TRAJECTORY PREDICTION LEARNING USING DEEP GENERATIVE MODELS

JING LI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTERS OF SCIENCE

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

NOV 2023

Abstract

Trajectory prediction refers to the problem of estimating or forecasting the future path or trajectory of an object or entity based on its current state and historical data. This can have various applications in different domains, such as autonomous vehicles, robotics, human motion analysis, and more. Recently, deep learning techniques have been proposed for trajectory prediction learning that are trained on historical trajectory data to learn patterns and relationships, enabling them to make predictions about future trajectories based on past observations. However, these models encounter challenges in handling complex spatial dependencies because of the complexity of trajectory data and the dynamic nature of the environment which makes it harder to make accurate predictions. To address these limitations, we propose TrajLearn, a novel model for trajectory prediction that leverages generative models of higher-order mobility flow representations (hexagons). Given the recent history of a trajectory and its current state as input, our model accurately predicts the next k steps (hexagons) of its future path. Our model incorporates a variant of beam search to simultaneously explore multiple candidate paths while respecting spatial constraints for path continuity. We show that our model outperforms the state-of-the-art method and other sensible baselines, by as much as $\sim$60% on several real-world trajectory datasets. We also experiment for varying values of k, which determines the prediction horizon of the trajectory. Furthermore, we conduct a resolution sensitivity analysis, and an ablation study to determine the impact of the various components to the model's performance.

Acknowledgements

First and foremost, I express my deepest gratitude to my supervisor, Dr. Manos Papagelis, whose expertise, understanding, and patience, added considerably to my graduate experience. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. His encouragement and genuine support gave me the strength and confidence to overcome the challenges I faced during my studies. His guidance was invaluable in writing this thesis.

I would also like to thank my thesis committee members, Dr. Ruth Urner and Dr. Costas Armenakis, for their insightful comments and suggestions. I must also extend my gratitude to my lab mates Ali Faraji and Amirhossein Nadiri. From brainstorming to discussions, their insights and friendship have made my time in the lab both productive and enjoyable. This thesis is the culmination of a collaborative effort with them, whose invaluable contributions and teamwork were essential to its success.

Lastly, I am particularly grateful to my family. Their sacrifices, love, and belief in me have been the bedrock of my resilience and persistence. This journey has been not only an academic pursuit but a life-changing experience, and I am deeply grateful to everyone who played a part in it.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The development of tracking and geolocation technology has facilitated the collection of large-scale mobility data, encompassing both objects and individuals [1, 2]. Mining interesting patterns in mobility data is of increased research and development interest due to a wide range of practical applications. Technical problems in the area include trajectory classification, clustering, prediction, simplification, and anomaly detection [3, 4, 5].

1.1 Problem of Interest

In this research, we focus on the *trajectory prediction problem*, which refers to the task of predicting the future path or trajectory of an object (or individual) based on its current state and historical data. Efficient methods for trajectory prediction are highly desirable in various domains and applications, including transportation systems, human mobility studies, autonomous vehicles, robotics, and more.

1.2 Current Approaches and Limitations

First attempts to address the problem considered statistical methods, such as matrix factorization [6, 7, 8] and Markov chain [9, 10, 11]. However, these methods frequently encounter challenges in capturing the intricate sequential and periodic characteristics of trajectories. Recent progress in deep learning has led to the emergence of deep neural models explicitly tailored to capture the sequential characteristics inherent in trajectories. Notably, approaches centered around Recurrent Neural Networks (RNNs) have exhibited promising results [12, 13]. However, despite their favorable performance, these models encounter challenges when confronted with sparse and imprecise trajectory data. Additionally, they often demand substantial quantities of meticulously labeled training data, a resource-intensive and time-consuming endeavor. Furthermore, there is a risk of overfitting to the training dataset, resulting in suboptimal generalization capabilities when faced with unseen data. Moreover, their primary focus lies in Next Point of Interest (POI) prediction, a pertinent but distinct problem from the one at hand. Certain prior studies have explored trajectory prediction using comprehensive GPS log datasets, as opposed to relying on POI check-ins [14, 15, 16]. However, these studies address specialized versions of the problem, which lack effective generalization to the broader issue at hand.

1.3 Our Contributions

To address these limitations, we propose a novel approach that leverages deep generative models trained (from scratch) on large-scale historical higher-order trajectory data to accurately predict the future path of an object based on its current state. Our contributions can be summarized as follows:

- We present a method that given a GPS log trajectory data set of moving objects generates

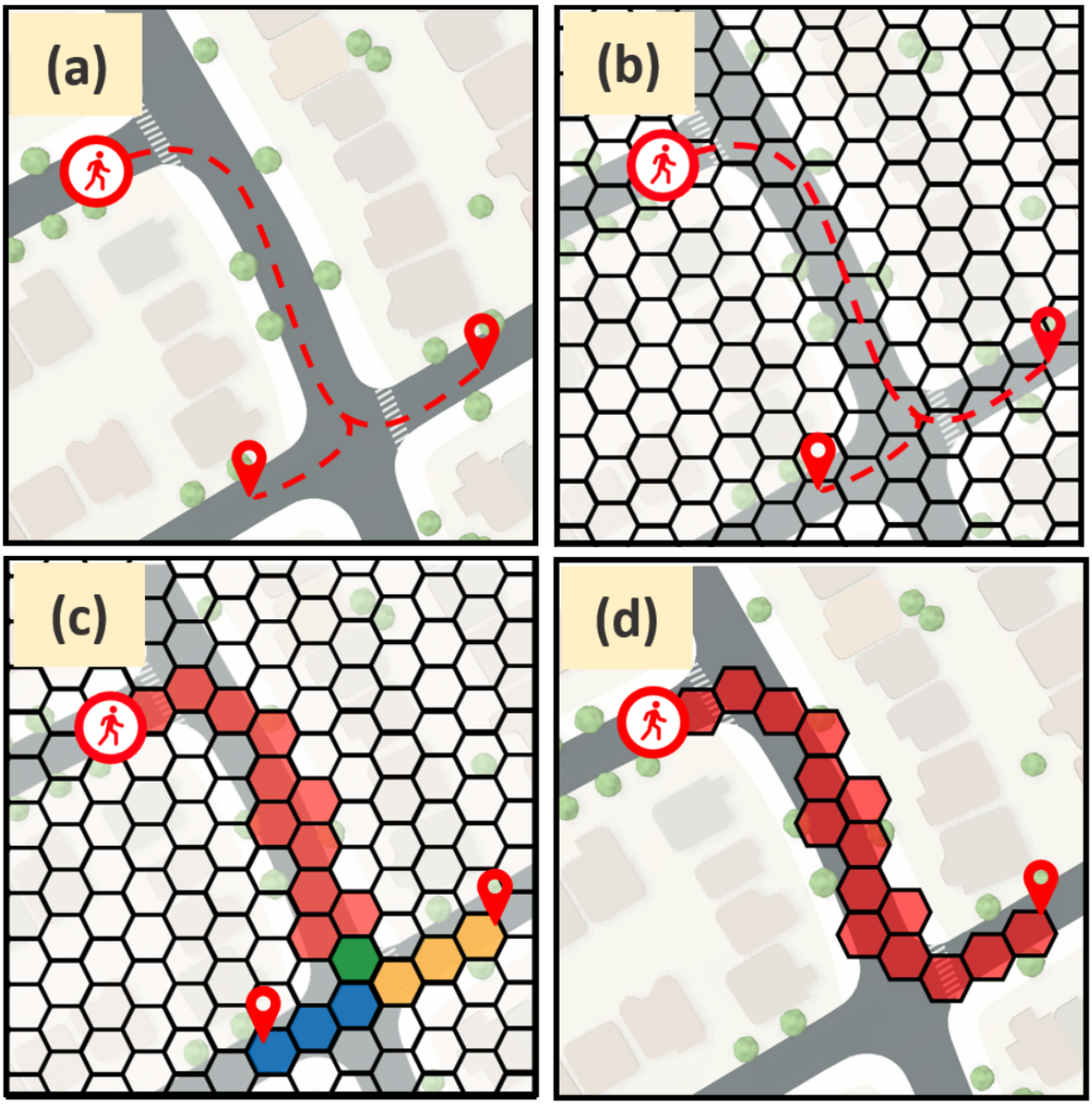


Figure 1.1: Illustrative example of the trajectory prediction problem using higher-order mobility flow representations (hexagons); (*a*) there are two possible trajectories for the pedestrian, (*b*) trajectories are represented on an hexagon-based tesselated map, (*c*) given the historical data (red) and the current location (green), two possible trajectories are predicted (blue and orange), (*d*) the actual trajectory followed.

higher-order mobility flow data, where trajectories are represented as sequences of hexagons defined on a hexagonal map tessellation. The method is versatile and capable

of constructing mobility flow data at varying map resolutions, thereby facilitating distinct levels of analysis.

- We propose TrajLearn, a trajectory generative model based on the Transformer architecture [17] and inspired by GPT-2 [18]. TrajLearn is trained on historical trajectory data provided in the form of higher-order mobility flow data, and is able to make accurate predictions about future trajectories based on past observations (see example in Figure 1.1).

- We demonstrate empirically that TrajLearn outperforms the state-of-the-art methods and sensible baselines on different evaluation metrics. We also perform an ablation study and a parameter sensitivity analysis that evaluate the impact of the different parameters in the performance of TrajLearn.

- We make our source code publicly available to facilitate the reproducibility of our work.

1.3.1 Publications

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

- Ali Faraji[1], Jing Li[1], Gian Alix, Mahmoud Alsaeed, Nina Yanin, Amirhossein Nadiri, Manos Papagelis, "Point2Hex: Higher-order Mobility Flow Data and Resources", Proceedings of the 31st International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2023) [19].

- Amirhossein Nadiri, Ali Faraji, Jing Li, Manos Papagelis, "TrajLearn: Leveraging Generative Models for Trajectory Prediction Learning", submitted.

[1]These authors contributed equally to this work.

1.3.2 Co-authorship Statement

In the aforementioned publications, my contributions were made in collaboration with Ali Faraji and Amirhossein Nadiri and included the following aspects: formulating the initial research concept, conducting a comprehensive literature review, executing experiments along with analyzing their outcomes (which involved developing necessary tools), drafting the papers, and contributing significantly to the final versions of the manuscripts.

1.4 Thesis Organization

For the remainder of this thesis, we first provide in Chapter 2 a thorough review of related works in the literature that are relevant to the study of interest. Then, in Chapter 3, we introduce some preliminaries of the problem, including the common notations and definitions throughout this thesis. In the same chapter, we also formalize the problems that we aim to address. Since our approach relies on higher-order mobility flow dataset, wherein trajectories are represented as sequences of hexagons on a tessellated map, Chapter 4 gives a more in-depth discussion of working with higher-order mobility flow data, and its advantages. Additionally, we update the problem definition to accommodate the new data representation. In Chapter 5 we delves into the specifics of the trajectory prediction model. Then we describe the design of the experiments in Chapter 6, including the datasets used, the evaluation metrics, the various baseline models, the results of the experiments, insightful discussions based on the observed results, and other details relevant to the experimental design. Finally, we end the thesis with some concluding remarks in Chapter 7.

Chapter 2

Related work

In this chapter, we first review the trajectory data mining in both statistical and deep-learning-based methods. Then we delve into trajectory prediction problem in different research fields. Finally we review the most significant efforts relevant to deep generative models.

2.1 Trajectory Data Mining

Trajectory data contain huge amount of information about moving objects, encouraging many studies and a variety of downstream applications such as constructing location-based social networks, predicting traffic dynamics and planning urban services [3]. Next we discuss statistical and deep learning models in trajectory data mining based on different mining tasks. As trajectory prediction is the primary focus of the this thesis, we discuss it in more detail in the next section.

2.1.1 Statistical Models

Pattern Mining. In tradition, there are several basic mining tasks that have been explored within this area. Pattern mining of trajectory data is an important domain that includes

clustering, frequent sequence mining [20], periodic pattern mining [21] and moving together pattern mining [22]. Specifically, clustering is to divide trajectory data in different clusters based on the movement properties. A general clustering approach is to represent each trajectory with a feature vector, and then, measure the similarity between trajectories by calculating the distance between their feature vectors [23, 24]. Lee et al. [25] propose a partition-and-group-based framework for clustering trajectories. Trajectories are divided into multiple line segments and similar line segments are grouped together into a cluster. Yan et al.[26] present a model that is hybrid of spatiotemporal and semantic information of trajectories. The model encapsulates both the geometry and semantics of mobility data, which can be used in trajectory clustering. Li et al. [27] propose a road-network-based clustering algorithm. Instead of clustering the moving objects, road segments are clustered based on the density of common traffic they share. Similarly, Han et al. [28] introduce NEAT, a method designed for efficiently clustering spatial trajectories of mobile objects on road networks. This model considers three key aspects: the physical constraints of the road network, the network proximity and the traffic flows among consecutive road segments, to effectively group trajectories into spatial clusters. Pechlivanoglou and Papagelis [29] present an one-pass sweep-line algorithm over the trajectories (SLOT) that can effectively compute the node importance of moving objects in a trajectory network. Sawas, Abuolaim, Afifi and Papagelis [30] propose a novel versatile method, *timeWgroups*, for efficient discovery of pedestrian groups dynamics, including various definitions of group gathering and dispersion.

