TrajGPT: Controlled Synthetic Trajectory Generation Using a Multitask Transformer-Based Spatiotemporal Model

Shang-Ling Hsu
University of Southern California
Los Angeles, California, USA
hsushang@usc.edu

Emmanuel Tung
Novateur Research Solutions
Ashburn, Virginia, USA
etung@novateur.ai

John Krumm
University of Southern California
Los Angeles, California, USA
jkrumm@usc.edu

Cyrus Shahabi
University of Southern California
Los Angeles, California, USA
shahabi@usc.edu

Khurram Shafique
Novateur Research Solutions
Ashburn, Virginia, USA
kshafique@novateur.ai

ABSTRACT

Human mobility modeling from GPS-trajectories and synthetic trajectory generation are crucial for various applications, such as urban planning, disaster management and epidemiology. Both of these tasks often require filling gaps in a partially specified sequence of visits, – a new problem that we call "controlled" synthetic trajectory generation. Existing methods for next-location prediction or synthetic trajectory generation cannot solve this problem as they lack the mechanisms needed to constrain the generated sequences of visits. Moreover, existing approaches (1) frequently treat space and time as independent factors, an assumption that fails to hold true in real-world scenarios, and (2) suffer from challenges in accuracy of temporal prediction as they fail to deal with mixed distributions and the inter-relationships of different modes with latent variables (e.g., day-of-the-week). These limitations become even more pronounced when the task involves filling gaps within sequences instead of solely predicting the next visit.

We introduce TrajGPT, a transformer-based, multi-task, joint spatiotemporal generative model to address these issues. Taking inspiration from large language models, TrajGPT poses the problem of controlled trajectory generation as that of text infilling in natural language. TrajGPT integrates the spatial and temporal models in a transformer architecture through a Bayesian probability model that ensures that the gaps in a visit sequence are filled in a spatiotemporally consistent manner. Our experiments on public and private datasets demonstrate that TrajGPT not only excels in controlled synthetic visit generation but also outperforms competing models in next-location prediction tasks–Relatively, TrajGPT achieves a 26-fold improvement in temporal accuracy while retaining more than 98% of spatial accuracy on average.

SIGSPATIAL '24, October 29-November 1, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1107-7/24/10
https://doi.org/10.1145/3678717.3691303

CCS CONCEPTS

• **Information systems** → **Location based services**; • **Computing methodologies** → *Mixture models*; *Bayesian network models*; **Neural networks**.

KEYWORDS

Spatiotemporal modeling, human mobility modeling, Synthetic Trajectory generation, Transformers

ACM Reference Format:
Shang-Ling Hsu, Emmanuel Tung, John Krumm, Cyrus Shahabi, and Khurram Shafique. 2024. TrajGPT: Controlled Synthetic Trajectory Generation Using a Multitask Transformer-Based Spatiotemporal Model. In *The 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3678717.3691303

1 INTRODUCTION

Modeling human mobility is important for understanding traffic, urban dynamics, commerce, health, and equity. Ideally, researchers and practitioners would have access to relevant, detailed visit sequences of large numbers of people. In reality, however, it is difficult to get a large volume of high-quality visit sequences, due to concerns about privacy, confidentiality, low-resolution measurements, missing observations, the cost of commercially available data, or minimal motivation for people to measure and share their location data.

To solve this problem, researchers and practitioners can attempt to fix low-quality visit sequences from real people, or they can generate completely synthetic visit sequences. Both approaches lead to the problem of filling gaps in the sequence. In the case of a real visit sequence, with missing parts due to privacy concerns, poor measurements, or dropouts, we have a partially specified sequence. The gaps can be filled with purely the most likely computed visits, or they can be filled with visits that meet some prior background knowledge of where the person went, such as a time-space cube. For instance, the gap may come with only an approximate location, such as somewhere in the vicinity of certain cell tower, which constrains the filled-in visits.

Likewise, for synthetic sequences, we may want to intentionally drop certain visits and replace them with other, loosely specified

visits to simulate certain behavior. This is a way to simulate temporarily popular hot spots (e.g. a concert) or travel to new points of interest or newly developed neighborhoods.

For both cases, filling visits in real data or replacing visits in synthetic data, the problem becomes one of replacing gaps with likely visits to complete the sequence, with possible constraints on the filled-in visits. We define this problem of filling gaps as a new challenge that we call "controlled" synthetic trajectory generation.

Gaps in the data present unique challenges, and to the best of our knowledge there are currently no methods designed for this problem. Traditional models for next-location prediction or synthetic generation are not equipped to effectively handle realistically filling gaps in a partially specified sequence. This is challenging for several reasons. First, sequence generation may be subject to certain pre-specified constraints, e.g., location and time of a hot spot being modeled. Second, the number of visits to insert into a gap is unspecified. Finally, the filled-in visits must be specified with not only a realistic visit location (based on an agent's history), but also an accompanying arrival time, visit duration, and travel time that conform to the location choice. For example, if the location choice is a dentist, we will likely not specify a 3 a.m. arrival time and a four-hour visit after an eight-hour drive.

