falls from 0.137 to 0.086 and 0.040. This trend reflects the increasing difficulty of predicting exact co-location in smaller spatial cells. Conversely, on GeoLife—with its denser trajectories—the optimal resolution is 9, yielding the highest recall (0.706), marginally outperforming resolution

Table 6.8: Improvement of MobiNetForecast over the best trajectory-prediction and graph-prediction baselines.

Metric	Dataset	Resolution	Best TrajPred	Best GraphPred	MobiNetF.	%Δ vs. Traj	%Δ vs. Graph
Recall	San Francisco	8	0.261 (Flashback++)	0.277 (ROLAND)	0.289	+8.4 %	+4.3 %
		9	0.200 (Flashback++)	0.220 (DyTed)	0.226	+13.0 %	+2.7 %
		10	0.187 (Flashback++)	0.181 (DyTed)	0.208	+11.2 %	+14.9 %
	GeoLife	8	0.553 (Flashback++)	0.513 (DyTed)	0.667	+20.6 %	+30.0 %
		9	0.586 (Flashback++)	0.528 (DyTed)	0.706	+20.5 %	+33.7 %
		10	0.572 (GRU)	0.518 (DyTed)	0.659	+15.2 %	+27.3 %
Precision	San Francisco	8	0.118 (GRU)	0.001 (DyTed)	0.137	+16.1 %	+ 13 600 %
		9	0.075 (GRU)	0.001 (DyTed)	0.086	+14.7 %	+ 8 500 %
		10	0.035 (Flashback++)	0.0008 (DyTed)	0.040	+14.3 %	+ 4 900 %
	GeoLife	8	0.171 (Flashback++)	0.125 (AGCRN)	0.206	+20.5 %	+64.8 %
		9	0.165 (Flashback++)	0.163 (DyTed)	0.199	+20.6 %	+22.1 %
		10	0.160 (Flashback++)	0.154 (DyTed)	0.189	+18.1 %	+22.7 %
F1-Score	San Francisco	8	0.160 (Flashback++)	0.002 (DyTed)	0.186	+16.3 %	+ 9 200 %
		9	0.108 (GRU)	0.002 (DyTed)	0.125	+15.7 %	+ 6 150 %
		10	0.059 (Flashback++)	0.002 (DyTed)	0.068	+15.3 %	+ 2 800 %
	GeoLife	8	0.261 (Flashback++)	0.194 (AGCRN)	0.315	+20.7 %	+62.4 %
		9	0.258 (Flashback++)	0.250 (DyTed)	0.311	+20.5 %	+24.4 %
		10	0.250 (Flashback++)	0.238 (DyTed)	0.294	+17.6 %	+23.5 %

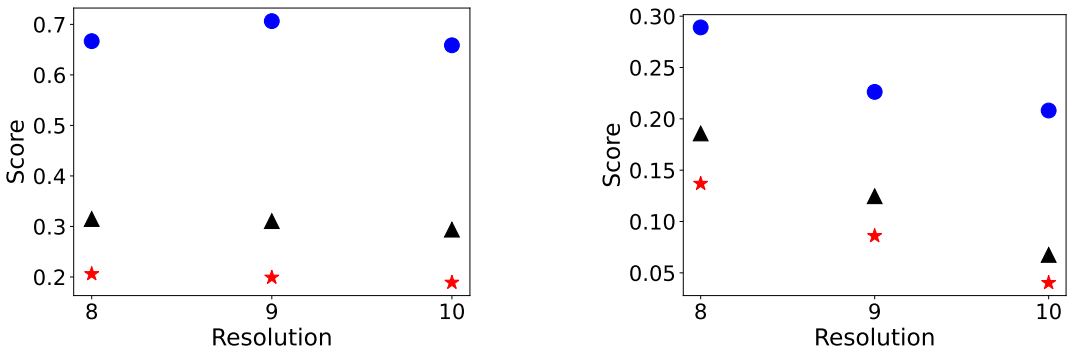


Figure 6.7: MobiNetForecast performance metrics (precision: ⋆, recall: ●, F1-Score: ▲) for H3 resolutions 8, 9, and 10 on GeoLife (left) and SFCO (right) datasets.

8 (0.667) and resolution 10 (0.659). Precision and F1 similarly peak at mid-level resolution. Therefore, while coarse resolutions (larger hexagons) ease contact detection but sacrifice granularity, and very fine resolutions challenge prediction accuracy, an intermediate resolution (e.g., 9) often offers the best balance of specificity and predictability in densely sampled datasets.

Discussion. Our experiments validate that

- *Trajectory-driven forecasting* outperforms purely contact-based or purely trajectory-based methods. By first predicting individual movements with a high-fidelity sequence model and then inferring contacts, MobiNetForecast captures both spatial and temporal dynamics more effectively than baselines that analyze contacts directly or treat trajectories in isolation.

- The transformer-based TrajLearn notably elevates the performance of the overall framework. Its self-attention mechanism and constrained beam-search ensure both long-range dependency capture and spatial continuity, driving substantial gains over RNN-based predictors.

- Tessellation resolution critically affects contact-prediction outcomes. Larger hexagons smooth over minor location errors but reduce precision, whereas very small hexagons demand more exact predictions. Intermediate resolutions strike a balance, especially in high-density settings, underscoring the importance of selecting a resolution that aligns with data sampling density and the spatial scale of the phenomenon of interest.

Taken together, these findings validate our design of combining state-of-the-art trajectory modeling with structured contact inference, yielding a robust and flexible framework for forecasting dynamic mobility networks.

Chapter 7

Additional Enhancements

In this chapter, we describe a series of additional enhancements to our framework that further improve its practical applicability and predictive performance. These enhancements include novel modules such as the mapping of predicted hexagon sequences back to GPS points for real-world spatial interpretation, and the development of hierarchical maps that adjust spatial resolution based on local data density. The hexagon-to-GPS mapping enables our predictions to be directly interpreted in geographic coordinates, facilitating integration with navigation systems and urban planning tools. Meanwhile, the hierarchical mapping approach allows the model to capture fine-grained spatial details in high-density areas while maintaining efficiency in regions with sparse activity. Together, these enhancements address key challenges in applying our prediction framework to practical, real-world scenarios. In addition, we provide flexible contact definitions for the mobility network prediction problem and their real-world application scenarios.

7.1 Mapping Predicted Hexagons to GPS Points

Building upon the solution where future trajectories are predicted as sequences of hexagons on a tessellated map, we present results to map these predicted hexagons back to corresponding GPS points. This conversion can be useful for real-world applications, such as mapping predicted routes into the street map or performing further spatial analysis. This section explains the motivation behind such a conversion, describes our approach in detail, and presents a case study using real-world data.

7.1.1 Motivation

The use of hexagon-based representations confers several advantages for modeling and prediction. Hexagonal tessellation reduces sparsity, enables uniform spatial partitioning, and allows us to exploit methodologies from natural language processing by treating each hexagon as a token in a sequence. Despite these benefits, the hexagon representation is an abstraction that is not immediately interpretable in terms of latitude and longitude. For some real-world applications—including route planning, traffic management, and urban analytics—users might require predictions expressed as GPS coordinates.

Mapping predicted hexagons back to GPS points serves several purposes:

- **Interpretability:** By converting hexagon IDs to GPS coordinates, decision-makers can overlay predictions on road maps and satellite imagery, facilitating practical applications such as infrastructure planning and emergency response.
- **Integration:** Some downstream systems, such as navigation applications, require GPS inputs. Converting hexagon predictions ensures seamless integration into such systems.
- **Spatial Accuracy:** Although the hexagon representation efficiently captures spatial-

temporal patterns, the conversion process, especially when refined with map matching, ensures that predicted locations correspond closely to real-world features (e.g., roads, intersections).

This mapping is therefore essential not only for visualization but also for enhancing the practical impact of our mobility network prediction approach.

7.1.2 Approach

In the main results, future trajectories are predicted as sequences of hexagons on a tessellated map, where each hexagon represents a discrete spatial region. Using hexagons for spatial tessellation is advantageous due to their geometric efficiency and equidistant neighbors, which facilitate smoother and more computationally efficient predictions. Our approach for converting predicted hexagon sequences to GPS points is both straightforward and efficient. To convert these hexagon-based predictions into GPS points, we map each predicted hexagon to its centroid, denoted as (x_h, y_h). The centroid of a regular hexagon is the point that minimizes the average distance to all other points within the hexagon, making it a computationally efficient and reasonable approximation for the predicted GPS location. The process can be summarized in the following steps:

1. **Hexagon Tessellation:** We partition the study area into a regular grid of hexagons using the H3 geo-indexing system. Each hexagon is a polygon with a known shape, size, and location.

2. **Computing the Centroid:** For a regular hexagon, the centroid is computed as the average of its vertex coordinates. If a hexagon has vertices $(x_1, y_1), (x_2, y_2), \ldots, (x_6, y_6)$,

its centroid (x_h, y_h) is given by:

$$(x_h, y_h) = \left(\frac{1}{6} \sum_{i=1}^{6} x_i,\ \frac{1}{6} \sum_{i=1}^{6} y_i \right).$$

This centroid represents the "center" of the hexagon, and it is a computationally efficient approximation for the geographic location of that block.