Classification. Classification is another classical spatiotemporal data mining task. Zheng et al. [31] apply a decision tree-based inference model to classify different transportation modes based on GPS data. Bolbol et al.[32] use SVMs for transportation mode classification. Another sub-domain of classification is trajectory outlier or anomaly detection, which involves the detection of significantly anomalous observations compare to the majority of the datasets. One of the approaches are to partition trajectories into a set of sub-trajectories and then

detect the outlying sub-trajectories by applying a distance function or clustering approach. For example, Lee et al. [33] propose a Partition-and-Detect framework. The model includes a two-level trajectory partitioning strategy for the partition phase, which improves quality and efficiency. Additionally, for the detection phase, they introduce a method that use both distance-based and density-based approaches.

2.1.2 Deep Learning-based Models

[34] investigates the distribution of the trajectory data mining problems addressed by deep learning. Over 70 percent studies are predictive learning, and others are inference and anomaly detection, and so on. Next, we discuss deep learning models based on different tasks.

Graph-based Prediction Models. In some cases, graphs are constructed from trajectory data. For example, in road network-scale traffic prediction, the transportation network can be naturally modeled as a graph, making it suitable to apply graph-based deep learning models. Chai et al.[35] propose a multi-graph convolutional neural network (CNN) model to predict bike flow. More specifically, the authors generate multiple graphs that reflect heterogeneous inter-station relationship between bike stations. These graphs are then fused and convolutional layers are used to predict station-level future bike flow. Li et al.[36] present a model called the Diffusion Convolutional Recurrent Neural Network (DCRNN) for traffic forecasting. In their paper traffic flow on a transportation network is viewed as information that propagates through a graph, and the model employs bidirectional random walks to capture spatial dependencies among graph nodes and an encoder-decoder architecture with scheduled sampling to capture temporal dependencies. Zhang et al.[37] propose TrafficGAN, an adversarial learning framework to predict traffic flow. To capture the spatial and temporal correlations among roads in a road network, CNN and LSTM are embedded in the framework.

Estimation and Inference. There have been a lot of studies about estimation and inference,

which mainly focus on the problem of estimating travel time, trip purpose and travel mode from the mobility trajectory data. Accurately estimating the duration of a trip based on its origin and destination locations and departure time is essential in numerous real-world applications. To tackle this problem, Li et al.[38] present a deep multi-task representation learning model that estimates the arrival time of a trip. The model utilizes the underlying road network and spatiotemporal knowledge to generate trip representations that retain various trip properties. Fu et al.[39] develop CompactETA, a novel ETA learning system that can provide accurate online travel time inference. Their approach leverages a spatiotemporal weighted road network graph and applies a graph attention network to encode high-order spatial and temporal dependencies into trip representations. In addition, they encode the sequential information of the travel route using positional encoding. Martin et al. [40] study the problem of inferring the purpose of a user's visit to a specific location using trajectory data. To address this issue, they propose a Graph Convolutional Neural Network (GCN) for inferring activity types from GPS trajectory data generated by personal smartphones. The mobility graphs of a user, which are fed into GCNs, are generated based on all their activity areas and the edges are constructed based on trajectory data. Zhu et al.[41] propose a Semi-supervised Federated Learning (SSFL) framework that includes a novel identification module called Convolutional Neural Network-Gated Recurrent Unit, which can accurately infer travel modes from GPS trajectories. In addition, the authors design a pseudo-labeling method that allows to set pseudo-labels on their local unlabeled datasets using a small public dataset on the server. Alix and Papagelis [42] present a reinforcement learning method PathletRL for constructing a small set of basic building blocks that can represent a wide range of trajectories. This dictionary can be useful in various tasks and applications, such as trajectory compression, travel time estimation, route planning, and so on.

Anomaly Detection. The other important task for trajectory data is anomaly detection or outlier detection, which aims to identify the rare observations that differ remarkably from the

majority of the data. Chen et al.[43] collect huge amount of traffic accident data and users' GPS records to study the influence of human mobility on traffic accident risk. They propose a deep model of Stack denoise Autoencoder to learn hierarchical feature representations of human mobility, which can be utilized to predict traffic accident risk level. Deng et al.[44] propose a generative adversarial network that leverages graph convolutional gated recurrent unit (GCGRU) to help the generator and discriminator learn the spatiotemporal features of traffic dynamics and traffic anomalies respectively. After adversarial training, both the generator and discriminator can act as separate detectors. The generator learns to model normal traffic patterns, while the discriminator provides detection criteria based on spatiotemporal features.

2.2 Trajectory Prediction

Predicting trajectories has been explored by different areas of computing, including (a) *computer vision*, and (b) *mobile data analysis*.

2.2.1 Computer Vision

Autonomous Driving. Trajectory prediction involves predicting the future path or movement of objects in a scene over time. For this, they rely on camera-generated video frames, where trajectories can be represented by (x, y) coordinates within the frame [45, 46, 47]. One of the very popular focuses in academia and industry on trajectory prediction is for autonomous driving. For instance, Ren et al. [48] develop a deep learning model tailored for forecasting an occupancy map showing the earliest possible time each location may be occupied by both seen and unseen vehicles. They validate their model using the extensive nuScenes autonomous driving dataset [49], which combines 6 cameras, 5 radars and 1 lidar installed on vehicles. And the model proposed by Song et al.[50] is to tackle the vehicle

prediction problem in the multi-agent setting on highway datasets such as HighD [51], where camera-equipped drones are used to measure every vehicle's position and movements from an aerial perspective.

Pedestrian Mobility. The other popular research area for trajectory prediction in computer vision is pedestrian mobility prediction at a small scale, within the camera angles. Duan et al.[52] develop a dual-path architecture to identify both common and unusual patterns in spatial and temporal dimensions for pedestrian movement prediction. And Sun et al.[53] propose a method to predict human future movements using only two frames for observation. Both methods have been conducted on ETH [54] and UCY [55] dataset wherein video frames were recorded from birds-eye view at various locations. In addition to the dataset format, the methods are designed to detect or predict human–human and human–vehicle interactions [45, 56], for example, potential collisions.

In summary, when we consider the domain of trajectory prediction within computer vision, it becomes apparent that the scope of this current work diverges significantly in several key aspects. Notably, there is a distinct difference in the format of the data utilized, the predictive range as well as in the specific interests and objectives of the problem being addressed. Consequently, due to these fundamental discrepancies, this particular line of research does not align with, and thus falls outside, the scope of our present study.

2.2.2 Mobile Data Analysis

In this section, trajectory prediction involves predicting the future path of objects, given historical trajectory data represented as GPS geo-coordinates over time. Two types of predictive analysis can be identified: *macroscopic* and *microscopic*.

Macroscopic analysis. Macroscopic analysis focuses on high-level mobility models for crowd flow prediction [57], traffic flow prediction [58], taxi demand prediction [59], and city-wide

mobility prediction [60, 61]. In greater detail, Lin et al.[57] develop a deep learning-based convolutional model to predict the inflow and outflow of each region in the metropolis, given the historical flow data. The model captures the the spatial dependence among crowd flows in different regions and it also leverages POI distributions and time factor that affect the crowd movements. Ji et al.[62] present a self-supervised learning framework to model spatial and temporal heterogeneity in traffic flow prediction. Yao et al.[59] harness the power of CNN and LSTM in a joint model to capture spatial and temporal relations among regions to better predict taxi demands and similarly Zhou et al.[63] employ an encoder-decoder framework to predict multi-step citywide passenger demands.

These models are important as they provide aggregated insights to guide solutions for urban planning at a city-level. However, in contrast to our current research, these models primarily aim to forecast the flow of traffic across various regions. This is typically achieved by dividing a map into grids or specific areas. In contrast, our research is dedicated to understanding and analyzing the mobility patterns of individual entities. Our problem focus on the individual-level mobility prediction and it involves predicting the future trajectories of individuals based on their historical mobility data. This kind of prediction can have applications in various domains such as individual transportation planning, urban development, personalized services, and targeted marketing.

Microscopic Analysis. Microscopic analysis focuses on object-level mobility prediction, that is, to predict the mobility patterns of individuals. First attempts to address the problem considered statistical methods, such as matrix factorization [6, 7, 8] and Markov chain [9, 10, 11]. For instance, Cheng et al. [6] include social information and geographical influence into a generalized matrix factorization framework for personalized location recommendation. In a similar way, Lian et al. [8] incorporate spatial clustering in human mobility and leveraged the weighted matrix factorization technique. Mathew et al. [10] introduce a hybrid approach

that involves clustering location histories based on their attributes and subsequently training individual Hidden Markov Models (HMMs) for each cluster to forecast human mobility. However, these approaches often struggle to capture the complex sequential and periodic features of human mobility in trajectories.

Recent progress in the field of deep learning has led to the development of specialized deep neural network models that are adept at capturing the sequential nature of trajectory data. In particular, approaches employing Recurrent Neural Networks (RNNs) have shown notable effectiveness. For instance, Liu et al. [12] introduce ST-RNN, a variant of RNN designed to simultaneously consider both temporal and spatial contexts. Similarly, Feng et al. in their research [13] develop DeepMove, an attentional recurrent network-based model for predicting human movement patterns. Despite their impressive performance, these models face challenges in dealing with trajectory data that is sparse or inaccurately recorded. In order to address the sparsity problem of spatiotemporal information, Lian et al.[64] introduce GeoSAN, a Geography-aware sequential recommendation system. This system highlights the importance of using informative negative samples and employs a self-attention based geographic encoder to represent the hierarchical grid structure of each GPS coordinate. Yang et al.[65] introduce Flashback, a RNN-based model specifically crafted for handling sparse user mobility data. They uniquely search back the historical hidden states that share similar spatiotemporal contexts to the current one for predicting the next location. Luo et al.[66] propose STAN that considers all relevant visits from user trajectory and incorporates spatiotemporal correlation between non-adjacent locations and non-contiguous visits for location recommendation.

However, these models are mostly designed for the POI prediction, which is a pertinent but distinct problem from the current work. Specifically, the inputs are records of users' historical check-ins and the outputs are most plausible POI candidates of the future. In this thesis,

instead of only predicting POIs, we deal with higher-order mobility flow of individuals and predict next k hexagons within a map. In other words, we do not limit locations to POIs, it can be any hexagons of a tessellated map.

Some previous work study the trajectory prediction based on general GPS log dataset rather than POI check-ins. For instance, Jiang et al. [14] employ a seq2seq model to analyze human trajectories and made predictions about future movements of individuals. However, the key idea of their study is that human's movements could be irrelevant to their past movements. In particular, it focuses on the very short-term mobility prediction based on big rare events or disasters such as large earthquakes or severe traffic accidents. Sadri et al. [15] present a framework for continuous trajectory prediction. They particularly focused on predicting a user's afternoon trajectory, given their morning trajectory. Therefore, it heavily relies on a single historical record of an individual. Amichi et al. [16] propose to first predict the type of next movement, i.e., the purpose of visiting the next location. Based on this, the predictor outputs the next location or zone where the individual would be. The location or zone are defined by projecting the GPS coordinates to spatial grid IDs of a gridded map, where locations are fine-grained while zones are more coarse-grained. Nevertheless, these prior research primary focus on specific instances of the problem and their models are insufficient in addressing the general trajectory prediction problem.

2.3 Deep Generative Models

Generative models are a class of machine learning models designed to learn and mimic the underlying data distribution of a given dataset (see Bond-Taylor et al. [67] for a comprehensive survey). The fundamental idea behind generative models is to capture the statistical patterns and relationships present in the data so that the model can generate new samples that look and feel like the real data. They have shown remarkable results in creating realistic data

samples, and they have been used for various creative and practical applications including image synthesis, text generation, music composition, and more. There are several types of generative models, but some of the most prominent ones include Generative Adversarial Networks (GANs) [68], Variational Autoencoders (VAEs) [69], Autoregressive Models [70], Flow-based Models [71], and Transformers [17]. In our research, we propose the use of generative models for addressing the trajectory prediction problem. We specifically employ a transformer-based architecture such as GPT (Generative Pre-trained Transformer) [72]. To the best of our knowledge, this is the first attempt to employ deep generative models to the (non vision-based) trajectory prediction problem.

2.3.1 Transformer-based Models

Transformer models [17] are primarily known for their use in natural language processing (NLP) tasks, but can also be used for generating text and sequences in other domains. Traditionally, sequence data was processed using recurrent neural networks (RNNs) [73], which suffered from limitations like vanishing gradients and sequential computation, making them slow to train. The Transformer architecture addressed these issues by employing a self-attention mechanism, which allows for parallelization and efficient learning of long-range dependencies in the data. In our research, we treat historical trajectories as sequences of hexagons on a hexagon-based tessellated map. These sequences resemble statements of a language, and therefore can serve as input to a Transformer-based generative model. Note as well that although we largely followed the GPT-2 architecture, our method is generic and any similar large language model architecture could be utilized, such as PaLM [74], and LLaMA [75], to name a few.

Chapter 3

Problem

In this chapter, we first introduce notations and preliminaries related to our research. Then, we formally define the problem of interest.

3.1 Preliminaries

Definition 3.1 (Map). A map $\mathcal{M}$ represents the administrative boundaries of a finite and continuous geographic area of Earth, such as a city. Since $\mathcal{M}$ represents a relatively small region, the curvature of the Earth's surface within this area is negligible, allowing us to approximate $\mathcal{M}$ as a finite 2-dimensional Euclidean space $\mathbb{R}^2$.

Definition 3.2 (Trajectory). A trajectory represents the movement of an object or entity over time. It consists of spatiotemporal points, each of which includes a time identification t and a location identification l. The trajectory sequence is denoted as $T = p_1 p_2 ... p_n$, where p_i represents a spatiotemporal point.