Existing models for next-location prediction or synthetic trajectory generation lack the necessary mechanisms to constrain the generated sequences of visits. Typically, these models predict only the location of visits [34, 35]. While some recent methods have attempted to model both location and time [2, 36], they have significant limitations that affect their performance and the realism in generated trajectories:

Assumption of independence between location and time: Existing methods frequently treat location and time as independent factors, relying heavily on an independence assumption that fails to hold true in real-world scenarios. For example, suppose an agent leaves their office at lunchtime and is equally likely to visit either a coffee shop or a tea shop. It takes eight minutes to travel from the office to the tea shop, while it takes only two minutes to reach the coffee shop. A model that treats location and time independently might predict the mean travel time (five minutes) regardless of the actual destination, thereby introducing unrealistic artifacts into the generated trajectory. We visualize this example in Figure 1a.

Temporal accuracy challenges in single-value time predictions: Existing methods usually predict a single value of time, such as an expected value derived from regression [36] or the most probable value determined by the argmax of probability [2]. This approach can compromise the temporal accuracy of generated visit sequences. For instance, traffic congestion around a school tends to be significantly heavier on game days compared to other days. To accurately predict realistic arrival times while considering such factors, a model should implicitly distinguish between game days and non-game days instead of simply averaging the two possibilities. We illustrate this in Figure 1b.

These limitations are even more pronounced when the task involves filling gaps within sequences rather than solely predicting the next visit. Existing methods fail to offer effective solutions for either i) generating a realistic sequence of visits with joint spatiotemporal modeling or ii) adequately controlling the output of a regression model while adhering to strict constraints.

To address these issues, we propose TrajGPT, a transformer-based, multi-task, joint spatiotemporal generative model. TrajGPT leverages the transformer architecture to predict locations, while the visit duration and travel time between visits are approximated by taking into account the predicted location. Taking inspiration from recent large language models [1, 15], TrajGPT poses the problem of controlled trajectory generation as that of text infilling in natural language. By allowing the pre-fixing of specific locations and times within a sequence, TrajGPT can effectively fill in the gaps in a manner that maintains spatiotemporal consistency.

TrajGPT also learns the parameters of a Gaussian mixture to model the distributions of visit duration and travel time between visit locations. The integration of spatial and temporal models is facilitated through a Bayesian probability model, incorporated as a nonparametric joint likelihood loss function. This innovative approach ensures that TrajGPT can fill gaps and generate sequences that are both spatially and temporally consistent. It explicitly avoids the problems illustrated in Figure 1 due to its joint probability representation. This capability is crucial for applications requiring precise control over synthetic trajectory generation, such as simulating movement patterns in a partially known scenario.

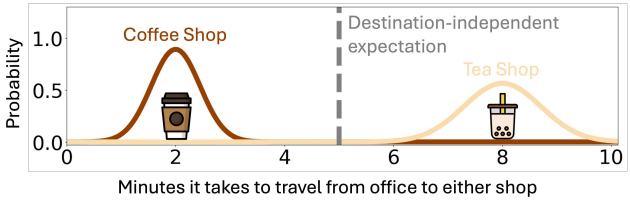
Our extensive experiments on both public and private datasets highlight the effectiveness of TrajGPT in controlled synthetic visit generation. The results demonstrate that TrajGPT not only excels in filling gaps within sequences but also outperforms competing models in next-location prediction tasks. This superior performance underscores the potential of TrajGPT to advance the field of human mobility modeling by providing a robust and flexible solution for generating controlled synthetic trajectories.[1]

In summary, the main contributions of this work are:

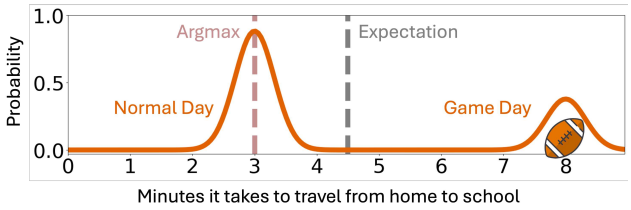
(1) We introduce the novel problem of "controlled" synthetic trajectory generation, which addresses the need to fill gaps in sequences with specific constraints on locations and times.
(2) We propose TrajGPT, a transformer-based, multi-task, joint spatiotemporal generative model that integrates a Gaussian mixture model and a Bayesian probability model to ensure spatiotemporal consistency and accuracy.
(3) We demonstrate the effectiveness of TrajGPT through extensive experiments on both public and private datasets, highlighting its superior performance in controlled synthetic visit generation and next-location prediction tasks compared to existing models.

In the remainder of the paper, we describe related work in Section 2. Section 3 gives a precise definition of the new problem we solve, and Section 4 presents our solution in the form of likelihood maximization with probability distributions computed from a transformer model. Section 5 presents our performance evaluation on trajectory data, showing how our approach is superior to the state-of-the-art alternatives, as well as an ablation study. We conclude in Section 6.

[1]The code is available at https://github.com/ktxlh/TrajGPT.



(a) Importance of probabilistic dependence between location and time: An agent at their office will travel to either the coffee shop or tea shop, each with equal probability. It takes eight minutes to reach the tea shop (lighter PDF curve) and two minutes to reach the coffee shop (darker PDF curve). Without knowing the destination, a model might predict an unrealistic five-minute travel time, the average of the two.



(b) Drawbacks of predicting expected value or most probable point: An agent traveling from home to school faces variable traffic based on whether a sports game is scheduled. Predicting the expected time yields a low probability value, while predicting the most probable timestamp misses all game day scenarios. Additionally, if the predicted arrival time falls outside a gap in a sequence of visits, there's no clear method to adjust it to fit within the gap.