3. **Centroid Mapping:** Each predicted hexagon in the trajectory sequence is mapped to its corresponding centroid, thereby producing a series of GPS coordinates. These coordinates form the predicted trajectory in geographical terms.

4. **Refinement via Map Matching (Optional):** For applications demanding higher precision, we can apply map-matching techniques. For example, using routing engines such as OSRM [125], the centroids can be snapped to the nearest road segments. This additional step helps to correct minor spatial misalignments due to the abstraction and produces outputs that are tightly coupled with the actual road network.

Figure 7.1 visually summarizes this conversion process. It shows the predicted hexagon sequence on the tessellated map and the corresponding centroid-based GPS trajectory.

For applications that demand greater precision, especially when the hexagons are large or when finer spatial details are needed, a more refined approach can be applied. One such approach is to employ map matching techniques, which align GPS trajectories to the nearest road network or paths [125]. In this context, map matching can adjust the centroid-based predictions by snapping them to the most likely path within or around the predicted hexagon, considering the road network and the historical trajectory of the moving entity. This can significantly improve the accuracy of the mapped GPS points, particularly in urban environments with dense road networks.

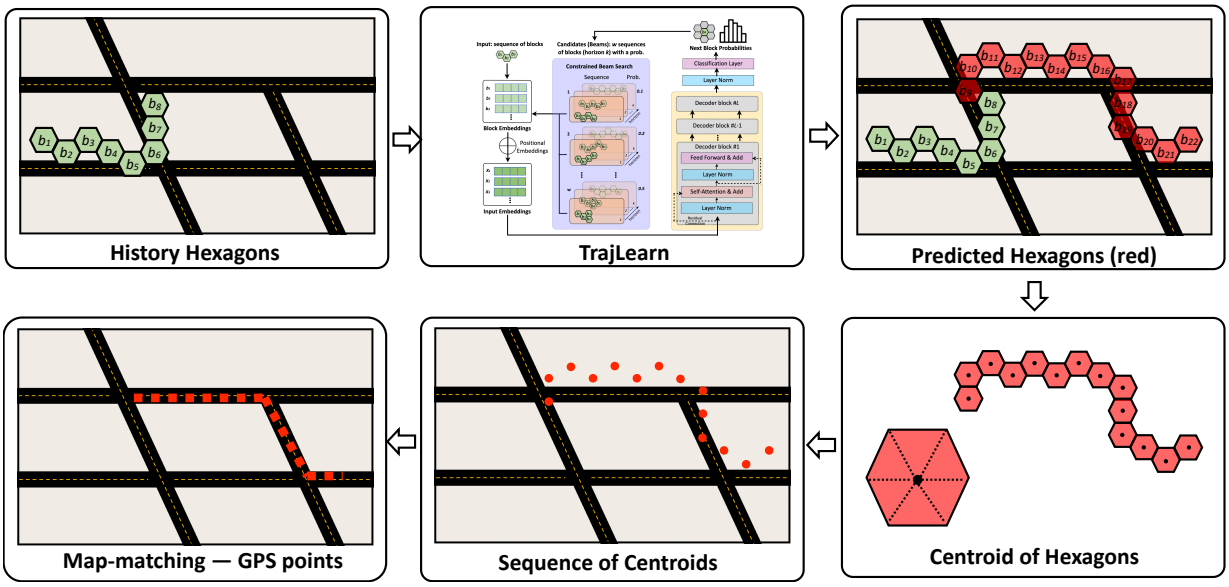


Figure 7.1: Pipeline for mapping predicted hexagons to GPS points.

7.1.3 Case Study

To illustrate the effectiveness of our centroid-based mapping approach, we conducted a detailed case study using the GeoLife dataset. In this experiment, we selected a representative trajectory from a pedestrian and divided it into two segments:

- **Input Trajectory:** The initial segment of the trajectory, which serves as input for our trajectory prediction model.
- **True Future Trajectory:** The ground truth path followed by the pedestrian after the input segment.

Our TrajLearn model was used to predict the next sequence of hexagon blocks from the input trajectory. These predicted hexagon blocks were then converted to GPS points by calculating the centroids. The resulting trajectory comprised:

1. **Hexagon Centroids:** The raw centroids derived directly from the predicted hexagons.

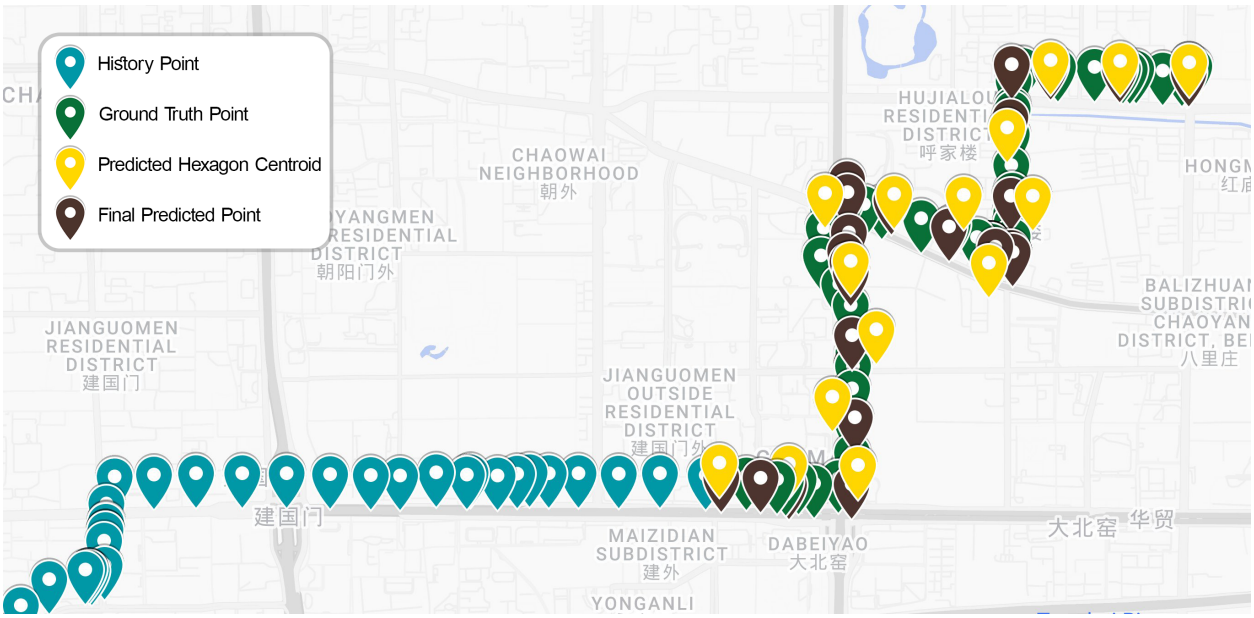


Figure 7.2: Visualization of a trajectory from the GeoLife dataset, presenting the input trajectory GPS points, ground truth history points of the trajectory, hexagon centroids of predicted hexagons using TrajLearn, and final GPS prediction generated using map-matching the hexagon centroids.

2. **Final GPS Prediction:** The centroids were subsequently refined using a map-matching algorithm (based on OSRM) to adjust for the road network and produce a final set of GPS coordinates.

Figure 7.2 presents the visualization from this case study. The figure includes:

- The original input trajectory (displayed as a series of GPS points).
- The ground truth future trajectory.
- The centroids of the predicted hexagons.
- The final GPS predictions after map matching.

This approach reveals not only how well the hexagonal representations approximate real paths but also highlights the accuracy improvements when applying map-matching

techniques to correct centroid-based outputs. The visual comparison demonstrates how centroid predictions alone can be offset, particularly in complex urban areas, but significantly align more closely with true paths post map-matching.

Figure 7.2 showcases these trajectories, marking differences between centroid estimates and adjusted GPS points, effectively bridging hexagonal predictions and practical spatial analysis.

7.2 Hierarchical Maps

In this section, we describe a hierarchical mapping strategy that further enhances the spatial resolution of our mobility data representation. Hierarchical maps enable the model to dynamically adjust the tessellation resolution based on the local density of trajectory data, resulting in a more precise representation in areas of high activity while conserving computational resources in low-density regions.

7.2.1 Motivation

Urban environments are characterized by significant spatial heterogeneity. In densely populated city centers, the movement patterns are complex and require fine-grained spatial resolution to capture detailed trajectories and subtle interactions. In contrast, in suburban or rural areas, coarse spatial representation may be sufficient. A uniform tessellation using a single resolution cannot optimally model such variance. Accurate trajectory prediction requires a nuanced understanding of movement patterns, which can differ markedly between densely populated urban centers and sparsely inhabited outskirts. Uniform map resolutions often fail to provide the necessary granularity, leading to generalized predictions that lack spatial specificity in critical areas. For instance, predicting movements in a bustling city

center demands finer granularity to distinguish between closely situated points of interest, whereas suburban regions may not require such detailed representation.