Definition 3.3 (Partial trajectory). Given a trajectory T of an individual i, a partial trajectory of length l is a subsequence $T_i^l = p_{i_1} p_{i_2} ... p_{i_l}$ of T, where l is the l'th spatiotemporal point in T.

Symbol	Description
$\mathcal{M}$	The map of a geographic area
T	Trajectory sequence of spatiotemporal points
p_i	Spatiotemporal point in the trajectory sequence
T^l	Trajectory history of length l
k	Prediction horizon (# trajectory steps to predict)
l	Input trajectory length
$\mathcal{B}$	Set of hexagonal blocks (or hexagons) forming a tessellation of $\mathcal{M}$

Table 3.1: Summary of notations.

Definition 3.4 (Trajectory history). The trajectory history of a specific length T^l, encompasses all the previously occurred partial trajectories of length l.

Definition 3.5 (Prediction horizon). The prediction horizon k defines the number of future trajectory steps to be predicted.

3.2 Problem Definition

We are now in position to formally define the problem of estimating or forecasting the future path or trajectory of an object or entity based on its current state and historical data.

Problem 1 (Trajectory Prediction). Given a map $\mathcal{M}$, the corresponding trajectory history S^l, a partial trajectory $T = p_{i_1}p_{i_2}...p_{i_l}$, and an integer $k > 0$, the objective is to predict the next k spatiotemporal points $p_{i_{l+1}}, \ldots, p_{i_{l+k}}$ of the partial trajectory T.

Note that we currently only state the general trajectory prediction problem. Once we introduce the idea of higher-order mobility flow, we will revisit the trajectory prediction problem and formalize it in the context of predicting the next k hexagons (see section 4.6).

Chapter 4

Higher-order Mobility Flow Data

In this section, we provide a rationale of working with higher-order mobility flow data, and its advantages. Additionally, we update the problem definition to accommodate the new data representation.

4.1 Motivation

While some mobility datasets are available for research purposes, the majority of them are typically discovered through ad-hoc searches and lack comprehensive documentation. Furthermore, researchers frequently resort to generating their own synthetic datasets, which are often unpublished and lack details of their generation process and potential reproducibility. Moreover, working with mobility data presents its own challenges, due to its large size, high sparsity, and complexity. To address these problems, there has been a growing interest in a new type of mobility data, describing trajectories using *higher-order geometric elements* [76]. Higher order means higher hierarchy or higher level(s) of abstraction for representing trajectories. First, a map is uniformly tessellated using a geometric element, say hexagons, and then trajectories are represented as sequences of hexagons. Fig. 4.1 shows an overview

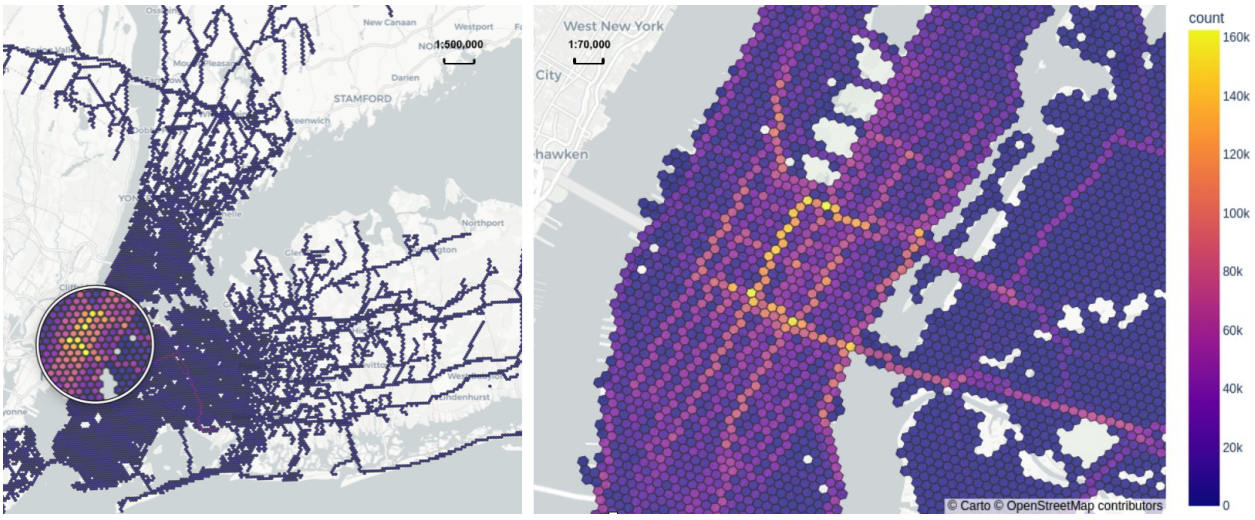


Figure 4.1: An example of hexagon-sequence trajectories on a tessellated map.

of trajectories of taxis represented by sequences of hexagons on a tessellated map of New York City. The representation provides some advantages: it decreases sparsity and allows for analysis at varying granularity based on map tessellation; it is compatible with well-known machine learning models; and it promotes better generalization, minimizes overfitting, and supports effective visual analytics, such as heatmaps.

4.2 Higher-Order Trajectory Representation

Rationale. Working with raw trajectory datasets (either in the form of GPS data points or POI check-ins) is challenging because (longitude, latitude) geo-coordinates: (*i*) are sparse and large amounts are needed to learn meaningful relationships, and (*ii*) are not very compatible (as input) to popular machine learning architectures, due to their continuous nature. We therefore propose to resort to a higher level of abstraction for representing trajectories. This transformation is done by first obtaining the routes by connecting raw trajectory data points through publicly available routing algorithms[2]. Then, the trajectory is represented as a

[2] `https://developers.google.com/maps/documentation/directions`

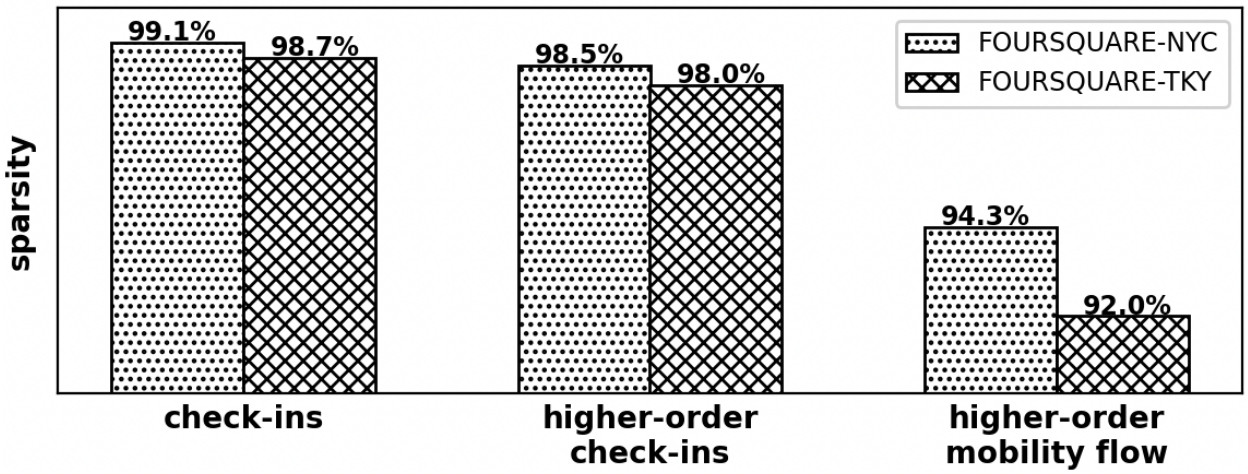


Figure 4.2: Impact of higher-order abstraction on sparsity.

sequence of the higher-order elements (hexagons) traversed by the route.

Advantages. The benefits of such a transformation are multi-fold. First, the sparsity will be decreased, as multiple data points (or checkins) will belong to the same higher abstraction element (e.g., hexagons). Specifically, sparsity is defined by the number of zeros in a user-POI or user-Hex matrix. In a user-Hex matrix, an entry is equal to `1` if the user has visited any place in the corresponding hexagon and `0` otherwise. Multiple POIs can be located within the same hexagon, leading to a decrease in sparsity. For example, consider the simple scenario where there are three users and three POIs, forming a 3×3 user-POI matrix (of nine elements/entries). Now assume all (three) users have visited the same single POI. In this case, the sparsity of the user-POI matrix will be 66.6% since six out of the nine elements will be `0` and only three of them will be `1`. If POI visits are projected to a higher dimension and assuming two (of the three) POIs fall in the same hexagon, then the user-Hex matrix will now have six entries (3 users $\times$ 2 hexagons). In this case, three of the elements/entriess will be equal to `1`, reducing the sparsity to 50%. Fig. 4.2 illustrates the impact of higher-order representations on decreasing the sparsity on two benchmark datasets (Foursquare-Nyc and Foursquare-Tky [77]). We observe about a 1% decrease in sparsity when using

higher-order check-ins, and more than a 5% decrease in the case of higher-order mobility flow. Furthermore, trajectories can now be treated as sequences of a predefined set of higher abstraction elements (e.g., hexagons), which are compatible (as input) to popular machine learning models, such as sequence models and Transformer-based models. In addition, the higher-order mobility flow data represents continuous routes/paths between location data points. While these routes are not representing the actual path an object has followed (this is unknown), they can capture implicit information and other semantics that lie in-between the different data points. The premise of operating on higher-order mobility data is that the deep learning models can avoid overfitting and generalize better as it simplifies the data in a way that omits specific details. We formalize these concepts below.

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

Definition 4.2 (POI). Let $\mathcal{P} = \{p_1, p_2, \ldots, p_{|P|}\}$ be a set of points of interests (POIs) on a map $\mathcal{M}$.

Definition 4.3 (Visits or Check-ins). In location-based services, a visit or check-in of a person to a location or place at a particular time is a record represented by a quadruplet $r = (u, l, t, \langle x, y \rangle)$, for user u, location ID l, time of visit t, and longitudinal-latitudinal tuple $\langle x, y \rangle$. We represent the set of all visits or check-ins by R. In this research, we use the term visit or check-in interchangeably.

Definition 4.4 (GPS Trace Points). A GPS trace point can similarly be defined using a triplet $s = (o, t, \langle x, y \rangle)$, for object o, timestamp t, and geocoordinate tuple $\langle x, y \rangle$. We denote the set of all traces by S.

Definition 4.5 (Trajectory). See the definition from Chapter 3.

Definition 4.6 (Map Tessellation). Let $\mathcal{B} = \{b_1, b_2, ..., b_n\}$ be a set of (regular) disjoint blocks that can fully tessellate map $\mathcal{M}$, forming a regular tiling. Each block $b_k \in \mathcal{B}$ is assumed to be a polygon. In our study, we opt for *hexagons*. The terms 'hexagons' and 'blocks' will

be used interchangeably throughout the thesis. An hexagon-based map tessellation offers several advantages over traditional square-based map tessellation (commonly known as a grid or tile system) [78]. Note as well that the tessellation can happen at different levels of resolution, by defining different sizes of the hexagons. The smaller the hexagon size, the higher the resolution.

Definition 4.7 (Higher-order check-ins). Since $\mathcal{M}$ is fully tessellated, each check-in/POI $p \in \mathcal{P}$ there is a $g \in \mathcal{G}$, s.t. p is in g.

Definition 4.8 (Higher-order GPS traces). Similarly, with $\mathcal{M}$ fully tessellated, for every $s \in S$ there is a $g \in \mathcal{G}$, such that s is in g.

Definition 4.9 (Higher-order Trajectory). Given a trajectory $T = p_1 p_2 ... p_n$, and since every point p_i belongs to a unique block $b_i \in \mathcal{B}$ of a tessellated map $\mathcal{M}$, we can translate every trajectory to a sequence of blocks. The outcome is a higher-order trajectory.

Definition 4.10 (Higher-order mobility flow). Given a higher-order trajectory $Tr = \{g_1, g_2, ..., g_m\}$, we can define a higher-order mobility flow as a new trajectory:

$$Tr = \{g_1, \mathcal{G}_{[1,2]}, g_2, \mathcal{G}_{[2,3]}, ..., \mathcal{G}_{[m-1,m]}, g_m\}$$

where each $\mathcal{G}_{[i,i+1]} \subset \mathcal{G}$, for $i = 1, ..., m-1$, represents the sequence of grid cells traversed between g_i and g_{i+1} of the original trajectory.

By associating trajectory points with individual blocks, we imply that the predicted targets move in a step-wise manner, transitioning sequentially from one block to another. This sequence of hexagons representation enables us to establish a structured temporal framework, aligning the trajectories based on a uniform time scale.

4.3 The Data Generation Pipeline

In this section, we discuss the general pipeline for generating higher-order mobility flow data from original trajectory datasets. Fig. 4.3 shows the various steps involved in transforming trajectory data points into sequences of hexagons. Although the procedure is straightforward, it is not trivial and can sometimes be time-consuming. This is primarily due to the involvement of specialized algorithms, such as routing, map-matching, and computationally intensive geometry tasks. We further elaborate on these issues.