Figure 1: Examples motivating (a) spatiotemporal joint probability modeling and (b) Gaussian mixture models, with their corresponding probability density functions (PDFs).

2 RELATED WORK

Human mobility data can be categorized into two types: point-based and visit-based. While both are often grouped under the term "trajectories," they differ significantly. Point-based data consists of a sequence of observations from sensors tracking an object's movement, such as GPS signals from a mobile phone [7, 10, 16, 41] or a car's navigation system [11, 26, 38]. This type of data is typically dense, with observations collected at frequent intervals (e.g., every 30 seconds) and includes raw coordinates (latitude and longitude) without any associated semantic information. On the other hand, visit-based data captures the sequence of places an individual visits. Each visit includes details such as location (both latitude/longitude and semantic information like points of interest), arrival time, and departure time.

For forecasting tasks such as next location prediction [12, 19, 42], point-based data is rich and easier to predict due to its short and regular inter-point intervals and additional contextual knowledge like road networks [24]. However, this type of sequence is not the focus of our paper.

Our focus is on visit-based data, which is typically sparser and more challenging to predict. This data is often collected through check-ins (e.g., from Foursquare [9] and Gowalla [3]) and usually includes 3-5 visits per person per day with irregular inter-arrival times. Research on visit-based sequences often centers on the downstream task of predicting the next location [2, 8, 21, 32–36, 39, 40]. These locations are typically represented as Points-of-Interest (POIs). Among the papers on POI recommendation, DeepJMT [2], MobTCast [35], GETNext [36], and STAR-HiT [34] are the most relevant to our work as they employ transformers [31] as the underlying encoder. While MobTCast and STAR-HiT predict the subsequent POI without considering check-in time, MobTCast integrates various contextual factors, including temporal, semantic, social, and geographical contexts, alongside a consistency loss mechanism. Conversely, STAR-HiT, featuring a hierarchical transformer architecture, employs stacked encoders and subsequence fusion modules to capture multi-granularity spatiotemporal patterns within user check-in sequences, facilitating interpretability.

DeepJMT and GETNext predict location and time independently, although the time spent at a location (e.g., coffee shop vs. gym) and the time between locations are highly dependent on the specific locations. For instance, GETNext employs a deterministic approach to predict a single temporal value, while DeepJMT predicts a temporal distribution during training and predicts the timestamp with the highest probability for inference. While the assumption of independence might not significantly impact the task of next location prediction, it poses a challenge when predicting the arrival time or duration of the next visit, as demonstrated in our experiments (see Section 5) using these approaches as baselines. This issue becomes even more pronounced when trying to fill in the visits between known visits, which is the primary focus of our paper.

Although no existing work in human mobility modeling directly addresses the task of filling visit gaps, related concepts can be found in language modeling. Following the introduction of the transformer model and large language models (LLMs) [1, 5, 17, 22, 28, 37], subsequent research adapted its encoder architecture for masked language modeling (MLM), exemplified by BERT [15], and its autoregressive decoder for causal modeling, as demonstrated by GPT [27]. However, these popular approaches each have their drawbacks when it comes to filling in gaps. BERT can only fill known-length gaps, which is inadequate given the variability in spans of gaps. Conversely, GPT relies solely on the preceding context, lacking the ability to leverage information following a gap in a sequence. Consequently, neither model effectively addresses the challenge of infilling variable-sized gaps constrained by contextual factors. Some previous studies [4, 6, 29] have tackled this unknown-length blank infilling problem; thus, we follow the infilling paradigm from natural language processing (NLP) for our specific task of inferring human visit sequences. Nevertheless, this infilling approach cannot be directly applied to our problem due to the absence of a spatial and temporal component, which is crucial in our context where location, visit duration, and inter-arrival timings are important. Therefore, we leverage Space2Vec [23] and Time2Vec [14] for spatiotemporal representation learning and design spatiotemporal joint prediction for controlled synthetic trajectory generation.

r_i	the region where visit i is located, such as a grid cell
t_i^a	the arrival time of visit i
t_i^d	the departure time of visit i
x_i	an attributes tuple (r_i, t_i^a, t_i^d) of visit i
X	a sequence of contiguous visits $X = [x_1, x_2, ..., x_i]$
X'	a subsequence of X, input for tasks
P	a true possibility mass or density function
$\hat{P}$	an approximated possibility mass or density function
H	a sequence of visit embeddings
$\Delta t_i^{\mathcal{T}}$	the travel time from visit $i-1$ to i
$\Delta t_i^{\mathcal{D}}$	the duration of visit i

Table 1: Notable notations used in this article.

Geo-CETRA [20], a recent study, also addresses the problem of constraint-based trajectory generation, but our work differs in several key ways. In Geo-CETRA, the constraints are defined as spatiotemporal ranges that the synthetic trajectory must satisfy, whereas we define constraints as a set of known visits that the trajectory is required to pass through. As a result, Geo-CETRA focus on identifying realistic visits within the spatiotemporal boundaries, while our approach, inspired by language models, concentrates on filling in the visits between these fixed points. Due to the nature of our discrete constraints, we discretize space into grid cells to construct a "vocabulary" for our model, whereas Geo-CETRA operate directly in a continuous spatiotemporal space.

3 PROBLEM STATEMENT

We give a precise definition of our problem here, followed by our solution in Section 4.