To address this challenge, we introduce a *hierarchical mapping* approach that dynamically adjusts hexagon sizes based on the local density of trajectory data. Hierarchical mapping addresses this problem by employing a mixed-resolution tessellation strategy:

- **Enhanced Spatial Precision:** Smaller hexagons in high-activity regions enable more precise localization and detailed trajectory predictions, capturing nuanced movement patterns that uniform resolutions might miss.
- **Scalability:** The adaptive nature of hierarchical maps allows the system to scale seamlessly with increasing data volumes and urban expansion, maintaining performance without a linear increase in resource consumption.
- **Efficient Resource Utilization:** By allocating larger hexagons to low-activity areas, the approach reduces computational and memory overhead, focusing resources on regions where fine-grained predictions are most beneficial.
- **Adaptive Data Representation:** By matching the map's resolution to the inherent data distribution, hierarchical maps provide a more accurate and meaningful representation of spatial dynamics, enhancing overall performance.

This approach mitigates the limitations of fixed-resolution tessellation and leads to more accurate trajectory predictions and contact detection, especially in environments with widely varying movement densities.

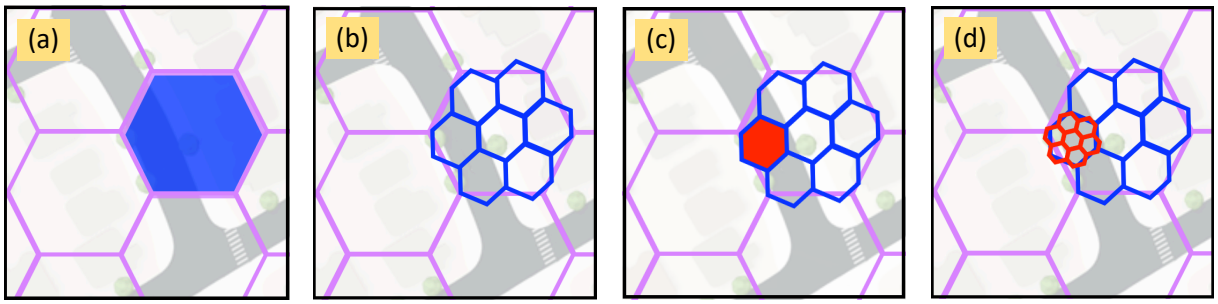


Figure 7.3: An example of a hierarchical map. (a) Initial tessellation with a blue hexagon representing a high-activity area; (b) Subdivision of the blue hexagon into seven smaller hexagons for increased precision; (c) Identification of another high-activity area with a red hexagon; (d) Final tessellation displaying varying resolutions across the map, with smaller hexagons in high-activity regions.

7.2.2 Approach

The hierarchical map generation process begins with an initial tessellation at a base resolution R_{min}. Hexagons are iteratively subdivided based on local trajectory density, allowing finer granularity in areas with concentrated movement patterns. This subdivision continues until a predefined maximum resolution R_{max} is reached or until no further subdivisions are necessary based on the termination criteria.

The algorithm proceeds as follows:

In this algorithm, `termination_condition_fn` and `splitting_condition_fn` are functions that determine when the iterative process should stop and which hexagons should be further subdivided, respectively. These functions can be defined based on various metrics such as frequency distributions, skewness, entropy, variance, local density measures, or other domain-specific criteria like proximity to critical infrastructure or areas of interest. This methodology allows the map to adaptively adjust its resolution in different regions, providing higher granularity where the data is dense and lower granularity where the data is sparse. The general algorithm is abstracted to accommodate various definitions of splitting and

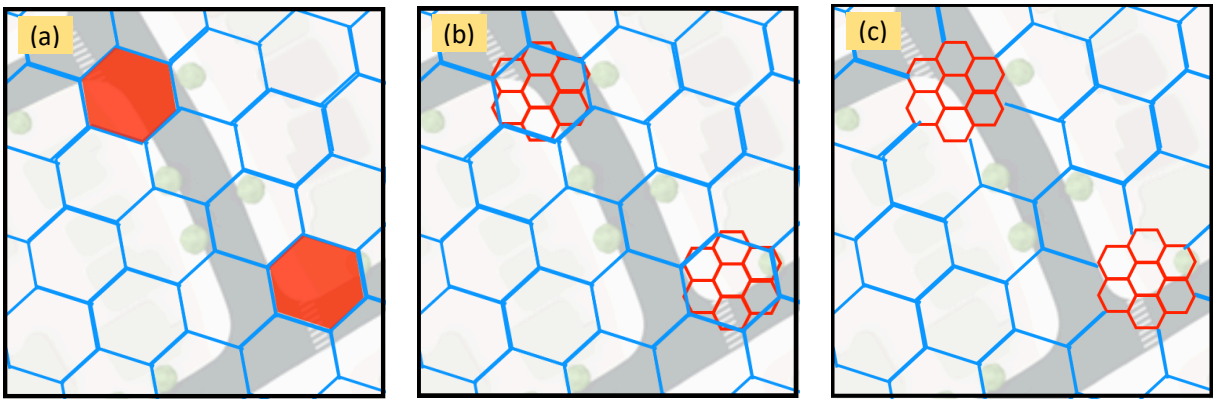


Figure 7.4: An example of a mixed-resolution map. (a) A map is tessellated and two red hexagons represent busy areas, (b) busy hexagons are further split into smaller hexagons, (c) final tesselation with varying resolutions in the map.

Input: Map M, Minimum resolution R_{min}, Maximum resolution R_{max}, Maximum iterations max_iter
Output: Hierarchical map H
`// Initialize hexagon set at the minimum resolution`
$H_0 \leftarrow$ `Tessellate`(M, R_{min});
for $i \leftarrow 0$ **to** $max\_iter - 1$ **do**
  if `termination_condition_fn`(H_i) **then**
    return H_i
  end
  foreach *hexagon* $h \in H_i$ **do**
    if `splitting_condition_fn`(h) ***and*** `Resolution`$(h) < R_{max}$ **then**
      $H_{i+1} \leftarrow$ `Tessellate`$(h,$ `Resolution`$(h))$;
    else
      $H_{i+1} \leftarrow h$;
    end
  end
end
return H_{max_iter}

Algorithm 1: Hierarchical Map Generation Algorithm

termination conditions, making it flexible for different applications and datasets.

Figure 7.3 conceptually illustrates the hierarchical mapping process. Initially, a blue hexagon, representing a high-activity region, is subdivided into seven smaller hexagons.

Similarly, additional high-activity areas are identified and refined, leading to a final tessellation with mixed resolutions. Figure 7.4 further demonstrates a real example of such a mixed-resolution map.

7.2.3 Implementation Details

Our implementation of the hierarchical mapping approach leverages trajectory frequency and spatial distribution to guide the subdivision of hexagons. By focusing on areas with high trajectory density and significant spatial variability, the map becomes more granular where detailed predictions are essential, while larger hexagons suffice in regions with uniform or low activity. Below, we outline the splitting and termination conditions for the iterative hierarchical map tessellation process, and discuss the rationale behind the parameter selection.

Splitting Condition. A hexagon h is eligible for splitting if it satisfies the following conditions:

1. **High Trajectory Density:** The frequency $f(h)$ of trajectories passing through h exceeds a threshold δ. This ensures that only regions with substantial movement data are considered for finer granularity.

2. **Spatial Variability:** The spatial variance $\sigma^2(h)$ within h surpasses a threshold ϕ. This condition targets areas where diverse movement patterns require more detailed representation.

3. **Resolution Cap:** The current resolution of h is below the maximum resolution $R_{\max}$. This prevents excessive subdivision and controls the map's overall granularity.

Mathematically, the splitting condition for a hexagon h can be expressed as:

$$\texttt{splitting\_condition\_fn}(h) = \begin{cases} \text{True} & \text{if } f(h) > \delta \text{ and } \gamma(h) > \phi \text{ and } R(h) < R_{\max}, \\ \text{False} & \text{otherwise,} \end{cases}$$

where $R(h)$ denotes the resolution of hexagon h.

Termination Condition. The iterative subdivision process terminates when the overall spatial variability across all hexagons falls below a threshold θ, indicating that the map has achieved sufficient granularity for accurate trajectory prediction. Formally, the termination condition is:

$$\texttt{termination\_condition\_fn}(H_i) = \begin{cases} \text{True} & \text{if } \Gamma(H_i) < \theta \text{ or } |H_i| = |H_{i-1}|, \\ \text{False} & \text{otherwise,} \end{cases}$$

where $\Gamma(H_i)$ is the skewness of the frequency distribution of the hexagon set H_i.

Parameter Selection. The thresholds δ, ϕ, and θ are critical for balancing map precision and computational efficiency. These parameters were empirically determined based on dataset-specific characteristics:

- **Trajectory Density Threshold (δ):** Set to identify hexagons with trajectory frequencies significantly above the median, ensuring that only the most active regions are refined.
- **Spatial Variability Threshold (ϕ):** Chosen to detect areas with diverse movement patterns, prompting subdivision where spatial heterogeneity is high.
- **Overall Variability Threshold (θ):** Selected to balance the trade-off between map detail and computational resources, terminating the refinement process when spatial variability is sufficiently low.

7.2.4 Experimental Results

We evaluated the effectiveness of our mixed-resolution mapping approach on the Ho-GeoLife, Ho-Rome, and Ho-Porto datasets. We present the model's performance using the mixed-resolution maps over the metrics Acc@1, Acc@3, Acc@5, and BLEU scores, which measure the accuracy of trajectory predictions at different levels. For our experiments, we set the minimum resolution $R_{\min} = 7$ and the maximum resolution $R_{\max} = 9$ with input length $l = 10$ & prediction horizon $k = 5$. The results are summarized in Table 7.1.