The Input. Trajectory datasets are typically becoming available either in the form of GPS trace data, such as the following:

```
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
```

or, in the form of POI check-in data, such as the following:

```
userId, venueId, venueCategoryId, venueCategory, lat, lon, time
589, 4d646c, 4bf5d8, Gym, 35.6099, 139.8256, Apr 03 20:09:41 2012
290, 4b535e, 4bf5d8, Park, 35.7495, 139.5865, Apr 03 20:14:18 2012
```

Map-Matching The original trajectories are represented as a sequence of GPS-based data points. However, GPS data can be noisy and inaccurate, leading to deviations from actual roads and paths. Map-matching aims to correct these inaccuracies and align the raw GPS points with the corresponding road network [79]. While popular methods, such as IVMM [80], exist for map-matching, one can use a routing machine like OSRM [81] to first find the shortest paths between consecutive intermediate data points, and then concatenate (in the same sequence) the output shortest paths to form the map-matched trajectory.

Computational Geometry To transform a map-matched trajectory to a sequence of hexagons, we rely on computational geometry methods. More specifically, every trajectory is

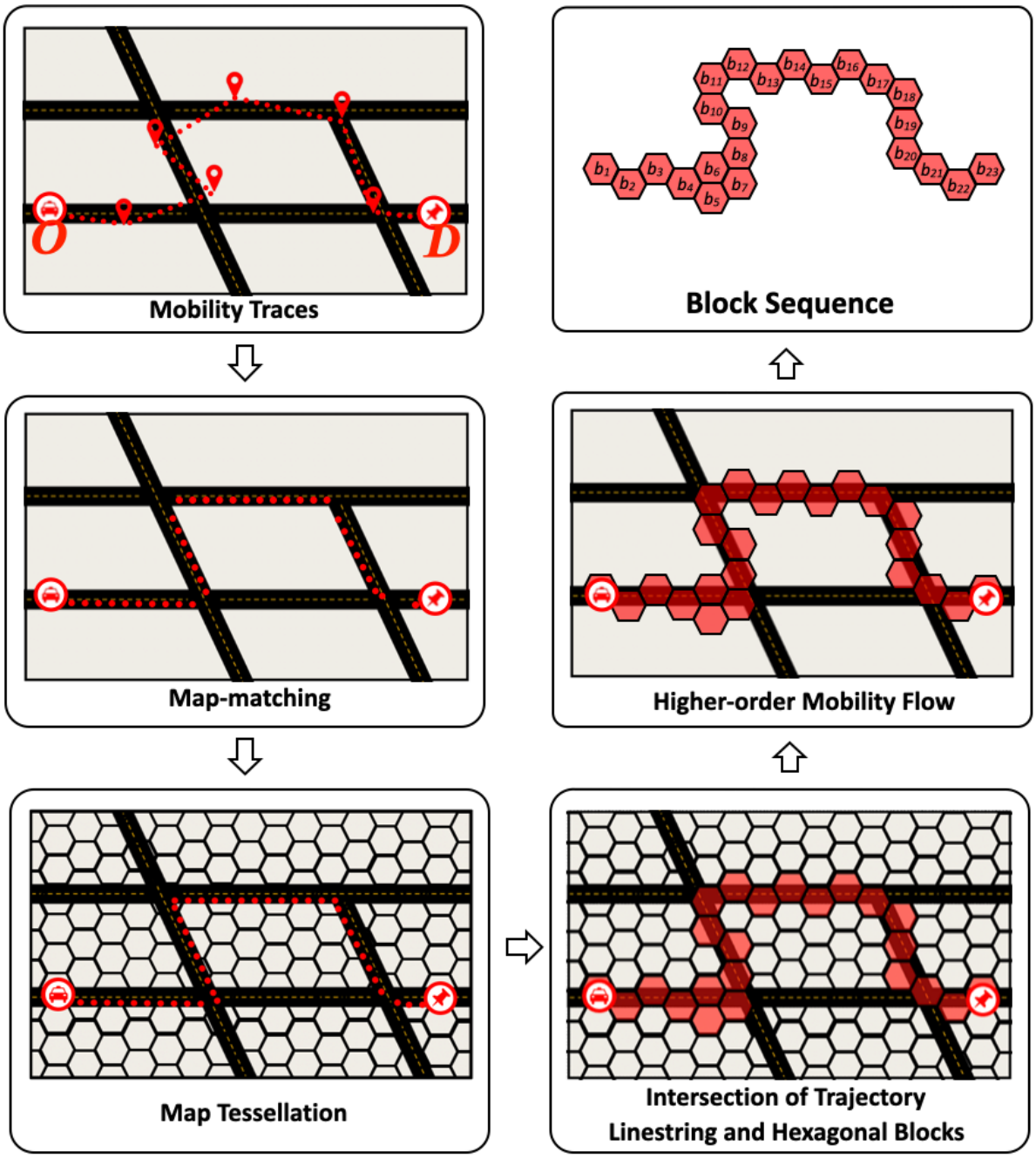


Figure 4.3: Construction of higher-order trajectory data.

modeled as a `linestring` shape type, and every hexagon as a `polygon` shape type. Then, their intersection can be computed using off the shelf methods of popular computational

Resolution	Hex Edge Length (Km)	Hex Area (Km^2)
Hex@6	3.725	36.129
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 4.1: Properties of hexagons of different resolutions[3].

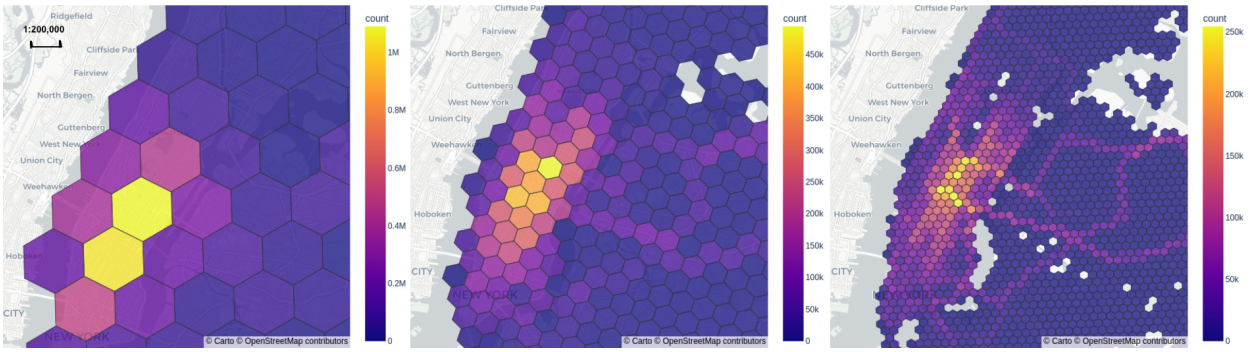


Figure 4.4: An illustrative example of varying resolutions – 7, 8, 9 respectively.

geometry libraries. Recall that a map $\mathcal{M}$ can be tessellated using hexagons of a different size, which defines the map's resolution. We construct datasets of five different resolutions, namely $\{6, ..., 10\}$. Table 4.1 shows the size properties of each hexagon for each resolution. At the end of the data preparation process, each trajectory in the dataset has been transformed to a sequence of hexagonal blocks. Fig. 4.4 provides an illustrative example of higher-order mobility flow data projected on an area of a map that has been tessellated using three different resolutions. The figure shows how higher-order mobility data captures a city's road infrastructure and physical constraints (e.g., bridges). It also shows how different resolutions can be useful in diverse problems and applications, as they allow for analysis at a different level of granularity, ranging from microscopic to macroscopic [82].

The Output. The final output comprises continuous sequences of hexagons which represent

[3] `https://h3geo.org/docs/core-library/restable/`

	Dataset	# Objects	# Trajectories	Time Period	Resolutions	File Size
Gps traces	Ho-T-Drive	9,987	65,117	02/02/08 – 02/08/08	{6, ..., 10}	379 MB
	Ho-Porto	442	1,668,859	07/01/13 – 06/30/14	{6, ..., 10}	370.2 MB
	Ho-Rome	315	5,873	02/01/14 – 03/02/14	{6, ..., 10}	16.6 MB
	Ho-Geolife	57	2,100	04/01/07 – 10/31/11	{6, ..., 10}	3 MB
visits	Ho-Foursquare-Nyc	1,083	49,983	04/12/12 – 02/16/13	{6, ..., 10}	12.3 MB
	Ho-Foursquare-Tky	2,293	117,593	04/12/12 – 02/16/13	{6, ..., 10}	29.3 MB
	Ho-Nyc-Taxi	N/A	2,062,554	01/01/16 – 06/30/16	{6, ..., 10}	341.4 MB

Table 4.2: Statistics of the higher-order mobility flow datasets that we provide (the original datasets are prefixed by 'Ho-').

each trajectory.

4.4 The Datasets

We provide the higher-order mobility flow of the following datasets (see Table 4.2 for a summary of the datasets' statistics):

T-Drive [83, 84]. Taxi trajectory datasets on the Beijing road network that reaches ∼9 million kilometres in total.

Porto [85]. Trajectories of Portuguese taxis that operate through a dispatch central & uses mobile data terminals installed in the taxis.

Rome [86]. Consisting of recorded traces of taxi cabs in Rome, Italy.

Geolife [87, 88, 89]. Collected from several users' Gps-based devices during their outdoor activities (a total distance of ∼1.2 million km)

Nyc-Taxi [90]. Taxi cab trajectory dataset recorded by digital devices installed in the vehicles of New York City.

Foursquare [77]. Poi check-ins recorded by the phones of several users in the city of New York and Tokyo.

4.5 Broader Impact

In this section, we provide insights into how the new datasets (potentially with other datasets) can be used in diverse domains.

Transportation Systems and Urban Planning. Our datasets can be used to create illustrative heatmaps, such as in Fig. 4.4, that depict the density in a particular area or route of the map. These heatmaps can be used to visually analyze traffic patterns, congestion levels and travel times. They can also allow urban planners to design and optimize public transit networks and infrastructure.

Environmental Monitoring and Conservation. Another potential application of higher-order mobility data is in the form of assessing environmental impact (e.g., emissions, air quality, etc.). This helps evaluate the effectiveness of eco-friendly initiatives (e.g., relating to transportation) and supports sustainability planning.

Public Healthcare. Microscopic modeling of spatiotemporal epidemics dynamics research [91, 82] can benefit healthcare practitioners and researchers in surveying and monitoring disease spread through individual mobility behaviors. This can also inform public policy about the effectiveness of targeted actions that aim to mitigate the epidemic spread compared to horizontal measures.

4.6 Problem Definition (Revisited)

Based on the introduction of higher-order trajectory representations, we now revisit the problem of trajectory prediction in the context of predicting k future blocks (hexagons).

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

blocks $b_{i_{l+1}}, \ldots, b_{i_{l+k}}$ of the partial trajectory T.

Chapter 5

Methodology

In this section we provide details of our TrajLearn model. In particular, we present information regarding (i) the Dependency Capturing Module, which is designed to capture complex dependencies of the trajectories, (ii) the training of the model, and (iii) the beam search with constraints that allows to efficiently explore multiple possible future trajectory paths.

5.1 Dependency Capturing Module

The Dependency Capturing Module leverages the power of the Transformer architecture [17] to capture underlying dependencies within the trajectories. Figure 5.1 shows the framework of the training process. In particular, the transformer-based Dependency Capturing Module is the Decoder Stack. This module largely follows the GPT-2 model. Because the use of Transformers and GPT-2 in particular has become common we will omit an exhaustive background description of the model architecture and refer readers to Radford et al. [18] as well as excellent guides such as "The Illustrated GPT-2"[4]. In particular, it is a L-layer

[4]The Illustrated GPT-2. `https://jalammar.github.io/illustrated-gpt2/`

decoder-only Transformer each with A masked self-attention heads and length H dimensional state. In contrast to the original Transformer architecture that used sinusoidal positional encodings, we used learned position embeddings. These learned embeddings can be more flexible as they are capable of learning and adapting to complex patterns within the data. This way, the input to the Transformer is:

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

where $B = (b_1, \ldots, b_l)$ is the higher-order mobility flow, W_e is the block embedding matrix, and W_p is the position embedding matrix.

In each layer (decoder block) of the Transformer, layer normalization is used. In greater detail, layer normalization is a process that standardizes the inputs of a layer across each feature. It helps mitigate unstable training dynamics by ensuring that the distribution of inputs to a layer remains more consistent across training steps, leading to faster convergence of the model. Furthermore, unlike original transformers, layer normalization was moved to the input of each sub-block and an additional layer normalization was added after the final self-attention block. This positioning is strategic; it normalizes the inputs before they enter the self-attention mechanism (preparing them for optimal processing) and then again normalizes the outputs of the self-attention before they proceed to the next layer. Formally, the computation of the hidden state at each transformer layer $j \in [1, L]$ can be described as:

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

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

where LayerNorm$(\cdot)$, Self-Attention$(\cdot)$, and FeedForward$(\cdot)$ denote layer normalization, the masked 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 [92] on all non-bias or gain weights. As an activation function we opted for the Gaussian Error Linear Unit (GELU) [93], which is chosen due to its performance in tasks involving NLP and its ability to alleviate the vanishing gradient problem, allowing for a more effective learning process. GELU is defined as:

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

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

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

In masked attention, every token is constrained to only attend to its left context. The attention mechanism can be formalized as:

$$\begin{gathered} \textit{Self-Attention}(E) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \\ Q = EW_Q, \quad K = EW_K, \quad V = EW_V \end{gathered} \tag{5.6}$$

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`. 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:

$$P(b_{l+1}|B) = \text{softmax}(\text{FeedForward}(\text{LayerNorm}(h_L))) \tag{5.7}$$

Note that although we largely followed GPT-2 architecture for this module, our method for trajectory prediction is generic and any large language model architecture could be utilized as the Dependency Capturing Module. As a result, any further development and recent release on language models is orthogonal and can be equally beneficial to our approach with minimum effort.