3.1 Terminology

We formally define a *visit* as a tuple $x = (r, t^a, t^d)$, where r represents the location of the visit, such as a region or a Point-of-Interest (POI), t^a represents the arrival time of the visit, and t^d represents the departure time of the visit. A sequence of visits X contains all visits made by a single agent within a time range. We use P to denote the *true* possibility mass function (PMF) or possibility density function (PDF), and $\hat{P}$ to denote the PMF or PDF approximated by TrajGPT. We summarize the notations in Table 1.

3.2 Controlled Synthetic Trajectory Generation

The main problem we solve, "Controlled Synthetic Trajectory Generation," is to, given an incomplete sequence of visits X', predict the missing visits within X'. Let X be a (complete) sequence of visits. An incomplete sequence of visits, X', refers to any sub-sequence of X that is missing at least one visit. If each contiguous span of missing visits is replaced with a placeholder marker (i.e., a blank), then, the task is to predict the missing visits $\hat{X}$ for each blank, specifying both the temporal ordering of such predicted missing visits and the correspondence of the predicted missing visits to the blanks. Given the predicted missing visits $\hat{X}$ and incomplete visit sequence X', it is trivial to construct the resultant (complete) sequence of visits. If x is the first missing visit within the incomplete visit sequence X', the probability distribution $P(x|X')$ can be rewritten as $P(x|X') = P(r, t^a, t^d|X') = P(r|X')P(t^a|X', r)P(t^d|X', r, t^a)$ according to the chain rule of probability.

3.3 Next Visit Prediction

As a byproduct of solving "Controlled Synthetic Trajectory Generation," we can also solve "Next Visit Prediction," which is a generalized version of the traditional "Next Location Prediction" task. Given context X' which consists of a contiguous sequence of visits, we model not only the probability distribution $P(r|X')$ of the next visit's location r, but also the probability distribution $P(t^a|X', r)$ of the arrival time t^a of the visit, as well as the probability distribution $P(t^d|X', r, t^a)$ of the departure time t^d of the visit. Therefore, given $X = [x_1, ..., x_{i-1}]$, which is a sequence of consecutive visits, we predict the next visit x_i, which includes its region r_i, its arrival time t_i^a, and its departure time t_i^d. We make the trivial observation that the Next Visit Prediction task is a special case of the Controlled Synthetic Trajectory Generation task where the only missing visits, $X \setminus X'$, are those occurring in the future, following the last visit in X'.

4 METHODOLOGY

In this section, we introduce our solution to the problem of "Controlled Synthetic Trajectory Generation." To utilize techniques from autoregressive sequence modeling, we begin by rearranging each visit sequence, enabling the use of an autoregressive model for the infilling task (Section 4.1). Subsequently, we design a spatiotemporal autoregressive model that learns its parameters from these rearranged visit sequences (Section 4.2).

4.1 Visit Infilling

In order to: (1) train the model to be capable of infilling any number of items into each blank, with one or more blanks at any point in the sequence, and (2) take advantage of the auto-regressive nature of transformers and allow efficient training of the model to do both infilling and next-item prediction, we restructure our sequence data for the infilling task (Section 3.2), following the approach outlined by Donahue et al. [4]. Each visit sequence X is composed of visits $X = [x_1, ..., x_n]$, and each visit $x_i = (r_i, t_i^a, t_i^d)$ is defined by its region, arrival time, and departure time, respectively. The dataset as it is in its innate form, which consists of many visit sequences, can be used to train a transformer model to predict the next visit, given a partial visit sequence. We then reframe our data for the infilling task by applying the following process to each visit sequence $X = [x_1, ..., x_n]$.

To rearrange a visit sequence for the infilling task, we first add a special `SEP` non-visit token to the end of the sequence to denote the end of the original sequence. Then, we sample a Bernoulli distribution for each visit except for the first and last $(x_i; i \in \{2, 3, ..., n-1\})$. Sampling a 1 means we drop the visit, and a 0 means we retain the visit. For each contiguous span of visits we dropped, we insert a single `BLANK` token where the span used to be located within the sequence. Next, for each span we dropped, we append that span and an `ANS` token to the end of the sequence; the `ANS` token marks the end of each span. In this way, the reframed sequence contains

the partially specified sequence in the first half and the ground truth filled-in visits in the second half. Since this is an infilling task, we never drop the first or last visit. See Figure 2 for an example.

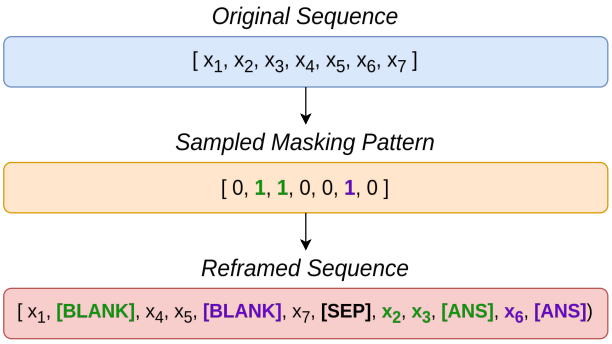


Figure 2: Reframing sequence data for infilling.