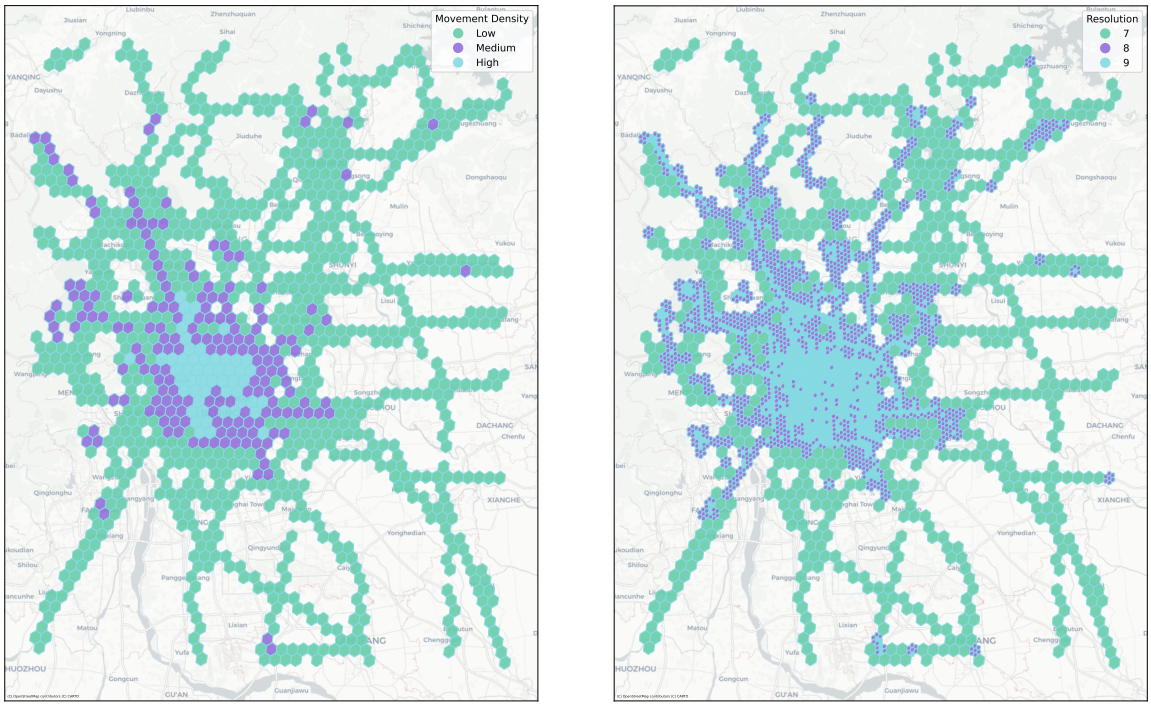
Dataset	Mixed Resolution (Res = 7–9)			
	Acc@1	Acc@3	Acc@5	BLEU
Ho-Porto	0.4476	0.5333	0.6099	0.5289
Ho-Rome	0.3213	0.4915	0.5605	0.3747
Ho-GeoLife	0.4854	0.5727	0.6496	0.5267

Table 7.1: TrajLearn's prediction performance using hierarchical maps across various datasets.

Figure 7.5 illustrates the distinction between fixed-resolution and hierarchical tessellations for the Ho-GeoLife dataset. The hierarchical map achieves greater spatial specificity in high-density areas, facilitating more accurate and localized trajectory predictions without incurring significant computational overhead in less active regions.

7.3 Flexible Contact Definitions

Predicting dynamic mobility networks depends critically on how "contact" is defined. In Chapter 3 we applied a strict same-hexagon criterion. Here, we propose three progressively relaxed definitions, each easily plugged into MobiNetForecast's contact-detection module, to better suit diverse real-world requirements.



(a) Movement density at resolution 7 tessellation. (b) Hierarchical tessellation (Resolutions 7, 8, 9).

Figure 7.5: Comparison of tessellations for the HO-GEOLIFE dataset: (a) Fixed resolution; (b) Hierarchical resolution. The hierarchical map provides finer granularity in high-activity areas, enabling more specific trajectory predictions.

7.3.1 Motivation

In the real world, defining a "contact" as two individuals occupying exactly the same hexagon at the same instant can be overly rigid. GPS noise, sampling asynchrony, and partial data loss frequently cause minor misalignments in recorded trajectories. Moreover, brief near-miss encounters—where two people pass just across a hexagon boundary—may still be epidemiologically or socially significant. To better support diverse application needs, it is essential to introduce more flexible criteria that (1) tolerate spatial or temporal uncertainty, (2) capture fleeting or approximate interactions, and (3) integrate historical co-location context.

7.3.2 Approach

Building on the strict same-hexagon rule of Chapter 3, we define four alternative contact predicates, each progressively relaxing the requirement for exact co-location. Let $b_u^t, b_v^t \in \mathcal{B}$ be the hexagon IDs of individuals u and v at discrete time t, and define $d_H(\cdot,\cdot)$ as the shortest-path distance on the hexagon adjacency graph and $d_E(\cdot,\cdot)$ as Euclidean distance between continuous coordinates. We consider:

1. Same-or-Neighboring Hexagon at the Same Time A contact occurs at time t if the two individuals occupy either the same hexagon or one of its six adjacent neighbors. Formally, letting $d_H(b_u^t, b_v^t)$ denote the shortest-path distance on the hexagon adjacency graph, we declare contact whenever

$$d_H\big(b_u^t, b_v^t\big) \;\le\; 1.$$

This definition provides a one-cell buffer that accommodates small GPS errors and captures interactions that straddle hexagon boundaries.

2. Time-Windowed Contact To address temporal misalignment, we allow a sliding window of w past time steps. A contact at time t is recorded if, within the interval $[t-w,\, t]$, the two individuals satisfy the same-or-neighboring condition at any t'. That is,

$$\exists\, t' \in [t-w,\, t] \;:\; d_H\big(b_u^{t'}, b_v^{t'}\big) \le 1.$$

This windowed approach captures brief encounters that might be missed in a single time slice.

3. Proximity-Based Contact Rather than relying on hexagon IDs, one can compute the Euclidean (or geodesic) distance $d_E(\ell_u^t, \ell_v^t)$ between the continuous coordinates ℓ_u^t, ℓ_v^t. A

contact is declared if

$$d_E\big(\ell_u^t, \ell_v^t\big) \;\le\; d_{\max},$$

where $d_{\max}$ is a configurable threshold (e.g., 5–10 m). This definition directly accommodates raw GPS data—after map-matching or centroid conversion—and can be tuned to the application's spatial scale.

4. Probabilistic Contact Finally, we can combine spatial adjacency, continuous distance, and historical co-location frequency $p_{\text{past}}(u, v)$ into a probability model:

$$P\big(C_{uv}(t) = 1\big) = f\Big(d_H(b_u^t, b_v^t),\, d_E(\ell_u^t, \ell_v^t),\, p_{\text{past}}(u, v)\Big),$$

where f is a logistic or learned function producing a contact probability in $[0, 1]$. A user-specified threshold τ then determines whether $C_{uv}(t)$ is declared. This framework allows historical and contextual factors to inform contact likelihood.

Each of these definitions can be swapped into the contact-detection module with minimal code changes. Practitioners can select the variant that best balances precision and recall for their specific domain.

7.3.3 Applications

Different application domains benefit from different contact criteria:

Epidemic Surveillance. In crowded venues or healthcare settings, fleeting exposures may transmit disease. A time-windowed or same-$\pm$-neighbor definition boosts recall, ensuring that brief or near-miss interactions trigger alerts.

Traffic and Crowd Management. For vehicle or pedestrian flow control, a proximity-based criterion (e.g., within 5–10 m) captures realistic interaction distances, guiding congestion

mitigation and safety interventions.

Workplace & Campus Safety. In buildings or campuses, a probabilistic contact model integrating badge-swipe history and floor-plan adjacency identifies likely interactions even when spatial data are noisy or intermittent.

Location-Based Services. Retailers and advertisers can use same-or-neighboring hexagon contacts to trigger geo-fenced promotions, balancing precision with footfall uncertainty.

Privacy-Preserving Analytics. A relaxed hexagon-adjacency definition (rather than raw coordinates) reduces re-identification risk, allowing cities to share mobility insights while preserving anonymity.

By offering these plug-and-play contact definitions, MobiNetForecast can be tailored to the precision–recall trade-offs and domain constraints of any mobility application.

Chapter 8

Conclusion

This final chapter synthesizes the contributions of our work and outlines pathways for future exploration. In what follows, we provide a summary of our key findings, address the limitations encountered throughout our research, and describe promising directions for future work.