5.2 Model Training

In the context of language models, the `<End of Sentence>` or `<EOS>` token serves as a special symbol or marker used to indicate the end of a sentence or sequence of words. In the context of trajectories, we also need an endpoint signal to define the trajectory boundaries and help the model learn when to appropriately end a trajectory. We had to resort to two strategies:

(a) **Temporal cutoff.** We set a time threshold, and a gap in GPS data beyond this threshold indicates the end of the trajectory. This approach assumes that if there is no GPS data recorded for a certain period, the trajectory has ended.

(b) **Spatial cutoff.** We set a distance threshold, and if the distance between consecutive GPS points is greater than this threshold, we consider it as the end of the trajectory. This approach assumes that a significant distance between two consecutive points indicates a break in the trajectory.

These strategies are enforced during the preprocessing phase, where an endpoint is detected if either of the two criteria is met. Specifically, we compute both distance and time span between two consecutive points. If either the distance gap or the timestamp difference is beyond the respective threshold, we consider that the end of a trajectory is detected. Once the endpoint of a trajectory is determined, we represent it with an `<End of Trajectory>` or `<EOT>` special token. This step plays a vital role as it enables the model to simulate real-world

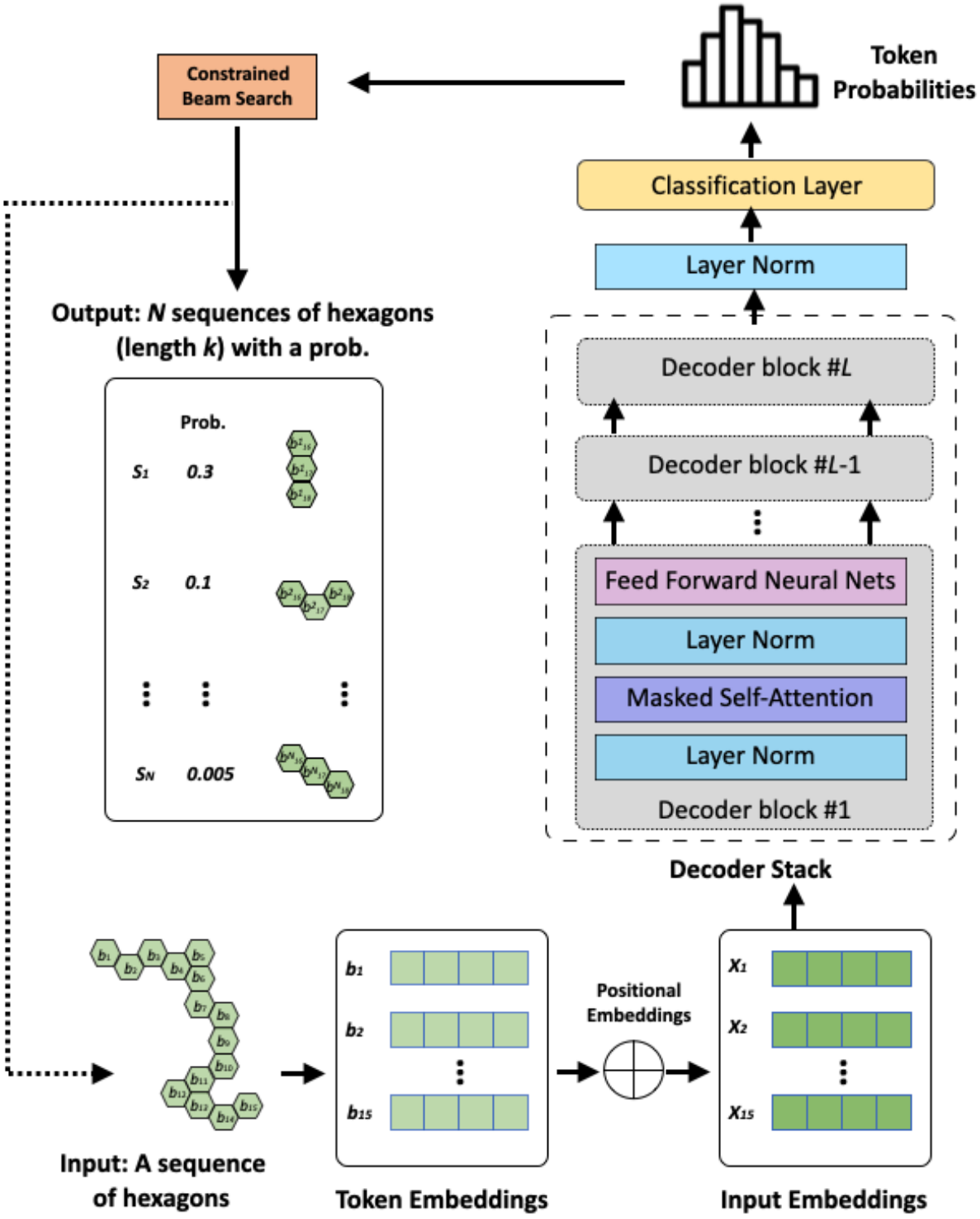


Figure 5.1: Framework architecture.

scenarios, where trajectories naturally conclude, rather than continuing pointlessly. It also helps the model to generate continuous trajectory paths where transitions mostly happen to adjacent hexagons.

Once all trajectories have been represented as sequences of blocks, we transform them into *partial trajectories* that are required for training the prediction model. Specifically, we generate partial trajectories of length $l + k$, as well as all combinations of partial trajectories of length between l and $l + k$. Note that $l \geq 1$ is a model parameter representing the size of the input, and $k \geq 1$ is the prediction horizon. Subsequently, the model is trained by providing as input the first l blocks of partial trajectories and predicting the remaining ones. The process continues until an `<EOT>` token is predicted, or until the trajectory has reached the prediction horizon.

Our training procedure involves the implementation of a method known as *teacher forcing*. This technique is applied to stabilize the training process and accelerate convergence. In text generation, each predicted word is subsequently used as input to predict the next word. Similarly, in our context, each previously predicted block serves as the input for predicting the following step. During the training process, regardless of the model's current prediction of the next block, the correct block (i.e., the ground truth target block) is used for forming the next time step's input. Figure 5.2 is an toy example to demonstrate this process. Without teacher forcing, the mdoel will always use the predicted block as part of the input for the next step prediction; while with teacher forcing, no matter what the model has predicted, the ground truth blocks are used as part of the input. This approach provides a robust supervision signal and effectively allows the model to learn the dependencies and patterns within the trajectory data.

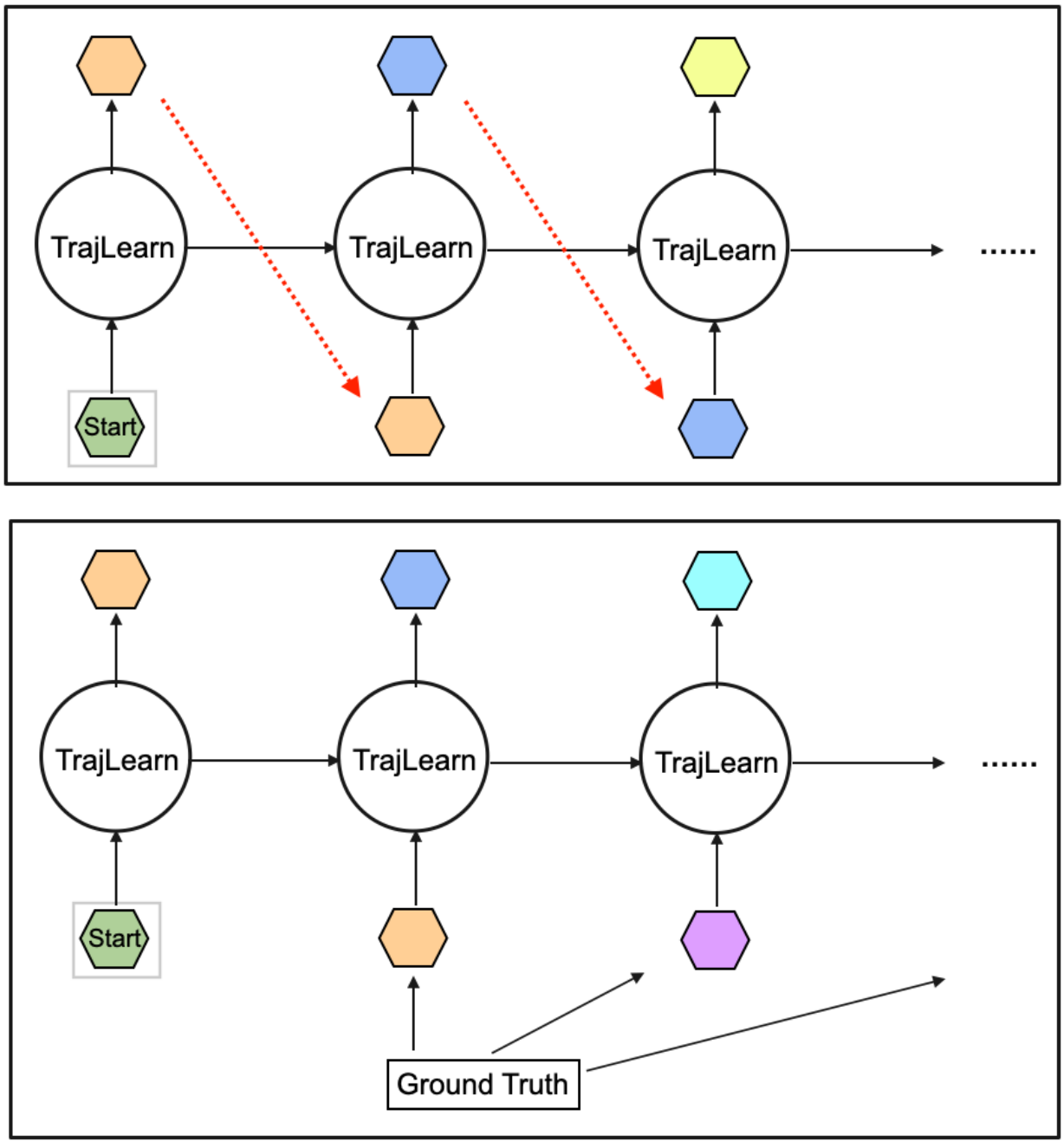


Figure 5.2: TRAJLEARN without teacher forcing (top) and with teacher forcing (bottom).

5.3 Beam Search with Constraints

To optimize the trajectory prediction process and improve efficiency, 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. In Figure 5.1, after calculating the probabilities for all hexagons for the next step prediction, we retain the top n most plausible blocks, each with its associated probability, instead of just the single most probable block. In the subsequent prediction round, probabilities will be computed based on the condition of these top n blocks from the previous step. By guiding the search process and focusing on the most likely trajectory paths, beam search improves the efficiency of prediction and increases the quality of the predicted trajectories. Figure 1.1 shows an example where two paths are explored. 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. 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 retain only the top k candi-

dates based on their cumulative probabilities, which inform the next beam expansion.

- **Termination.** The algorithm continues with expansion and pruning until it reaches the desired prediction horizon or until all beam candidates reach the `<EOT>` symbol.

Additionally, we introduce constraints to the beam search algorithm to preserve spatial continuity in the predicted trajectories. Formally, for a current block $b_i \in B$, 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 blocks that are adjacent to the current block during the next block prediction task we ensure *the validity* of the predicted trajectories, as they are guided to always 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.

Chapter 6

Experimental Evaluation

In this section, we present a comprehensive experimental evaluation of our model. First, we list the research questions we aim to explore. Then, we present details of the experimental configuration, datasets employed, the baseline methods, and evaluation metrics. Finally, we present the results and discuss insights and broader impact.

6.1 Experimental Scenarios

Our experiments aim to answer the following questions:

(Q1) Model Accuracy Performance. What is the accuracy performance of our method against sensible baseline methods?

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

(Q3) Map Resolution Analysis. How does the model's performance vary with different tessellation levels of the higher-order mobility flow representation?

(Q4) Ablation Study. How does beam search with the constraints impact the model's performance overall?

6.2 Experimental Configuration

Computational Environment. We run experiments on a server equipped with an NVIDIA RTX A6000 graphics card and 320GB of memory. The model was developed in Python 3, and was built and trained by employing the PyTorch 1.13 deep learning framework.

Map Tessellation and Resolutions. For tessellating a map, we utilized the `H3` geo-indexing system[5], which efficiently partitions the world into hexagonal cells of varying resolutions. Table 4.1 presents the size properties of each regular hexagon for the different resolutions considered; we specifically report hexagon's *edge length* (in km) and *surface area* (in 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.

6.3 Datasets

In the experiments, we employ higher-order mobility flow data representations of three popular real-world trajectory datasets. We briefly describe the semantics of each dataset (prefixed by "HO-" to indicated higher-order) and their statistics are shown in Table 4.2 Chapter 4.

HO-PORTO [85] consists of recorded mobility traces of taxis operating in Porto, Portugal.

HO-ROME [86] consists of recorded mobility traces of taxis operating in Rome, Italy.

[5] `https://h3geo.org/`

Ho-Geolife [87, 88, 89] consists of recorded mobility traces of individuals, covering a total distance of ∼1.2 million Km.

After transforming each dataset to higher-order mobility flow, we split the timely ordered trajectory data set into *training*, *validation*, and *test* sets with a 70%, 10%, and 20% ratio, respectively.