The aim of the model is to predict the values after the SEP token in order to complete a sequence. This coincides with the prediction of the missing items. By rearranging each sequence and marking special delimiters (BLANK, SEP, ANS), an autoregressive transformer model can learn to attend to the positions of these special tokens. In doing so, it can start infilling any number of items for the first blank after the SEP token, declaring the first blank to have been completely infilled by predicting an ANS token, and repeating this process for subsequent blanks until the number of ANS tokens matches the number of BLANK tokens. The task of interest is to predict the visits in the reframed sequence after the SEP token. Note that for our training, validation, and test split data, we assume that we have access to complete visit sequences, to which we can apply this reframing process. However, at inference time, we have incomplete visit sequences to be infilled, which constitute the visits before the SEP token.

4.2 Spatiotemporal Joint Modeling

In this section, we discuss the architecture of TrajGPT and explain the process by which it learns the model parameters. We use this architecture and learning process for both Controlled Synthetic Trajectory Generation and Next Visit Prediction. It is important to note that we employ teacher forcing throughout the training phase. For instance, when predicting an arrival time, we use the actual region as input rather than a predicted one.

4.2.1 Formulation. We derive a probabilistic model for spatiotemporal autoregressive sequence modeling as follows. As operationalized in Section 3.2, to predict the remaining visits $X \setminus X'$ given X', we parameterize a function $\hat{P}$ to approximate the conditional joint probability $P(x \mid X')$, denoting the parameters as θ, and learn it with maximum likelihood estimation (MLE):

$$\theta^* = \arg\max_{\theta} \prod_{x \in X \setminus X'} \hat{P}_\theta(x \mid X') \tag{1}$$

For simplicity, throughout this article, X' evolves as we add new, inferred visits, and we will omit θ from our notation going forward.

To achieve the approximation, we first factorize the targeted joint probability using Bayes' Rule:

$$\begin{aligned} P(x \mid X') &= P(r,\ t^a,\ t^d \mid X') \\ &= P(r \mid X')\ P(t^a \mid X',\ r)\ P(t^d \mid X',\ r,\ t^a) \end{aligned} \tag{2}$$

To approximate these factors, for each visit, we make TrajGPT approximate the distribution of each attribute of x one by one as follows. In other words, TrajGPT predicts region, arrival time, and departure time of a visit sequentially, taking all previous predictions into consideration when making a new prediction.

(1) Approximate region $P(r \mid X')$
(2) Conditioned on region, approximate arrival time $P(t^a \mid X',\ r)$
(3) Conditioned on region and arrival time, approximate departure time $P(t^d \mid X',\ r,\ t^a)$

We will elaborate on the realization of these steps in the subsequent sections. To define a loss function that encourages TrajGPT to predict the truth, we follow the Maximum Likelihood Estimation (MLE) paradigm and compute the negative log likelihood of predicting the ground truth for each of these approximated distributions, such as $-\log \hat{P}(r_n \mid X')$. Combining these likelihood variables with Equation 1 and 2, we obtain this elegant, non-parametric, negative log likelihood loss function for the joint probability[2]:

$$\mathcal{L} = -\sum_{r,t^a,t^d} \log \hat{P}(r \mid X') + \log \hat{P}(t^a \mid X', r) + \log \hat{P}(t^d \mid X', r, t^a) \tag{3}$$

where L stands for *loss* and sums over $(r, t^a, t^d) \in X \setminus X'$.

4.2.2 Model Architecture. We illustrate the architecture of TrajGPT in Figure 3. The process begins with fusing the spatiotemporal information in the subsequence of visits X' using a transformer encoder (Section 4.2.3). Following this, the region head module predicts the region r of the visit (Section 4.2.4) as a discrete probability mass function over possible visit locations. Subsequently, the model embeds and conditions on the predicted region to forecast the travel time of the visit using the travel time head. The travel time is then arithmetically converted to arrival time (Section 4.2.5). The arrival time is encoded and fed to the duration head to predict the duration of the visit. Finally, the duration is converted to departure time through arithmetic operations (Section 4.2.6).

4.2.3 Sequence Encoder. We design a sequence encoder to help TrajGPT understand complex spatiotemporal sequences. For each visit, we use Space2Vec [23] to encode the location l_i, known as location encoding, and Time2Vec [14] to encode the arrival and departure times, referred to as arrival and departure time encoding. To guide the model in recognizing region-specific information, such as land use, we embed the region where each visit occurs and make this embedding learnable, referring to it as region embedding. If a visit is a "special token" visit, as described in Section 4.1, we use the embedding of the special token instead of a region embedding since this pseudo visit does not contribute spatiotemporal information to the sequence. We then concatenate the location encoding, arrival time encoding, departure time encoding, and region or special token embedding. This sequence of concatenated embeddings is fed into

[2]For special tokens (see Section 4.1), region loss is replaced with the special token loss, and there is no temporal loss.