8.1 Summary of Findings

Our research set out to tackle the challenging problem of forecasting mobility networks in dynamic urban environments. To this end, we developed a comprehensive framework that integrates trajectory prediction learning with contact detection and network construction. The work can be summarized in three main contributions:

Trajectory Prediction Learning via TrajLearn

We introduced TrajLearn, a Transformer-based model designed to predict future trajectory segments, where trajectories are represented as sequences of hexagon blocks obtained via a higher-order mobility data representation. By drawing analogies from language model-

ing—treating hexagon IDs as tokens—we leveraged the self-attention mechanism to capture intricate spatial–temporal dependencies. Our model is characterized by:

- A decoder-only Transformer architecture with learnable positional embeddings that adapts to the structure of the mobility flow data.
- Training techniques such as teacher forcing, which have facilitated rapid convergence and enhanced performance.
- A constrained beam search during inference that ensures spatial continuity of predicted trajectories by restricting the next-step predictions to adjacent hexagons.

Empirical evaluations demonstrated that TrajLearn outperforms baseline approaches such as traditional Markov Chains, LSTM, GRU, and even more advanced models like Flashback++. These results confirm that our approach effectively captures both short-range and long-range dependencies inherent in human mobility patterns.

Robust Contact Detection and Mobility Network Construction

Building on the trajectory prediction component, we devised a methodology to identify contacts between moving objects by:

- Mapping each predicted trajectory to a sequence of block-time pairs, thereby synchronizing trajectories that start at different times.
- Defining contacts through a simple yet effective indicator function that flags a contact when two objects are present in the same hexagon at the same time.
- Aggregating these contacts over time to construct time-varying (dynamic) mobility network snapshots.

This pipeline, which we refer to as MobiNetForecast, effectively integrates individual future trajectory predictions into a cohesive dynamic graph, thereby enabling real-time analysis of urban mobility interactions.

Hierarchical Mapping and Data Representation

An important facet of our contribution lies in the transformation of raw GPS trajectory data into a structured, higher-order representation. By tessellating geographic areas into hexagonal blocks:

- We reduce data sparsity and noise, which is particularly important when working with GPS data that are inherently continuous and unevenly distributed.
- The hexagon-based approach provides a natural discretization, simplifying the task of sequence prediction and ensuring that the spatial relationships are uniform and isotropic.
- We further enhanced our spatial representation by introducing hierarchical maps that allow for variable resolutions. This enables finer granularity in high-density regions while maintaining computational efficiency in less active areas.

The combined effect of these techniques is a robust input representation that underpins the superior performance of our trajectory prediction and mobility network forecasting framework.

8.2 Limitations

Despite the promising results and comprehensive design of our framework, several limitations remain that are important to consider when interpreting our findings. First, while our hexagon-based representation effectively mitigates data sparsity inherent in raw GPS data, the overall quality of the input data still poses a significant challenge. In urban environments,

GPS signals are prone to noise and errors due to multipath effects, urban canyons, and intermittent data loss—all of which can adversely affect the accuracy of trajectory predictions.

Furthermore, existing trajectory datasets often contain inherent biases that may not comprehensively represent all demographic or geographic segments. Such biases can lead to uneven performance across different groups, potentially limiting the generalizability and fairness of our predictions. Although addressing these dataset biases is crucial for developing equitable and reliable trajectory prediction systems, a thorough investigation into these issues falls beyond the scope of the present work. We encourage future research to carry out comprehensive bias assessments on publicly available trajectory datasets to ensure that prediction systems are inclusive and robust.

8.3 Future Research Directions

Building on the current work, several promising avenues for future research have emerged. One key direction is the integration of additional sources of contextual data to enhance the richness of the input features in cases where such information is available. For instance, incorporating multi-modal data—such as weather conditions, traffic incidents, special event information, or even social media feeds—could provide a more holistic understanding of the factors that drive mobility patterns. Enriching the model with semantic information, for example by integrating points-of-interest (POI) and land-use patterns, may also help disambiguate areas with high interaction rates and yield predictions that are even more context-aware.

Another important direction is the exploration of pre-training techniques to enable transfer learning across diverse tasks and trajectory datasets. By leveraging pre-trained models, it may be possible to build a more general framework that adapts more effectively to different

urban and non-urban settings with limited task-specific data.

Moreover, it will be valuable to test and adapt our approach in a broader array of environments. Our current work primarily focuses on urban taxi and pedestrian data; future studies could explore its applicability to indoor navigation, multi-modal transportation systems, and rural mobility scenarios. Such investigations would help assess the generality and robustness of our methods, as well as highlight any necessary adaptations for different application contexts.

These future research directions, while extending beyond the scope of our current study, promise to build on the foundation we have established and address the limitations inherent in our present approach. They pave the way for more comprehensive mobility forecasting systems that are both equitable and broadly applicable across diverse environments.

Bibliography

[1] Yu Zheng. "Trajectory data mining: an overview". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 6.3 (2015), pp. 1–41.

[2] Moritz UG Kraemer, Joseph L-H Tsui, Serina Y Chang, Spyros Lytras, Mark P Khurana, Samantha Vanderslott, Sumali Bajaj, Neil Scheidwasser, Jacob Liam Curran-Sebastian, Elizaveta Semenova, et al. "Artificial intelligence for modelling infectious disease epidemics". In: *Nature* 638.8051 (2025), pp. 623–635.

[3] Moritz UG Kraemer, Chia-Hung Yang, Bernardo Gutierrez, Chieh-Hsi Wu, Brennan Klein, David M Pigott, Open COVID-19 Data Working Group, Louis Du Plessis, Nuno R Faria, Ruoran Li, et al. "The effect of human mobility and control measures on the COVID-19 epidemic in China". In: *Science* 368.6490 (2020), pp. 493–497.

[4] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. "Mobility network models of COVID-19 explain inequities and inform reopening". In: *Nature* 589.7840 (2021), pp. 82–87.

[5] Francisco Barreras and Duncan J Watts. "The exciting potential and daunting challenge of using GPS human-mobility data for epidemic modeling". In: *Nature Computational Science* 4.6 (2024), pp. 398–411.

[6] Wenchao Yu, Wei Cheng, Charu C Aggarwal, Haifeng Chen, and Wei Wang. "Link prediction with spatial and temporal consistency in dynamic networks." In: *IJCAI*. 2017, pp. 3343–3349.

[7] Srijan Kumar, Xikun Zhang, and Jure Leskovec. "Predicting dynamic embedding trajectory in temporal interaction networks". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 1269–1278.

[8] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. "Temporal graph networks for deep learning on dynamic graphs". In: *arXiv preprint arXiv:2006.10637* (2020).

[9] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. "Inductive representation learning on temporal graphs". In: *arXiv preprint arXiv:2002.07962* (2020).

[10] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. "Evolvegcn: Evolving graph convolutional networks for dynamic graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 5363–5370.

[11] Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. "Towards better evaluation for dynamic link prediction". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 32928–32941.

[12] John A. Miller, Mohammed Aldosari, Farah Saeed, Nasid Habib Barna, Subas Rana, I. Budak Arpinar, and Ninghao Liu. *A Survey of Deep Learning and Foundation Models for Time Series Forecasting*. 2024. arXiv: 2401.13912 [cs.LG]. URL: https://arxiv.org/abs/2401.13912.

[13] K. Benidis, S. S. Rangapuram, V. Flunkert, Y. Wang, D. C. Maddix, C. Türkmen, J. Gasthaus, M. Schneider, D. Salinas, L. Stella, F. Aubet, L. Callot, and T. Januschowski. "Deep learning for time series forecasting: tutorial and literature survey". In: *ACM Computing Surveys* 55 (6 2022), pp. 1–36. DOI: 10.1145/3533382.

[14] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[15] Dong Wang, Takayuki Miwa, and Takayuki Morikawa. "Big trajectory data mining: A survey of methods, applications, and services". In: *Sensors* 20.16 (2020), p. 4571.

[16] Gagan Atluri, Anuj Karpatne, and Vipin Kumar. "Spatio-temporal data mining: A survey of problems and methods". In: *ACM Computing Surveys (CSUR)* 51.4 (2018), pp. 1–41.

[17] Amirhossein Nadiri, Jing Li, Ali Faraji, Ghadeer Abuoda, and Manos Papagelis. "TrajLearn: Trajectory Prediction Learning using Deep Generative Models". In: *ACM Trans. Spatial Algorithms Syst.* 11.3 (May 2025). ISSN: 2374-0353. DOI: 10.1145/3729226. URL: https://doi.org/10.1145/3729226.

[18] Ali Faraji, Jing Li, Gian Alix, Mahmoud Alsaeed, Nina Yanin, Amirhossein Nadiri, and Manos Papagelis. "Point2Hex: Higher-order Mobility Flow Data and Resources". In: *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9798400701689. DOI: 10.1145/3589132.3625619. URL: https://doi.org/10.1145/3589132.3625619.

[19] W. Chen, Y. Liang, Y. Zhu, and Y. Chang. "Deep learning for trajectory data management and mining: A survey and beyond". In: *arXiv preprint arXiv:2403.14151* (2024).

[20] Tilemachos Pechlivanoglou, Gian Alix, Nina Yanin, Jing Li, Farzaneh Heidari, and Manos Papagelis. "Microscopic modeling of spatiotemporal epidemic dynamics". In: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Spatial Computing for Epidemiology*. 2022, pp. 11–21.