6.4 Baseline Methods

Although datasets presented in this thesis are commonly used throughout trajectory prediction literature, subsequent modifications in various research papers have made it challenging to compare results across different studies. One prominent issue stems from the inclusion of additional metadata in different papers, which was not originally present in the original dataset, such as incorporating point-of-interests (POI) [94] or adding weather data [95], thereby providing certain models with an advantage over others, irrespective of their architecture or training design. Furthermore, variations in the preprocessing pipelines employed by different research teams have resulted in significant disparities in the data generated. This may include the exclusion of trajectories with GPS measurement errors [96, 95] or outliers [97, 95] that form the most challenging trajectories to predict. In order to assess the performance of our model, we have selected five 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 [98]: 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 [99]: Long Short-Term Memory (LSTM) is a type of a recurrent neural network (RNN) designed to capture long-term sequential dependencies, a crucial aspect in mobility

prediction tasks.

LSTM-ATTN [100]: LSTM with attention is a variant of LSTM, which integrates an attention mechanism that allows the model to focus on specific parts of the sequence when making predictions.

GRU [101]: Gated Recurrent Units (GRU) is a variant of RNN, which was designed to address the vanishing gradient problem of RNNs. Similar to LSTM, it can be applied to sequence prediction tasks.

DeepMove [13]: DeepMove is a state-of-the-art method that combines a multi-modal recurrent network along with a historical attention mechanism to capture both spatial and temporal dependencies.

To ensure a fair comparison, we adopt the optimal parameter configurations as outlined in the original papers for all baseline methods.

6.5 Evaluation Metrics

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

Accuracy@n [↑]. This metric measures the proportion of true samples that are included in the predictions. Formally:

$$\text{Accuracy@n} = \frac{|\{s|s \in P, \text{true}(s) \in \text{Top}_n(s)\}|}{|P|} \tag{6.1}$$

where N is the number of N most probable sequences from the beam search, and each sequence is associated with a probability. $|P|$ denotes the total number of trajectories in the testing dataset. In other words, it is the length of the testing dataset. $true(s)$ is the true label for a prediction sequence s, and $\text{Top}_n(s)$ includes the n most probable predictions

for s, which which is analogous to the concept of ACCURACY@N. Note that for each trajectory in the testing dataset, we consider the prediction correct and assign a score of 1 to that trajectory only if the ground truth sequence appears exactly in the $\text{Top}_n(s)$. If not, the trajectory receives a score of 0. To calculate the overall accuracy, we tally the 1s and 0s for all trajectories in the testing dataset and then average these scores over the total number of trajectories, denoted as |P| which is defined earlier. In this study, we calculate accuracy at top 1, 3, and 5 (i.e., ACCURACY@1, ACCURACY@3, and ACCURACY@5).

BLEU SCORE [↑]. BLEU, or the Bilingual Evaluation Understudy, is a standard metric for sequence prediction tasks which measures the quality of predicted sequences by comparing them with the ground truth sequences. The BLEU score measures how many n-grams of the predicted sequence match with the n-grams in the actual sequence, adjusted by *a brevity penalty* for short predicted sequences. For precision p_n and brevity penalty BP, the BLEU score is defined as follows:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{6.2}$$

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

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{6.3}$$

where c is the length of the predicted sequence and r is the effective length of the ground truth sequence. In our experiments, following the common choice in the NLP literature, we consider n-grams up to 4-gram and uniform weight values.

6.6 Results and Discussion

(Q1) Model Accuracy Performance. We compare the performance of TrajLearn against the baseline methods 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 6.1 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 positions, and frequently (in 32 out of 36 instances) outperforming all competitors, (ii) TrajLearn's reported accuracy performance is getting better as the N of the Accuracy@N is increasing. This is due to the *teacher forcing* technique involved in the training stage (see 5.2), which provides supervision and corrects the prediction for any subsequent step, effectively allowing TrajLearn to learn.

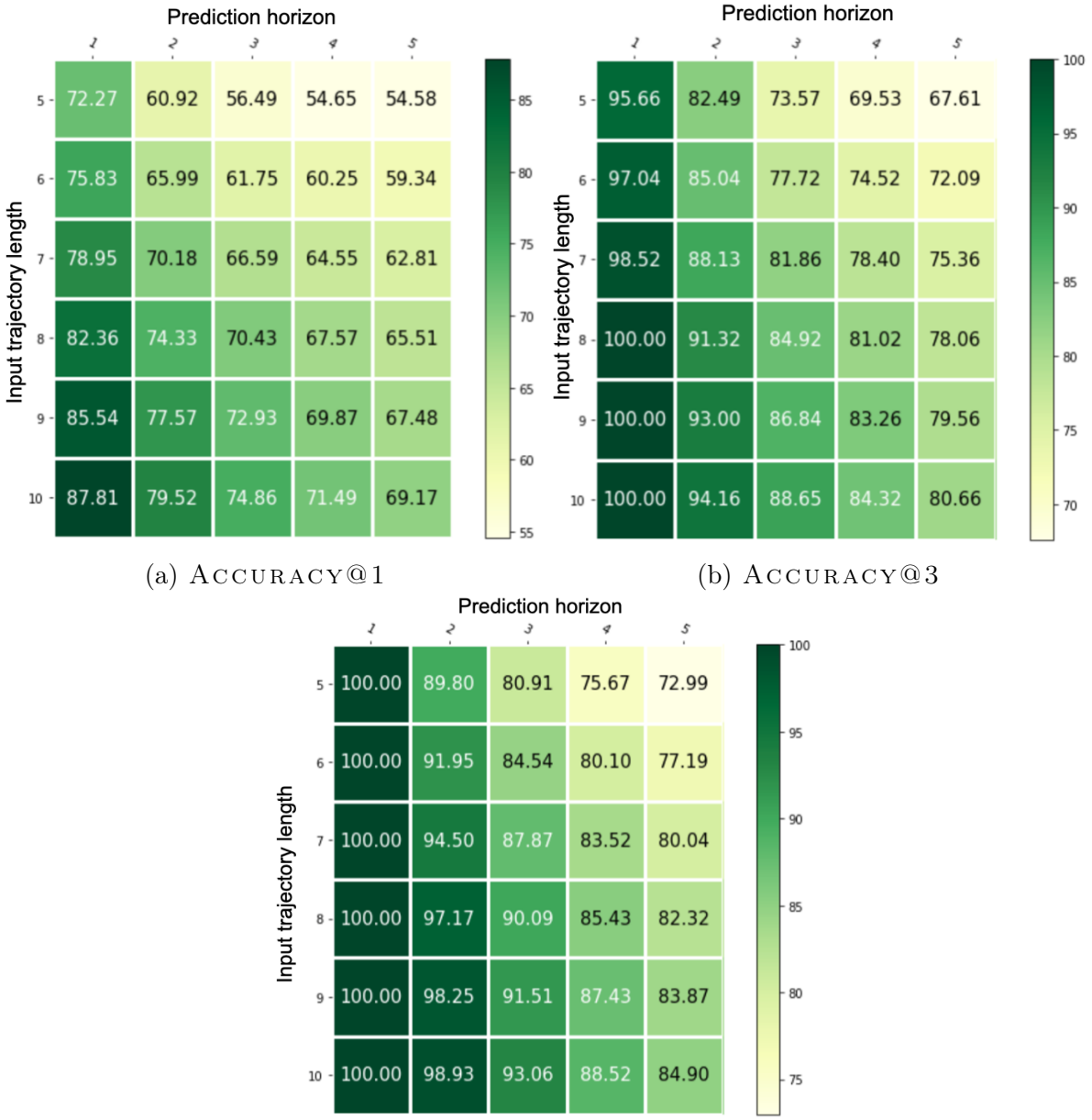
Note that in specific scenarios, we encountered instances of Out of Memory (OOM) errors due to GPU memory being exhausted during the execution of our experiments. This is primarily attributed to the approach employed by current state-of-the-art methods in training their trajectory prediction models, where training data heavily relies on sparsely available Points of Interest (POI) check-in information. Upon aligning their methodology with our continuous path approach (hexagonal traversal), the volume of "check-ins" experiences a substantial surge, consequently leading to the occurrence of OOM errors. Although these OOM errors hindered the completion of certain experiments, their impact on the overall conclusions remains minimal, as the remaining results clearly depict the generalizability and practicality of our approach.

(Q2) Parameter Sensitivity Analysis. In this experiment, we investigate the influence of varying input trajectory length ($5 \leq l \leq 10$) and prediction length ($1 \leq k \leq 5$) on the accuracy performance of TrajLearn and different evaluation metrics (Accuracy@1,

ACCURACY@3, and ACCURACY@5). Figure 6.1 presents the results, where the vertical axis represents the input trajectory length (l) and the horizontal axis represents the prediction horizon (k). A few key observations can be made: (i) the longest the history provided as input l, the higher the prediction accuracy, (ii) the shortest 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 results are getting better as the N of the ACCURACY@N is increasing.

(Q3) Map Resolution Analysis. In this experiment, we investigate the impact of a map's resolution to TRAJLEARN's accuracy 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 6.2 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, as the prediction horizon increases, (ii) the smaller the resolution, the slower the rate with which the accuracy decreases, as the distance travelled increases.

(Q4) Ablation Study. In this experiment, we study the impact of certain components of our trajectory prediction model. We selectively remove the beam search module and constraint mechanism and report the *accuracy performance change*. The results are provided in Table 6.2. In addition, we assessed the performance of TRAJLEARN through a hyperparameter analysis to gauge the impact of various parameters: *embedding dimension*, *number of encoder layers*, and *number of attention heads*. These tests were carried out using the GEOLIFE@7 dataset. The findings are illustrated in Figure 6.3. From our observations, TRAJLEARN's performance generally benefits from an increase in the number of layers, as it allows for a more comprehensive processing of dependencies. Likewise, enhancing the number of attention



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

(c) Accuracy@5

Figure 6.1: TrajLearn accuracy performance for varying input trajectory length l (vertical) & prediction horizon k (horizontal).

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

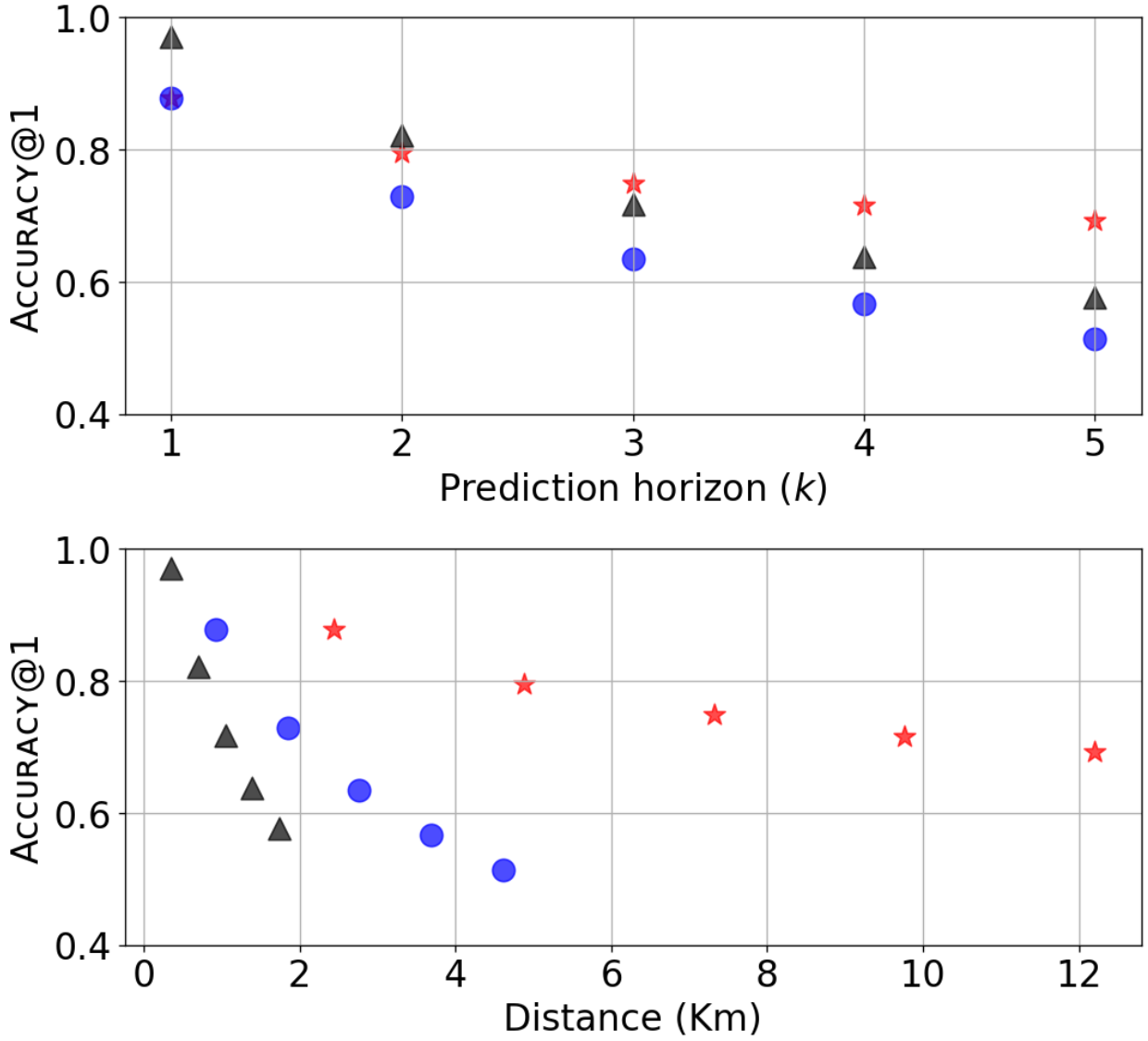


Figure 6.2: TrajLearn accuracy performance for varying resolutions (7: star, 8: circle, 9: triangle) on the Ho-Porto dataset. We report accuracy over prediction horizon k (top), and over the distance traveled since the last trajectory point (bottom).