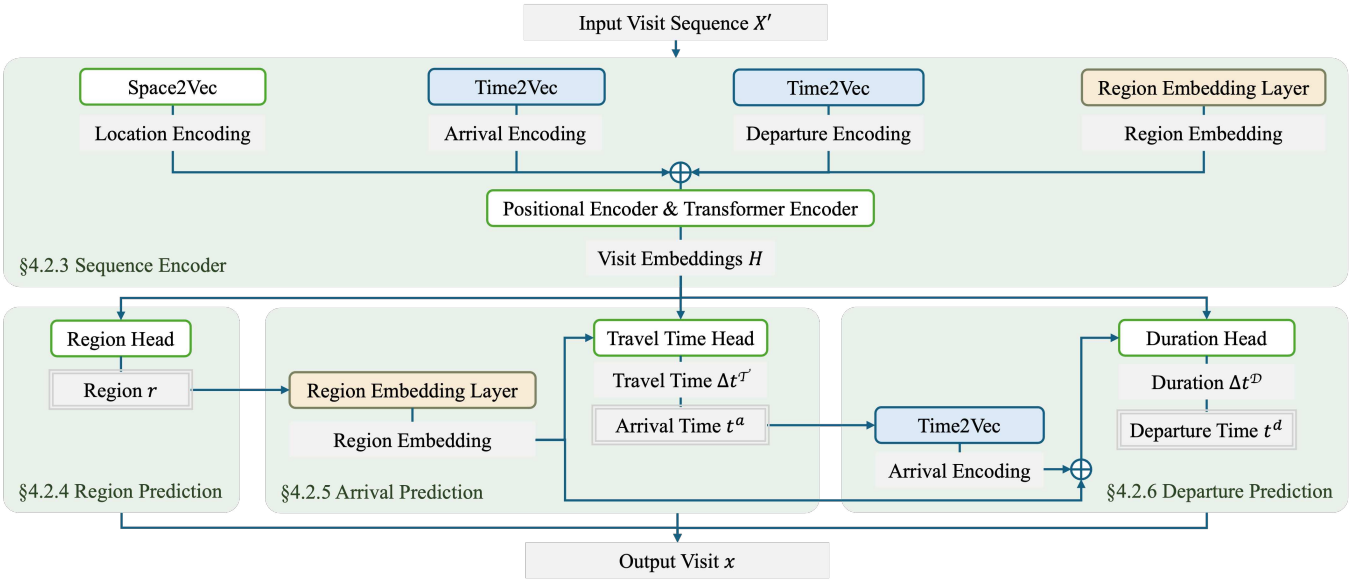


Figure 3: Overview of TrajGPT. Modules that share parameters are colored in the same shade.

the positional and transformer encoders proposed by [31]. The sequence of outputs from the transformer encoder will be referred to as visit embeddings H, which entails an implicit summary of the input sequence.

$$H := \text{TransformerEncoder}(\text{PositionalEncoder}(X')) \tag{4}$$

4.2.4 Region Prediction. We formulate the region prediction task as a classification problem. To predict the region where a visit is located, we feed the visit embeddings H to the region head, which contains another transformer encoder, a linear layer, and a softmax function for creating a proper probability mass function.

$$\hat{P}(r_i \mid X') := \text{Softmax}(\text{Linear}(\text{TransformerEncoder}(H))) \tag{5}$$

4.2.5 Arrival Time Prediction. To account for spatiotemporal dependencies, as shown in Figure 1a, we condition our arrival time predictions on the predicted region.

Arithmetically, to predict the arrival time t_i^a of visit i, we first derive the travel time $\Delta t_i^{\mathcal{T}}$ from the preceding visit $i-1$ to i, the current one we are predicting.

$$\Delta t_i^{\mathcal{T}} = t_i^a - t_{i-1}^d \tag{6}$$

To predict $\Delta t_i^{\mathcal{T}}$, we approximate its potentially complex distribution, as illustrated in Figure 1b, using a Gaussian Mixture Model (GMM). We will show that this approximation effectively models travel time based on visit-to-visit observations in the training data in Section 5.

$$\hat{P}(\Delta t_i^{\mathcal{T}} \mid X',\ r_i) := P_i^{\mathcal{T}}(\Delta t_i^{\mathcal{T}}) \tag{7}$$

where $P_i^{\mathcal{T}} : \mathbb{R} \to \mathbb{R}^+$ is the probability density function (PDF) of the GMM[3]. To predict the parameters of the GMM, denoted as $\text{Param}(P_i^{\mathcal{T}})$, we emulate the decoder of transformer [31] to enable cross attention between r_i and H:[4]

$$\text{Param}(P_i^{\mathcal{T}}) := \text{FF}(\text{MHA}(e_i,\ H,\ H)) \tag{8}$$

where $e_i = \text{Embedding}(r_i)$ is embedded using the same region embedding layer as Section 4.2.3; FF denotes feedforward neural networks;[5] MHA stands for multi-head attention, which projects $\mathbf{r_i}$ to *queries*, and H to *keys* and *values* to perform cross attention:

$$\begin{aligned} \text{MHA}(e_i, H, H) &:= \text{Concat}(\text{head}_1, ..., \text{head}_M) W^O \\ \text{where head}_j &:= \text{Attention}(e_i W_j^Q, H W_j^K, H W_j^V) \end{aligned} \tag{9}$$

where *Concat* denotes concatenation of vectors; $W_j^Q; W_j^K, W_j^V$ are parameter matrices.