[21] Tilemachos Pechlivanoglou, Jing Li, Jialin Sun, Farzaneh Heidari, and Manos Papagelis. "Epidemic Spreading in Trajectory Networks". In: *Big Data Research* 27 (2022), p. 100275.

[22] Chih-Chieh Hung, Wen-Chih Peng, and Wang-Chien Lee. "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes". In: *The VLDB Journal* 24 (2015), pp. 169–192.

[23] Yu Zheng, Yukun Chen, Quannan Li, Xing Xie, and Wei-Ying Ma. "Understanding transportation modes based on GPS data for web applications". In: *ACM Transactions on the Web (TWEB)* 4.1 (2010), pp. 1–36.

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

[25] Jae-Gil Lee, Jiawei Han, and Xiaolei Li. "Trajectory outlier detection: A partition-and-detect framework". In: *2008 IEEE 24th International Conference on Data Engineering*. IEEE. 2008, pp. 140–149.

[26] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. "Trajectory clustering: a partition-and-group framework". In: *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 2007, pp. 593–604.

[27] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. "Semantic trajectories: Mobility data computation and annotation". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 4.3 (2013), pp. 1–38.

[28] Xiaolei Li, Jiawei Han, Jae-Gil Lee, and Hector Gonzalez. "Traffic density-based discovery of hot routes in road networks". In: *Advances in Spatial and Temporal Databases: 10th International Symposium, SSTD 2007, Boston, MA, USA, July 16-18, 2007. Proceedings 10*. Springer. 2007, pp. 441–459.

[29] Binh Han, Ling Liu, and Edward Omiecinski. "Neat: Road network aware trajectory clustering". In: *2012 IEEE 32nd International Conference on Distributed Computing Systems*. IEEE. 2012, pp. 142–151.

[30] Adel Bolbol, Tao Cheng, Ioannis Tsapakis, and James Haworth. "Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification". In: *Computers, Environment and Urban Systems* 36.6 (2012), pp. 526–537.

[31] Quanjun Chen, Xuan Song, Harutoshi Yamada, and Ryosuke Shibasaki. "Learning deep representation from big and heterogeneous data for traffic accident inference". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[32] Nina Yanin and Manos Papagelis. "Optimal Risk-aware POI Recommendations during Epidemics". In: *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Spatial Computing for Epidemiology*. SpatialEpi '23. Hamburg, Germany: Association for Computing Machinery, 2023, pp. 1–12. ISBN: 9798400703607. DOI: 10.1145/3615898.3628256. URL: https://doi.org/10.1145/3615898.3628256.

[33] Mahmoud Alsaeed, Ameeta Agrawal, and Manos Papagelis. "Trajectory-User Linking using Higher-order Mobility Flow Representations". In: *2023 24th IEEE International Conference on Mobile Data Management (MDM)*. IEEE. 2023, pp. 158–167.

[34] Senzhang Wang, Jiannong Cao, and Philip Yu. "Deep learning for spatio-temporal data mining: A survey". In: *IEEE transactions on knowledge and data engineering* (2020).

[35] G. Jin, Y. Liang, Y. Fang, Z. Shao, and J. Huang. "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey". In: *IEEE Transactions on Knowledge and Data Engineering* (2023).

[36] Guangyin Jin, Yuxuan Liang, Yuchen Fang, Zezhi Shao, Jincai Huang, Junbo Zhang, and Yu Zheng. "Spatio-Temporal Graph Neural Networks for Predictive Learning in Urban Computing: A Survey". In: *IEEE Trans. on Knowl. and Data Eng.* 36.10 (Oct. 2024), pp. 5388–5408. ISSN: 1041-4347. DOI: 10.1109/TKDE.2023.3333824. URL: https://doi.org/10.1109/TKDE.2023.3333824.

[37] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. "Traffic flow prediction with big data: A deep learning approach". In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2014), pp. 865–873.

[38] Yuxuan Zhang, Senzhang Wang, Bing Chen, Jiannong Cao, and Zhiqiu Huang. "Trafficgan: Network-scale deep traffic prediction with generative adversarial nets". In: *IEEE Transactions on Intelligent Transportation Systems* 22.1 (2019), pp. 219–230.

[39] Daniel Ashbrook and Thad Starner. "Using GPS to learn significant locations and predict movement across multiple users". In: *Personal and Ubiquitous computing* 7 (2003), pp. 275–286.

[40] Libo Song, David Kotz, Ravi Jain, and Xiaoning He. "Evaluating location predictors with extensive Wi-Fi mobility data". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 7.4 (2003), pp. 64–65.

[41] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. "Wherenext: a location predictor on trajectory pattern mining". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 637–646.

[42] Preeti Bhargava, Thomas Phan, Jiayu Zhou, and Juhan Lee. "Who, what, when, and where: Multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data". In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 130–140.

[43] Huiji Gao, Jiliang Tang, and Huan Liu. "Exploring social-historical ties on location-based social networks". In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 6. 1. 2012, pp. 114–121.

[44] Chen Cheng, Haiqin Yang, Irwin King, and Michael Lyu. "Fused matrix factorization with geographical and social influence in location-based social networks". In: *Proc. of the AAAI conference on artificial intelligence*. Vol. 26. 1. 2012, pp. 17–23.

[45] Xutao Li, Gao Cong, Xiao-Li Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. "Rank-geofm: A ranking based geographical factorization method for point of interest recommendation". In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 2015, pp. 433–442.

[46] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. "GeoMF: joint geographical modeling and matrix factorization for point-of-interest

recommendation". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 831–840.

[47] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. "Predicting the next location: A recurrent model with spatial and temporal contexts". In: *AAAI*. Vol. 30. 2016.

[48] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. "Deepmove: Predicting human mobility with attentional recurrent networks". In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 1459–1468.

[49] Defu Lian, Yongji Wu, Yong Ge, Xing Xie, and Enhong Chen. "Geography-Aware Sequential Location Recommendation". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, pp. 2009–2019. ISBN: 9781450379984. DOI: 10.1145/3394486.3403252. URL: https://doi.org/10.1145/3394486.3403252.

[50] Dingqi Yang, Benjamin Fankhauser, Paolo Rosso, and Philippe Cudre-Mauroux. "Location prediction over sparse user mobility traces using rnns". In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. 2020, pp. 2184–2190.

[51] Yingtao Luo, Qiang Liu, and Zhaocheng Liu. "Stan: Spatio-temporal attention network for next location recommendation". In: *Proceedings of the web conference 2021*. 2021, pp. 2177–2185.

[52] Qiang Gao, Fan Zhou, Goce Trajcevski, Kunpeng Zhang, Ting Zhong, and Fengli Zhang. "Predicting human mobility via variational attention". In: *The world wide web conference*. 2019, pp. 2750–2756.

[53] Congcong Miao, Ziyan Luo, Fengzhu Zeng, and Jilong Wang. "Predicting human mobility via attentive convolutional network". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020, pp. 438–446.

[54] Weizhen Dang, Haibo Wang, Shirui Pan, Pei Zhang, Chuan Zhou, Xin Chen, and Jilong Wang. "Predicting Human Mobility via Graph Convolutional Dual-attentive Networks". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 192–200.

[55] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. "Where you like to go next: Successive point-of-interest recommendation". In: *Twenty-Third international joint conference on Artificial Intelligence*. 2013.

[56] Wesley Mathew, Ruben Raposo, and Bruno Martins. "Predicting future locations with hidden Markov models". In: *Proceedings of the 2012 ACM conference on ubiquitous computing*. 2012, pp. 911–918.

[57] Hongzhi Shi, Yong Li, Hancheng Cao, Xiangxin Zhou, Chao Zhang, and Vassilis Kostakos. "Semantics-aware hidden Markov model for human mobility". In: *IEEE Transactions on Knowledge and Data Engineering* 33.3 (2019), pp. 1183–1194.

[58] Bingqi Yan, Geng Zhao, Lexue Song, Yanwei Yu, and Junyu Dong. "PreCLN: Pretrained-based contrastive learning network for vehicle trajectory prediction". In: *World Wide Web* 26.4 (2023), pp. 1853–1875.

[59] Renhe Jiang, Xuan Song, Zipei Fan, Tianqi Xia, Quanjun Chen, Satoshi Miyazawa, and Ryosuke Shibasaki. "Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[60] Amin Sadri, Flora D Salim, Yongli Ren, Wei Shao, John C Krumm, and Cecilia Mascolo. "What will you do for the rest of the day? An approach to continuous trajectory prediction". In: *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 2.4 (2018), pp. 1–26.

[61] Licia Amichi, Aline Carneiro Viana, Mark Crovella, and Antonio AF Loureiro. "From movement purpose to perceptive spatial mobility prediction". In: *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. 2021, pp. 500–511.

[62] Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, and Manos Papagelis. "A versatile computational framework for group pattern mining of pedestrian trajectories". In: *GeoInformatica* 23 (2019), pp. 501–531.

[63] Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, and Manos Papagelis. "Trajectolizer: Interactive analysis and exploration of trajectory group dynamics". In: *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE. 2018, pp. 286–287.

[64] Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, and Manos Papagelis. "Tensor methods for group pattern discovery of pedestrian trajectories". In: *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE. 2018, pp. 76–85.