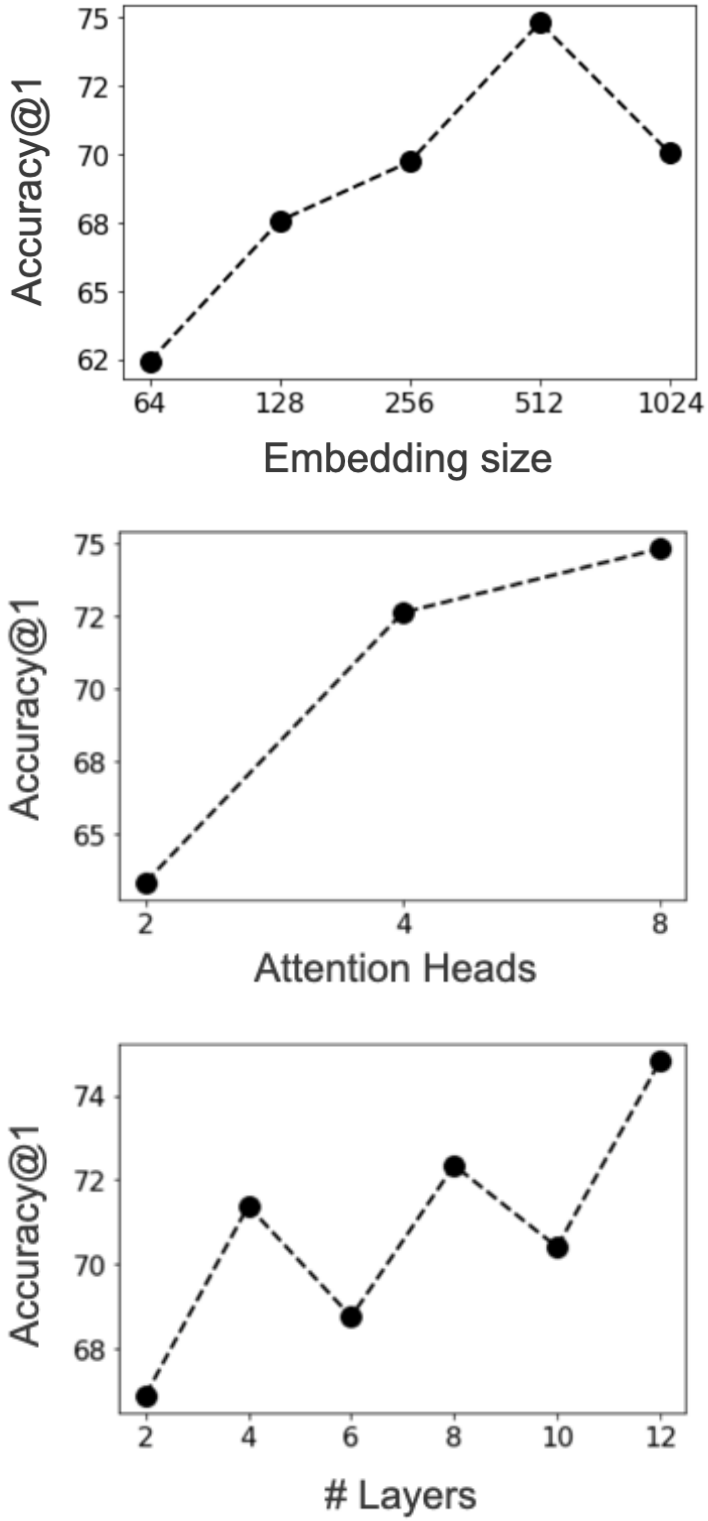


Figure 6.3: ACCURACY@1 results over Ho-Geolife@7 for varying embedding vector size (top), number of attention heads (middle), and number of Transformer layers (bottom).

Dataset	Model	Resolution 7 Acc@1	Acc@3	Acc@5	BLEU	Resolution 8 Acc@1	Acc@3	Acc@5	BLEU	Resolution 9 Acc@1	Acc@3	Acc@5	BLEU
Ho-Porto	**MC**	0.3284	0.4586	0.4908	0.2444	0.2478	0.3354	0.3893	0.2359	OOM	OOM	OOM	OOM
	LSTM	0.5970	0.6318	0.6400	0.6302	0.4579	0.5087	0.5172	0.5021	0.5044	0.5588	0.5643	0.5479
	LSTM-ATTN	0.1113	0.1923	0.2065	0.2035	0.1112	0.1705	0.1929	0.2065	0.2716	0.3682	0.4011	0.3842
	GRU	0.5532	0.5877	0.5957	0.5866	0.3154	0.3542	0.3606	0.3530	0.3649	0.4086	0.4144	0.4058
	DeepMove	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM	OOM
	TrajLearn (ours)	**0.6917**	**0.8066**	**0.8490**	**0.7691**	**0.5135**	**0.6931**	**0.7590**	**0.5918**	**0.5772**	**0.8022**	**0.8741**	**0.6379**
	Improvement (%)	15.86 %	27.65 %	32.64 %	22.04 %	12.14 %	36.25 %	46.75 %	17.86 %	14.43 %	43.56 %	54.90 %	16.43 %
Ho-Rome	**MC**	0.2088	0.3982	0.4690	0.0685	0.2374	0.3811	0.4590	0.1504	0.2100	0.3157	0.3564	0.1686
	LSTM	0.2820	0.3138	0.3227	0.3179	0.3932	0.4340	0.4407	0.4315	0.4617	0.5144	0.5186	0.5036
	LSTM-ATTN	0.1079	0.1522	0.1850	0.1819	0.2264	0.2845	0.3055	0.2998	0.2890	0.3704	0.3892	0.3735
	GRU	0.2966	0.3298	0.3385	0.3335	0.3997	0.4400	0.4468	0.4379	0.4638	0.5158	0.5199	0.5052
	DeepMove	0.3406	**0.4969**	**0.5793**	0.2821	0.3860	0.5036	0.5657	0.3286	OOM	OOM	OOM	OOM
	TrajLearn (ours)	**0.3746**	0.4740	0.5167	**0.4215**	**0.4974**	**0.6428**	**0.6996**	**0.5434**	**0.5671**	**0.7657**	**0.8431**	**0.6138**
	Improvement (%)	9.98 %	-4.61 %	-10.81 %	26.38 %	24.44 %	27.64 %	23.67 %	24.09 %	22.27 %	48.45 %	62.17 %	21.49 %
Ho-GeoLife	**MC**	0.2153	0.4917	0.6050	0.1113	0.2149	0.3951	0.4897	0.0866	0.2063	0.3314	0.3859	0.0848
	LSTM	0.5900	0.6086	0.6114	0.6117	0.5616	0.5836	0.5864	0.5838	0.5725	0.6057	0.6085	0.6039
	LSTM-ATTN	0.4944	0.5559	0.5621	0.5478	0.3496	0.4148	0.4249	0.4101	0.2905	0.3664	0.3959	0.3872
	GRU	0.6229	0.6435	0.6465	0.6439	0.5514	0.5742	0.5779	0.5732	**0.5799**	0.6132	0.6158	**0.6111**
	DeepMove	0.5295	0.6742	0.7370	0.3653	0.4529	0.5699	0.6374	0.3374	OOM	OOM	OOM	OOM
	TrajLearn (ours)	**0.7481**	**0.8247**	**0.8635**	**0.7785**	**0.6249**	**0.7404**	**0.7823**	**0.6558**	0.5664	**0.6781**	**0.7194**	0.6004
	Improvement (%)	20.10 %	22.32 %	17.16 %	20.90 %	11.27 %	26.87 %	22.73 %	12.33 %	-2.32 %	10.58 %	16.82 %	-1.77 %

Table 6.1: TrajLearn accuracy performance against five baselines, 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.

Dataset	Accuracy@1	Change
Ho-Porto@7	0.6844	-1.07%
Ho-Porto@8	0.4992	-2.86%
Ho-Porto@9	0.5672	-1.76%

Table 6.2: Impact of removing the beam search on accuracy.

Chapter 7

Conclusion

7.1 Summary and Contributions

We focused on the problem of trajectory prediction and proposed TrajLearn, a trajectory deep generative model that has shown remarkable results in predicting the future path of a trajectory. Our model's Transformer-based architecture is inspired by the GPT-2 model. We trained our model from scratch on large-scale higher-order mobility flow data sets that effectively represent trajectories as sequences of hexagons on a hex-tessellated map. TrajLearn incorporates a training data pipeline that converts GPS coordinates into higher-order mobility flow trajectories, leading to a notable data simplification. Representing trajectories as sequences of blocks enabled us to preserve crucial spatial and temporal information while reducing data complexity. As a result, the prediction process became more manageable and efficient, thanks to the advantages of reduced complexity. In addition, TrajLearn incorporates a sophisticated module for capturing dependencies within sequences of blocks representing trajectories. Together with the devised beam search with constraints algorithm, this module demonstrated effectiveness in handling trajectory-specific sequential data. The results of extensive experiments over three real-world data sets demonstrated the effectiveness

of the proposed model. TrajLearn consistently outperformed several strong baselines, for varying settings and parameters. We believe our proposed approach and model for trajectory prediction can have broad and useful applications.

7.2 Applications

Trajectory prediction has a wide range of applications across various fields, significantly impacting society, technology, and industry. Here's an overview of some key applications:

- **Navigation and Route Optimization**: Trajectory prediction is crucial for real-time navigation systems in vehicles and mobile applications. It helps in predicting traffic conditions and suggesting optimal routes, thereby reducing travel time and improving fuel efficiency.
- **Geospatial Analysis in Urban Planning**: Urban planners use GPS trajectory data and trajectory prediction to understand traffic patterns, pedestrian movements, and public transportation usage, which aids in urban design and infrastructure development.
- **Location-Based Services and Recommendations**: For businesses offering location-based services, trajectory prediction can provide personalized recommendations to users, like suggesting nearby restaurants, events, or retail offers based on their movement patterns.
- **Research in Human Mobility and Social Sciences**: Researchers analyze GPS trajectory data to study human mobility patterns, social behavior, and the interaction between humans and their environment, contributing to various fields like sociology, geography, and epidemiology.

These applications demonstrate the versatility and significance of trajectory prediction in various domains, contributing to advancements in technology, safety, and efficiency.

7.3 Broader Impact

Addressing the trajectory prediction problem has a broad and multifaceted impact across various sectors and aspects of society. Here are some of the key impacts:

- **Enhanced Safety**: In fields like autonomous driving, aviation, and maritime navigation, accurate trajectory prediction significantly improves safety by reducing the risk of collisions and accidents. This has a direct impact on saving lives and preventing injuries.
- **Improved Efficiency and Resource Management**: In logistics, supply chain management, and urban planning, better trajectory prediction leads to optimized routes and schedules, reducing fuel consumption, lowering emissions, and enhancing overall efficiency.
- **Enhanced Public Services**: In public transportation and urban planning, better trajectory prediction can lead to more efficient public transport systems, less traffic congestion, and improved urban living conditions.

In summary, addressing the trajectory prediction problem not only provides immediate practical benefits in specific industries but also contributes to broader societal goals such as safety, sustainability, economic growth, and technological advancement.

7.4 Future Work

Several paths are available for further investigation in this field, which we will briefly outline in the following discussion.

7.4.1 Extension of Current Model

For future work of this research, two potential directions can be pursued with regard to the current model. First, exploring other Large Language Models (LLMs) such as BERT, PaLM and LLaMA could offer diverse architectural strengths, possibly enhancing the model's ability to understand and predict complex trajectory patterns. Different LLMs bring unique capabilities in processing sequential data, which might yield improved accuracy in trajectory forecasting. Secondly, instead of leveraging deep generative models, other methods such as flow normalization can be explored for hexagonal trajectory prediction task. This approach, known for its efficiency in modeling complex distributions, could potentially capture intricate patterns in hexagonal sequences of moving objects, leading to precise and reliable trajectory predictions. Both these directions not only promise to increase the model's performance but also pave the way for groundbreaking advancements in trajectory prediction.

7.4.2 Mixed Resolution on Map Tessellation

In this thesis, we tessellate the map into uniform hexagons. In other word, each hexagon in these tessellated maps is of the same size. For future work, one potential direction is to vary the hexagon sizes in tessellation based on trajectory density. In areas with denser trajectories, smaller hexagons (higher resolution) could be used to reveal more detail. Conversely, in less frequented areas, larger hexagons (smaller resolution) could be employed. This approach could accelerate the training process by reducing the number of hexagons to process, and it also aligns more closely with real-world scenarios, making it a more practical solution.

7.4.3 Interaction Prediction in Mobility Networks

Given trajectories where multiple objects are continuously moving in the same space, they can be found in close proximity to each other and interact. All pair-wise interactions taking

place at a certain time t form a proximity network. If we define a time period $[0, T]$, all proximity networks formed for each $t \in [0, T]$ form a mobility network. A mobility network represents patterns of interactions between objects. Mobility network analysis is of great importance in several downstream tasks including predicting the sequence of interactions between individuals, and predicting node importance in spatiotemporal networks. It is also the key to real-world applications such as understanding the spread of infectious diseases, and crowd behavior analysis, to name a few. However, analyzing mobility networks is a challenging problem due to the dynamicity of the networks, where interactions are formed and dissolved over time due to the changes of the physical proximity of the moving objects. One potential extension of the current work is to predict future interactions in a mobility network based on the accurate prediction of the mobility patterns of individual objects. Specifically, the result from trajectory prediction can be used to predict future pair-wise interactions of objects, and therefore future instances of the mobility network. This approach allows to leverage LLMs to address a challenging spatiotemporal graph problem.