4.2.6 Departure Time Prediction. Similar to how we predict arrival time in Section 4.2.5, we first derive duration $\Delta t_i^{\mathcal{D}}$

$$\Delta t_i^{\mathcal{D}} = t_i^d - t_i^a \tag{10}$$

Then, we use cross attention and a GMM to predict the duration $\Delta t_i^{\mathcal{D}}$ of the visit

$$\hat{P}(\Delta t_i^{\mathcal{D}} \mid X',\ r_i,\ t_i^a) := P_i^{\mathcal{D}}(\Delta t_i^{\mathcal{D}}) \tag{11}$$

where $P_i^{\mathcal{D}} : \mathbb{R} \to \mathbb{R}^+$ is the PDF of the GMM. Different from arrival time prediction, instead of using only the embedding of r_i as the query for MHA, we first concatenate $e_i = \text{Embedding}(r_i)$ with the encoding of t_i^a as input c_i

$$c_i := \text{Concat}(e_i, \text{Time2Vec}(t_i^a)) \tag{12}$$

[3]For inference, we clip the distribution by setting the probability of negative values to zero and re-normalizing it.

[4]For brevity, we omit the residual connections and normalization layers in Equation 8 and 13. For details, please refer to Section 3 of the transformer paper [31].

[5]To ensure the predicted weights and scales of the GMM are always positive, we apply a softplus function and add a small positive value to the output of the feedforward network. Equations are omitted for brevity.

where the encoding of t_i^a is generated by the Time2Vec encoder in Section 4.2.3. Then, we compute the cross attention between c_i and H to approximate the parameters of the GMM.

$$\text{Param}(P_i^{\mathcal{D}}) \;:=\; \text{FF}(\text{MHA}(c_i,\ H,\ H)) \tag{13}$$

$$\begin{aligned} \text{MHA}(c_i, H, H) &:= \text{Concat}(\text{head}_1, ..., \text{head}_M)W^O \\ \text{where head}_j &:= \text{Attention}(c_i W_j^Q, HW_j^K, HW_j^V) \end{aligned} \tag{14}$$

In summary, in this section, we described how we train TrajGPT to approximate each probability function of the joint probability in Equation (2): $P(r \mid X')\ P(t^a \mid X',\ r)\ P(t^d \mid X',\ r,\ t^a)$.

4.2.7 Inference. To conduct inference with a model trained using the above methodology, one can replace teacher forcing with autoregression. In other words, to predict one visit, one can follow these steps:

(1) Conditioned on X', predict the region of the visit, denoted $\hat{r}$.
(2) Conditioned on X', $\hat{r}$, predict the arrival time, denoted $\hat{t}^a$.
(3) Conditioned on X', $\hat{r}$, $\hat{t}^a$, predict the departure time.

Since the effectiveness of such an autoregressive procedure depends on the choice of a decoding algorithm, such as beam search [25] or nucleus sampling [13], which is not the focus of this work, we resort to evaluating TrajGPT with teacher forcing in Section 5.

5 EVALUATION

5.1 Experimental Setup

5.1.1 Data. We employed two trajectory datasets, GeoLife and MobilitySim, for our experiments. GeoLife [41] is a public real-world trajectory dataset based in Beijing, featuring data from 102 agents collected between 2008 and 2009. To demonstrate the scalability of our approach, we also utilized a private, simulated trajectory dataset, MobilitySim. The dataset contains a realistic simulation of 2,000 agents performing daily activities in San Francisco, over a period of 30 days. The simulation contains second-by-second location of each agent, as they perform recurring daily activities, such as going to school or work, as well as occasional recreation and maintenance activities, such as visits to restaurants, gym, and doctors office. The simulation also incorporates daily and weekly patterns, such as work schedules and days off. We summarize the dataset statistics in Table 2.

5.1.2 Processing. To convert point-based trajectories into visit-based sequences, we first identify visits [18]. A visit is operationally defined as a (location, arrival time, departure time) tuple, describing where and when an agent remains stationary for a contiguous period of time. For GeoLife, we identify visits spatially within a 200-meter radius and temporally for a minimum duration of 10 minutes. For MobilitySim, we identify visits for each agent based on a minimum period of 6 minutes during which the agent remains perfectly stationary. After identifying visits, we form "regions" by discretizing the locations using Uber's H3 index [30]. For GeoLife, we set the Uber H3 Resolution to 7, and for MobilitySim, we set it to 10. We convert the latitude and longitude of visits to the Universal Transverse Mercator (UTM) coordinate system to ensure the two dimensions of the geographical coordinates are on the same scale (in meters). For timestamps, including arrival and departure times, we subtract the oldest arrival time in each dataset from all other timestamps, converting these time differences into seconds. To prevent exploding gradients during training, we normalize the duration and travel time: the duration is scaled to days, and the travel time is scaled to hours.

For Controlled Synthetic Trajectory Generation (Sections 5.2 and 5.4), we treat each agent's visit sequence as an individual instance and divide the set of agents into training, validation, and test sets in an 8:1:1 ratio. Following a strategy similar to *dynamic masking* in RoBERTa [22], we treat "masking each visit" as an independent Bernoulli trial with a 20% probability. However, after dynamic masking, we replace each contiguous subsequence of masked visits with a `BLANK`. The model is then tasked with predicting an unknown number of visits for each blank.

For Next Visit Prediction (Section 5.3), we followed previous work [34, 36] by using a rolling window to extract instances, sorting them chronologically, and splitting them into training, validation, and test sets in an 8:1:1 ratio. We set the size of the rolling window to 128 visits, following [34].

5.1.3 Metrics. We evaluate the models with teacher forcing for these metrics: **Acc@k** presents the top-k accuracy for location prediction. Note that we report the evaluation on infilling location predictions, not on the predicting the special `ANS` token which indicates the model is finished predicting for the corresponding blank. $\mathbf{P}_{\pm}t$ shows the proportion (for scalar[6]) or probability (for distribution) of predictions that fall into the $g \pm t$ minutes interval, where g is the ground truth.