[65] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. "Continuous-Time Dynamic Network Embeddings". In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. <conf-loc>, <city>Lyon</city>, <country>France</country>, </conf-loc>: International World Wide Web Conferences Steering Committee, 2018, pp. 969–976. ISBN: 9781450356404.

DOI: 10.1145/3184558.3191526. URL: https://doi.org/10.1145/3184558.3191526.

[66] Yuan Zuo, Guannan Liu, Hao Lin, Jia Guo, Xiaoqian Hu, and Junjie Wu. "Embedding Temporal Network via Neighborhood Formation". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2857–2866. ISBN: 9781450355520. DOI: 10.1145/3219819.3220054. URL: https://doi.org/10.1145/3219819.3220054.

[67] da Xu, chuanwei ruan, evren korpeoglu, sushant kumar, and kannan achan. "Inductive representation learning on temporal graphs". In: *International Conference on Learning Representations (ICLR)*. 2020.

[68] Rakshit S. Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. "DyRep: Learning Representations over Dynamic Graphs". In: *International Conference on Learning Representations*. 2019. URL: https://api.semanticscholar.org/CorpusID:108296188.

[69] Baichuan Yuan, Hao Li, Andrea L. Bertozzi, P. Jeffrey Brantingham, and Mason A. Porter. "Multivariate Spatiotemporal Hawkes Processes and Network Reconstruction". In: *SIAM Journal on Mathematics of Data Science* 1.2 (2019), pp. 356–382. DOI: 10.1137/18M1226993. eprint: https://doi.org/10.1137/18M1226993. URL: https://doi.org/10.1137/18M1226993.

[70] Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. "Inductive Representation Learning in Temporal Networks via Causal Anonymous Walks". In: *ArXiv* abs/2101.05974 (2021). URL: https://api.semanticscholar.org/CorpusID:231627759.

[71] Zhihao Wen and Yuan Fang. "TREND: TempoRal Event and Node Dynamics for Graph Representation Learning". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. <conf-loc>, <city>Virtual Event, Lyon</city>, <country>France</country>, </conf-loc>: Association for Computing Machinery, 2022, pp. 1159–1169. ISBN: 9781450390965. DOI: 10.1145/3485447.3512164. URL: https://doi.org/10.1145/3485447.3512164.

[72] Meng Qin and Dit-Yan Yeung. "Temporal Link Prediction: A Unified Framework, Taxonomy, and Review". In: *ACM Comput. Surv.* 56.4 (Nov. 2023). ISSN: 0360-0300. DOI: 10.1145/3625820. URL: https://doi.org/10.1145/3625820.

[73] Umang Sharan and Jennifer Neville. "Temporal-Relational Classifiers for Prediction in Evolving Domains". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 540–549. DOI: 10.1109/ICDM.2008.125.

[74] Wenchao Yu, Charu C. Aggarwal, and Wei Wang. "Temporally Factorized Network Modeling for Evolutionary Network Analysis". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. WSDM '17. Cambridge, United Kingdom: Association for Computing Machinery, 2017, pp. 455–464. ISBN: 9781450346757. DOI: 10.1145/3018661.3018669. URL: https://doi.org/10.1145/3018661.3018669.

[75] Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. "Temporal link prediction by integrating content and structure information". In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. CIKM '11. Glasgow, Scotland, UK: Association for Computing Machinery, 2011, pp. 1169–1174. ISBN: 9781450307178. DOI: 10.1145/2063576.2063744. URL: https://doi.org/10.1145/2063576.2063744.

[76] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. "Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks". In: *IEEE Transactions on Knowledge and Data Engineering* 28.10 (2016), pp. 2765–2777. DOI: 10.1109/TKDE.2016.2591009.

[77] Xiaoke Ma, Shiyin Tan, Xianghua Xie, Xiaoxiong Zhong, and Jingjing Deng. "Joint multi-label learning and feature extraction for temporal link prediction". In: *Pattern Recognition* 121 (2022), p. 108216. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2021.108216. URL: https://www.sciencedirect.com/science/article/pii/S0031320321003976.

[78] Xiaoke Ma, Penggang Sun, and Yu Wang. "Graph regularized nonnegative matrix factorization for temporal link prediction in dynamic networks". In: *Physica A: Statistical Mechanics and its Applications* 496 (2018), pp. 121–136. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2017.12.092. URL: https://www.sciencedirect.com/science/article/pii/S0378437117313316.

[79] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning". In: *Knowledge-Based Systems* 187 (2020), p. 104816. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2019.06.024. URL: https://www.sciencedirect.com/science/article/pii/S0950705119302916.

[80] Taisong Li, Jiawei Zhang, Philip S. Yu, Yan Zhang, and Yonghong Yan. "Deep Dynamic Network Embedding for Link Prediction". In: *IEEE Access* 6 (2018), pp. 29219–29230. DOI: 10.1109/ACCESS.2018.2839770.

[81] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. "EvolveGCN: Evolving

Graph Convolutional Networks for Dynamic Graphs". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 5363–5370. DOI: 10.1609/aaai.v34i04.5984. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5984.

[82] Kai Lei, Meng Qin, Bo Bai, Gong Zhang, and Min Yang. "GCN-GAN: A Non-linear Temporal Link Prediction Model for Weighted Dynamic Networks". In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. Paris, France: IEEE Press, 2019, pp. 388–396. DOI: 10.1109/INFOCOM.2019.8737631. URL: https://doi.org/10.1109/INFOCOM.2019.8737631.

[83] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting". In: *arXiv preprint arXiv:1707.01926* (2017).

[84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[85] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[86] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. "Palm: Scaling language modeling with pathways". In: *arXiv preprint arXiv:2204.02311* (2022).

[87] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[88] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[89] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[90] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. "Deep autoregressive networks". In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1242–1250.

[91] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE transactions on pattern analysis and machine intelligence* 43.11 (2020), pp. 3964–3979.

[92] S. Jamali and M. Moradi. "Free energy calculations using generative models trained on molecular dynamics trajectories: A diffusion model approach". In: *Biophysical Journal* (2025).

[93] L. Zhang, J. Mbuya, L. Zhao, and D. Pfoser. "End-to-end Trajectory Generation: Contrasting Deep Generative Models and Language Models". In: *ACM Transactions on Spatial Algorithms and Systems* (2025).

[94] R. Ye, Z. Lv, Z. Xu, and J. Li. "MT-CNN: A Lightweight Spatial-Temporal Convolutional Neural Network for Deep Learning of Complex Trajectory Distributions". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2024.

[95] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[96] Mingtian Tan, Mike A Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. "Are Language Models Actually Useful for Time Series Forecasting?" In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024. URL: https://openreview.net/forum?id=DV15UbHCY1.

[97] Zipei Fan, Xiaojie Yang, Wei Yuan, Renhe Jiang, Quanjun Chen, Xuan Song, and Ryosuke Shibasaki. "Online trajectory prediction for metropolitan scale mobility digital twin". In: *Proc. of the 30th ACM SIGSPATIAL*. 2022, pp. 1–12.

[98] Dingqi Yang, Daqing Zhang, Vincent. W. Zheng, and Zhiyong Yu. "Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs". In: *IEEE SMC* 45.1 (2015), pp. 129–142.

[99] Colin P.D. Birch, Sander P. Oom, and Jonathan A. Beecham. "Rectangular and hexagonal grids used for observation, experiment and simulation in ecology". In: *Ecological Modelling* 206.3 (2007), pp. 347–359. ISSN: 0304-3800.

[100] Inc. Uber Technologies. *Conversion from latitude/longitude to containing H3 cell index*. URL: https://h3geo.org/docs/core-library/latLngToCellDesc/ (visited on 09/13/2024).

[101] Ilya Loshchilov, Frank Hutter, et al. "Fixing weight decay regularization in adam". In: *arXiv preprint arXiv:1711.05101* 5 (2017), p. 5.

[102] Dan Hendrycks and Kevin Gimpel. "Bridging nonlinearities and stochastic regularizers with gaussian error linear units". In: *CoRR, abs/1606.08415* 3 (2016).

[103] Luis Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. *ECML/PKDD 15: Taxi Trajectory Prediction Porto Dataset*. Kaggle, 2015.

[104] Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. *CRAWDAD roma/taxi*. IEEE Dataport, 2022.

[105] Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. "Understanding Mobility Based on GPS Data". In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. UbiComp '08. Seoul, Korea: Association for Computing Machinery, 2008, pp. 312–321.

[106] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. "Mining Interesting Locations and Travel Sequences from GPS Trajectories". In: *Proc. 18th ACM world wide web*. 2009, pp. 791–800.

[107] Yu Zheng, Xing Xie, and Wei-Ying Ma. "GeoLife: A Collaborative Social Networking Service among User, location and trajectory". In: *IEEE Data(base) Engineering Bulletin* 33.2 (2010), pp. 32–39.

[108] Alberto Rossi, Gianni Barlacchi, Monica Bianchini, and Bruno Lepri. "Modelling Taxi Drivers' Behaviour for the Next Destination Prediction". In: *IEEE Transactions on Intelligent Transportation Systems* 21.7 (2020), pp. 2980–2989.

[109] Patrick Ebel, Ibrahim Emre Göl, Christoph Lingenfelder, and Andreas Vogelsang. "Destination Prediction Based on Partial Trajectory Data". In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1149–1155.