Bibliography

[1] Sheng Wang et al. “A Survey on Trajectory Data Mgt., Analytics, & Learning”. In: *ACM Comp. Surv.* 54.2 (2021). ISSN: 0360-0300.

[2] Mahmoud Sakr, Cyril Ray, and Chiara Renso. “Big mobility data analytics: recent adv. & open problems”. en. In: *GeoInformatica* 26.4 (2022), pp. 541–549. ISSN: 1384-6175, 1573-7624.

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

[4] Gowtham Atluri, Anuj Karpatne, and Vipin Kumar. “Spatio-Temporal Data Mining: A Survey of Problems and Methods”. In: *ACM Comp. Surveys* 51.4 (2018). ISSN: 0360-0300.

[5] Ali Hamdi et al. “Spatiotemporal data mining: a survey on challenges and open problems”. en. In: *AIR* 55.2 (2022), pp. 1441–1488. ISSN: 0269-2821, 1573-7462.

[6] Chen Cheng et al. “Fused matrix factorization with geographical and social influence in location-based social networks”. In: *Proc. of the AAAI*. Vol. 26. 1. 2012, pp. 17–23.

[7] Xutao Li et al. “Rank-geofm: A ranking based geographical factorization method for point of interest recommendation”. In: *SIGIR*. 2015, pp. 433–442.

[8] Defu Lian et al. “GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation”. In: *SIGKDD*. 2014, pp. 831–840.

[9] Chen Cheng et al. "Where you like to go next: Successive point-of-interest recommendation". In: *IJCAI*. 2013.

[10] 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.

[11] Hongzhi Shi et al. "Semantics-aware hidden Markov model for human mobility". In: *IEEE TKDE* 33.3 (2019), pp. 1183–1194.

[12] Qiang Liu et al. "Predicting the next location: A recurrent model with spatial and temporal contexts". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[13] Jie Feng et al. "Deepmove: Predicting human mobility with attentional recurrent networks". In: *Proceedings of the 2018 world wide web conference*. 2018, pp. 1459–1468.

[14] Renhe Jiang et al. "Deepurbanmomentum: An online deep-learning system for short-term urban mobility prediction". In: *AAAI*. Vol. 32. 1. 2018.

[15] Amin Sadri et al. "What will you do for the rest of the day? An approach to continuous trajectory prediction". In: *IMWUT* 2.4 (2018), pp. 1–26.

[16] Licia Amichi et al. "From movement purpose to perceptive spatial mobility prediction". In: *Proc. of the 29th Int. Conf. on Advances in Geographic Information Systems*. 2021, pp. 500–511.

[17] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[18] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[19] Ali Faraji et al. “Point2Hex: Higher-order Mobility Flow Data and Resources”. In: *Proc. of the 31st Intl. Conf. on Adv. in Geographic Info. Sys.(SIGSPATIAL'23)*. 2023.

[20] H Cao, H Mamoulis, and DW Cheung. “Mining frequent spatiotemporal sequential patterns Fifth IEEE International Conference on Data Mining (ICDM'05)”. In: *IEEE, pp 8pp* (2005).

[21] Huiping Cao, Nikos Mamoulis, and David W Cheung. “Discovery of periodic patterns in spatiotemporal sequences”. In: *IEEE Transactions on Knowledge and Data Engineering* 19.4 (2007), pp. 453–467.

[22] Monica Wachowicz et al. “Finding moving flock patterns among pedestrians through collective coherence”. In: *International Journal of Geographical Information Science* 25.11 (2011), pp. 1849–1864.

[23] Weiming Hu et al. “An incremental DPMM-based method for trajectory clustering, modeling, and retrieval”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.5 (2013), pp. 1051–1065.

[24] 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.

[25] 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.

[26] Zhixian Yan et al. “Semantic trajectories: Mobility data computation and annotation”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 4.3 (2013), pp. 1–38.

[27] Xiaolei Li et al. "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.

[28] 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.

[29] Tilemachos Pechlivanoglou and Manos Papagelis. "Fast and accurate mining of node importance in trajectory networks". In: *2018 IEEE International Conference on Big Data (Big Data)*. IEEE. 2018, pp. 781–790.

[30] Abdullah Sawas et al. "A versatile computational framework for group pattern mining of pedestrian trajectories". In: *GeoInformatica* 23 (2019), pp. 501–531.

[31] Yu Zheng et al. "Understanding transportation modes based on GPS data for web applications". In: *ACM Transactions on the Web (TWEB)* 4.1 (2010), pp. 1–36.

[32] Adel Bolbol et al. "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.

[33] 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.

[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] Di Chai, Leye Wang, and Qiang Yang. "Bike flow prediction with multi-graph convolutional networks". In: *Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems*. 2018, pp. 397–400.

[36] Yaguang Li et al. "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting". In: *arXiv preprint arXiv:1707.01926* (2017).

[37] Yuxuan Zhang et al. "Trafficgan: Network-scale deep traffic prediction with generative adversarial nets". In: *IEEE Transactions on Intelligent Transportation Systems* 22.1 (2019), pp. 219–230.

[38] Yaguang Li et al. "Multi-task representation learning for travel time estimation". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 1695–1704.

[39] Kun Fu et al. "Compacteta: A fast inference system for travel time prediction". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3337–3345.

[40] Henry Martin et al. "Graph convolutional neural networks for human activity purpose imputation". In: *NIPS 2018 Spatiotemporal Workshop*. OpenReview. 2018.

[41] Yuanshao Zhu et al. "Semi-supervised federated learning for travel mode identification from GPS trajectories". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2021), pp. 2380–2391.

[42] Gian Alix and Manos Papagelis. "PathletRL: Trajectory Pathlet Dictionary Construction using Reinforcement Learning". In: *Proc. of the 31st ACM SIGSPATIAL. Accepted* (2023).

[43] Quanjun Chen et al. “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.

[44] Leyan Deng et al. “Graph convolutional adversarial networks for spatiotemporal anomaly detection”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.6 (2022), pp. 2416–2428.

[45] Yingfan Huang et al. “Stgat: Modeling spatial-temporal interactions for human trajectory prediction”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6272–6281.

[46] Peishan Cong et al. “Stcrowd: A multimodal dataset for pedestrian perception in crowded scenes”. In: *CVPR*. 2022, pp. 19608–19617.

[47] Lihuan Li, Maurice Pagnucco, and Yang Song. “Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction”. In: *CVPR*. 2022, pp. 2231–2241.

[48] Xuanchi Ren et al. “Safety-aware motion prediction with unseen vehicles for autonomous driving”. In: *ICCV*. 2021, pp. 15731–15740.

[49] Holger Caesar et al. “nuscenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.

[50] Haoran Song et al. “Pip: Planning-informed trajectory prediction for autonomous driving”. In: *ECCV*. 2020, pp. 598–614.

[51] Robert Krajewski et al. “The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems”.

In: *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE. 2018, pp. 2118–2125.

[52] Jinghai Duan et al. "Complementary attention gated network for pedestrian trajectory prediction". In: *AAAI*. Vol. 36. 1. 2022, pp. 542–550.

[53] Jianhua Sun et al. "Human trajectory prediction with momentary observation". In: *CVPR*. 2022, pp. 6467–6476.

[54] Stefano Pellegrini et al. "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, pp. 261–268.

[55] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. "Crowds by example". In: *Computer graphics forum*. Vol. 26. 3. Wiley Online Library. 2007, pp. 655–664.

[56] Bruno Ferreira de Brito et al. "Social-vrnn: One-shot multi-modal trajectory prediction for interacting pedestrians". In: *Conference on Robot Learning*. PMLR. 2021, pp. 862–872.

[57] Ziqian Lin et al. "Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis". In: *AAAI*. Vol. 33. 01. 2019, pp. 1020–1027.

[58] Yisheng Lv et al. "Traffic flow prediction with big data: A deep learning approach". In: *IEEE Transactions on Intelligent Transportation Systems* 16.2 (2014), pp. 865–873.

[59] Huaxiu Yao et al. "Deep multi-view spatial-temporal network for taxi demand prediction". In: *AAAI*. Vol. 32. 1. 2018.

[60] Zipei Fan et al. "Online deep ensemble learning for predicting citywide human mobility". In: *IMWUT* 2.3 (2018), pp. 1–21.

[61] Xuan Song, Hiroshi Kanasugi, and Ryosuke Shibasaki. "Deeptransport: Prediction and simulation of human mobility and transportation mode at a citywide level". In: *IJCAI*. 2016, pp. 2618–2624.

[62] Jiahao Ji et al. "Spatio-temporal self-supervised learning for traffic flow prediction". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 4. 2023, pp. 4356–4364.

[63] Xian Zhou et al. "Predicting multi-step citywide passenger demands using attention-based neural networks". In: *Proceedings of the Eleventh ACM international conference on web search and data mining*. 2018, pp. 736–744.

[64] Defu Lian et al. "Geography-aware sequential location recommendation". In: *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2020, pp. 2009–2019.

[65] Dingqi Yang et al. "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.

[66] 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.

[67] S. Bond-Taylor et al. "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models". In: *IEEE TPAMI* 44.11 (2022), pp. 7327–7347. ISSN: 1939-3539.

[68] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

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

[70] Karol Gregor et al. "Deep autoregressive networks". In: *ICML*. 2014, pp. 1242–1250.

[71] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. "Normalizing flows: An introduction and review of current methods". In: *IEEE TPAMI* 43.11 (2020), pp. 3964–3979.

[72] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

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

[74] Aakanksha Chowdhery et al. "Palm: Scaling language modeling with pathways". In: *arXiv preprint arXiv:2204.02311* (2022).

[75] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[76] Zipei Fan et al. "Online trajectory prediction for metropolitan scale mobility digital twin". In: *Proc. of the 30th ACM SIGSPATIAL*. 2022, pp. 1–12.

[77] Dingqi Yang et al. "Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.1 (2015), pp. 129–142. ISSN: 2168-2216.

[78] 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.

[79] Paul Newson and John Krumm. "Hidden Markov Map Matching through Noise and Sparseness". In: *SIGSPATIAL*. 2009, pp. 336–343.

[80] Jing Yuan et al. “An Interactive-Voting Based Map Matching Algorithm”. In: *IEEE MDM*. 2010, pp. 43–52.

[81] Dennis Luxen and Christian Vetter. “Real-time routing with OpenStreetMap data”. In: *SIGSPATIAL*. 2011, pp. 513–516.

[82] Tilemachos Pechlivanoglou et al. “Microscopic modeling of spatiotemporal epidemic dynamics”. In: *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Spatial Computing for Epidemiology*. 2022, pp. 11–21.

[83] Jing Yuan et al. “T-Drive: Driving Directions Based on Taxi Trajectories”. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS ’10. San Jose, California: Association for Computing Machinery, 2010, pp. 99–108.

[84] Jing Yuan et al. “Driving with Knowledge from the Physical World”. In: *Proc. of the 17th ACM SIGKDD*. 2011, pp. 316–324.

[85] Luis Moreira-Matias et al. *ECML/PKDD 15: Taxi Trajectory Prediction Porto Dataset*. Kaggle, 2015.

[86] Lorenzo Bracciale et al. *CRAWDAD roma/taxi*. 2022.

[87] Yu Zheng et al. “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.

[88] Yu Zheng et al. “Mining Interesting Locations and Travel Sequences from GPS Trajectories”. In: *TheWebConf*. 2009, pp. 791–800.

[89] Yu Zheng, Xing Xie, and Wei-Ying Ma. “GeoLife: A Collaborative Social Networking Service among User, location and trajectory”. In: *IEEE Dat. Eng. B.* (2010).

[90] Taxi and NYC Limousine Commission. *TLC Trip Record Data*. NYC Taxi and Limousine Commission, 2023.

[91] Gian Alix et al. "A Mobility-based Recommendation System for Mitigating the Risk of Infection during Epidemics". In: *23rd IEEE MDM*. 2022.

[92] Ilya Loshchilov and Frank Hutter. "Fixing weight decay regularization in adam". In: (2017).

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

[94] Alberto Rossi et al. "Modelling Taxi Drivers' Behaviour for the Next Destination Prediction". In: *IEEE TITS* 21.7 (2020), pp. 2980–2989.

[95] Patrick Ebel et al. "Destination Prediction Based on Partial Trajectory Data". In: *IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1149–1155.

[96] Hoang Thanh Lam et al. "(Blue) Taxi Destination and Trip Time Prediction from Partial Trajectories". In: *ECML PKDD Discovery Challenge*. 2015, pp. 63–74.

[97] Chengwu Liao et al. "Taxi-Passenger's Destination Prediction via GPS Embedding and Attention-Based BiLSTM Model". In: *IEEE TITS* 23.5 (2022), pp. 4460–4473.

[98] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "Next Place Prediction Using Mobility Markov Chains". In: *MPM Workshop*. 2012. ISBN: 9781450311632.

[99] M. Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE TSP* 45.11 (1997), pp. 2673–2681.

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

[101] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *EMNLP*. Oct. 2014, pp. 1724–1734.