[110] Hoang Thanh Lam, Ernesto Diaz-Aviles, Alessandra Pascale, Yiannis Gkoufas, and Bei Chen. "(Blue) Taxi Destination and Trip Time Prediction from Partial Trajectories". In: *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge*. 2015, pp. 63–74.

[111] Chengwu Liao, Chao Chen, Chaocan Xiang, Hongyu Huang, Hong Xie, and Songtao Guo. "Taxi-Passenger's Destination Prediction via GPS Embedding and Attention-Based BiLSTM Model". In: *IEEE TITS* 23.5 (2022), pp. 4460–4473.

[112] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "Next Place Prediction Using Mobility Markov Chains". In: *Proceedings of the First Workshop on Measurement, Privacy, and Mobility*. MPM '12. 2012. ISBN: 9781450311632.

[113] M. Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[114] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective approaches to attention-based neural machine translation". In: *arXiv preprint arXiv:1508.04025* (2015).

[115] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014, pp. 1724–1734.

[116] Bangchao Deng, Dingqi Yang, Bingqing Qu, Benjamin Fankhauser, and Philippe Cudre-Mauroux. "Robust Location Prediction over Sparse Spatiotemporal Trajectory Data: Flashback to the Right Moment!" In: *ACM Trans. Intell. Syst. Technol.* 14.5 (Sept. 2023). ISSN: 2157-6904. DOI: 10.1145/3616541. URL: https://doi.org/10.1145/3616541.

[117] Jingyuan Wang, Jiawei Jiang, Wenjun Jiang, Chao Li, and Wayne Xin Zhao. "LibCity: An Open Library for Traffic Prediction". In: *Proceedings of the 29th International Conference on Advances in Geographic Information Systems*. SIGSPATIAL

'21. Beijing, China: Association for Computing Machinery, 2021, pp. 145–148. ISBN: 9781450386647. DOI: 10.1145/3474717.3483923. URL: https://doi.org/10.1145/3474717.3483923.

[118] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. "Revealing the Dark Secrets of BERT". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Ed. by Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4365–4374. DOI: 10.18653/v1/D19-1445. URL: https://aclanthology.org/D19-1445.

[119] Hossein Amiri, Shiyang Ruan, Joon-Seok Kim, Hyunjee Jin, Hamdi Kavak, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Zufle. "Massive Trajectory Data Based on Patterns of Life". In: *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*. SIGSPATIAL '23. Hamburg, Germany: Association for Computing Machinery, 2023. ISBN: 9798400701689. DOI: 10.1145/3589132.3625592. URL: https://doi.org/10.1145/3589132.3625592.

[120] Bangchao Deng, Dingqi Yang, Bingqing Qu, Benjamin Fankhauser, and Philippe Cudre-Mauroux. "Robust Location Prediction over Sparse Spatiotemporal Trajectory Data: Flashback to the Right Moment!" In: *ACM Transactions on Intelligent Systems and Technology* 14.5 (2023), pp. 1–24.

[121] Jiaxuan You, Tianyu Du, and Jure Leskovec. "ROLAND: graph learning framework for dynamic graphs". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 2358–2366.

[122] Kaike Zhang, Qi Cao, Gaolin Fang, Bingbing Xu, Hongjian Zou, Huawei Shen, and Xueqi Cheng. "DyTed: Disentangled Representation Learning for Discrete-time Dynamic Graph". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '23. Long Beach, CA, USA: Association for Computing Machinery, 2023, pp. 3309–3320. ISBN: 9798400701030. DOI: 10.1145/3580305.3599319. URL: https://doi.org/10.1145/3580305.3599319.

[123] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. "Adaptive graph convolutional recurrent network for traffic forecasting". In: *Advances in neural information processing systems* 33 (2020), pp. 17804–17815.

[124] Benedek Rozemberczki, Paul Scherer, Yixuan He, George Panagopoulos, Alexander Riedel, Maria Astefanoaei, Oliver Kiss, Ferenc Beres, Guzman Lopez, Nicolas Collignon, et al. "Pytorch geometric temporal: Spatiotemporal signal processing with neural machine learning models". In: *Proceedings of the 30th ACM international conference on information & knowledge management*. 2021, pp. 4564–4573.

[125] Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. "A survey on map-matching algorithms". In: *Databases Theory and Applications: 31st Australasian Database Conference, ADC 2020, Melbourne, VIC, Australia, February 3–7, 2020, Proceedings 31*. Springer. 2020, pp. 121–133.

Appendices

A Broader Impact

Trajectory prediction holds transformative potential in a variety of sectors by enabling more accurate and timely predictions of movement patterns. Our work contributes to this domain by introducing a framework that integrates advanced trajectory forecasting with dynamic network construction. This integrated approach opens up a multitude of opportunities across diverse fields, with far-reaching consequences for society, technology, and industry.

One of the key applications of trajectory prediction lies in navigation and route optimization. By accurately forecasting the future positions of moving objects, our framework can support real-time navigation systems and vehicle routing applications. Enhanced route optimization reduces travel time, lowers fuel consumption, and minimizes traffic congestion. In practical terms, these improvements translate into economic benefits—such as reduced operating costs for logistics companies—and environmental benefits, including lower greenhouse gas emissions.

Our research also has major implications for urban planning and geospatial analysis. Urban planners increasingly rely on high-quality mobility data to understand traffic patterns, assess public transportation usage, and determine pedestrian flow in crowded city centers. With precise trajectory forecasts, planners are empowered to design more efficient transportation systems, optimize signal timings, and improve the layout of public spaces. In this way, our framework not only contributes to academic advancements in spatial analysis but also directly informs strategies for urban redevelopment and sustainable city design.

Moreover, trajectory prediction underpins many location-based services and personalized recommendations. Businesses, ranging from retail companies to service providers, can leverage accurate mobility predictions to offer tailored recommendations, such as suggesting nearby restaurants, events, or retail outlets based on an individual's movement. This level of

personalization enhances user satisfaction and can also drive business growth by connecting consumers with relevant services. The integration of predictive models into mobile applications thus has the potential to revolutionize how consumers engage with their environments.

Beyond commercial applications, trajectory prediction plays a vital role in public safety and emergency response. For example, accurate mobility forecasting supports intelligent transportation systems and autonomous vehicle technology by improving collision avoidance and route planning under dynamic conditions. In contexts such as disaster management or epidemic control, knowing where individuals are likely to move can help authorities deploy resources more efficiently, plan evacuation routes, and assess risk in real time. Such applications have a direct impact on public health and safety, ultimately saving lives and reducing economic losses during critical events.

Furthermore, by enabling the prediction of future mobility networks, our framework has the potential to inform the design of more resilient infrastructures. Urban systems that are capable of forecasting network dynamics can better withstand disruptions—whether caused by natural disasters, infrastructure failures, or public health crises. For instance, during an epidemic outbreak, real-time mobility network forecasts could help predict the spread of infection, thereby guiding interventions such as targeted lockdowns or vaccination campaigns. In this context, our work contributes to both public health preparedness and the creation of safer, more responsive urban environments.

Finally, our framework also has implications for the development and optimization of data-driven policymaking. By integrating disparate mobility data sources and deploying sophisticated predictive models, policymakers are equipped with actionable insights into real-world movement patterns. This can lead to the design of evidence-based interventions in sectors such as transportation, public safety, and urban planning, ultimately fostering economic growth and societal well-being.

B Ethical Implications

As with all applications of artificial intelligence and data analytics, the use of deep generative models for trajectory prediction and mobility network forecasting raises a number of ethical considerations that must be addressed. The implications of our work extend beyond technical performance to issues of privacy, fairness, and responsible use. In this section, we discuss these ethical challenges and the measures we have taken to mitigate potential negative impacts.

Privacy and Data Protection. Trajectory data inherently contains sensitive information about individual movements, which can lead to significant privacy concerns. To mitigate these concerns, every dataset used in our experimental evaluations has been carefully anonymized prior to analysis, and all work is conducted in accordance with ethical guidelines and data protection laws. Moreover, the data sources we employ are publicly available and have been collected with informed consent. By strictly adhering to the terms and conditions set by the dataset curators, we ensure that our research respects individual privacy.

Fairness and Biases in Trajectory Datasets. The quality and composition of trajectory datasets may exhibit inherent biases, as many existing datasets do not represent all demographic or geographic segments equally. Such biases have broader implications when these predictive systems are deployed in real-world settings, potentially exacerbating inequalities. While our current study primarily focuses on the technical challenges of trajectory prediction and mobility network forecasting, we acknowledge that bias in data is a critical concern. Future research should include thorough bias assessments and employ strategies to mitigate these biases to ensure that predictive systems are equitable and fair for all users.

Transparency and Accountability. The use of complex deep learning models inherently reduces interpretability. Although we have taken steps to enhance transparency, such as analyzing attention weights and providing interpretability studies, the "black-box" nature of

these models remains a challenge. It is ethically important to provide users, especially those affected by decisions guided by these predictions, with clear and understandable explanations of how predictions are made. In our work, we have made a concerted effort to interpret the internal workings of our models. However, real-world deployments should further prioritize transparency and accountability, ensuring that stakeholders can trust and verify model outputs.