5.1.4 Baselines. Since controlled synthetic trajectory generation is a new task we propose, we resort to compare with studies in next POI recommendation. We selected the following state-of-the-art baselines:

- **STAR-HiT** [34]: Hierarchical transformer for next POI recommendation with subsequence aggregation technique.
- **GETNext** [36]: Transformer for next POI recommendation with an auxiliary next check-in time prediction task, assuming next POI and next check-in time are independent.

Note that GETNext uses POI category information. As our datasets are generated from raw trajectories, not sequences of POIs, they do not contain such information. Hence, throughout this section, we remove its POI category components.

5.2 Controlled Synthetic Trajectory Generation

Since controlled synthetic trajectory generation is a new task we proposed, we aimed to evaluate the effectiveness of TrajGPT compared to existing models. For this purpose, we selected GETNext [36], the state-of-the-art model for human mobility that concurrently models both space and time. We adapted GETNext for the visit infilling task, naming it GETNext*, by incorporating special-token visits into the input sequences, as detailed in Section 3, and adding an additional departure time head with the same architecture as its original arrival time head.

As shown in Table 3, TrajGPT maintains similar accuracy in region prediction while achieving significantly higher accuracy in

[6]For fair comparison with clipped distribution (see Footnote 3), we replace negative predictions with zeros.

Dataset	#agent	#region	#visit	#trajectory*	Avg. #visit/agent	Avg. #visit/region
GeoLife	102	1,369	20,278	11,724	198.80	14.81
MobilitySim	2,000	3,481	191,963	6,178	95.98	55.15

Table 2: Data Statistics. #trajectory denotes the number of trajectories for next visit prediction (see Section 5.1.2).

arrival and departure time prediction. This outcome is expected because, although GETNext models both the next point of interest (POI) and the next check-in time, its primary focus is on next POI prediction. In fact, GETNext does not report metrics for temporal prediction, which explains the inferior accuracy in its temporal predictions compared to TrajGPT.

5.3 Next Visit Prediction

To ensure a fair comparison with existing approaches, we also adapted TrajGPT to predict the next visit rather than filling in gaps. This modification aligns the task more closely with next POI recommendation, which the baseline models are specifically designed for. The results of this comparison are presented in Table 4. Certain cells within the table intentionally remain unpopulated due to the inherent characteristics of STAR-HiT and GETNext: STAR-HiT is not designed to predict timestamps; GETNext predicts only one timestamp for each visit, and we opt to forecast the arrival time.

In the domain of next visit prediction, TrajGPT exhibits notable superiority over GETNext in temporal forecasting, with minimal adverse effects on its region prediction performance. Notably, TrajGPT achieves this without relying on the supplementary trajectory flow map and transition attention map proposed by GETNext. Furthermore, both GETNext and TrajGPT significantly outperform STAR-HiT, this suggests that learning a multi-task, spatiotemporal model, might help predict locations better.

The use of teacher forcing ensures that TrajGPT has access to the actual region when predicting arrival times, whereas GETNext, by design, lacks this advantage as it predicts both region and arrival time simultaneously and independently. To peek into the potential of TrajGPT during inference without teacher forcing, imagine the worst case: If the predicted, most-probable region were incorrect, the arrival time accuracy would be zero. In this case, we can multiply the arrival time accuracy of TrajGPT with its Acc@1 of region prediction. This minimum threshold of its arrival time accuracy will still surpass that of GETNext by a significant margin.

5.4 Ablation Study

To demonstrate the effectiveness of the key components of TrajGPT, we conducted an ablation study for the infilling task, with results shown in Table 5.

- The **TrajGPT w Independence** variant predicts the region, arrival time, and departure time independently, reflecting the spatiotemporal-independence assumption made by DeepJMT [2] and GETNext [36]. This assumption is expressed as:

$$P(r_i, t_i^a, t_i^d \mid X') = P(r_i \mid X')P(t_i^a \mid X')P(t_i^d \mid X') \tag{15}$$

- The **TrajGPT w Regression** variant replaces the GMM used for predicting arrival and departure times (Sections 4.2.5 and 4.2.6) with a regression head, mimicking the approach used by GETNext [36].

The results demonstrate that TrajGPT significantly outperforms both variants in predicting arrival and departure times, while also maintaining exceptional accuracy in regional predictions. This underscores the effectiveness of the proposed spatiotemporal modeling approach.

Replacing joint modeling with independent modeling greatly reduces the accuracy of departure time predictions. This suggests that the duration of a visit, which influences the departure time, varies significantly depending on the visit's location (i.e. the region).

Replacing GMM with regression significantly weakens the model's ability to predict time, highlighting the importance of learning a time distribution rather than relying on a single point estimate. The accuracy drop is particularly pronounced for departure time predictions, suggesting that predicting a point is even less suitable for duration than for travel time.

The performance drop appears more pronounced for temporal predictions than for region predictions. This may be because each variant *directly* impacts temporal predictions by altering either the input (TrajGPT w Independence) or the output (TrajGPT w Regression), whereas region predictions are only *indirectly* affected through the combined influence of the loss function and optimization process.

In summary, the ablation study demonstrates that each of the two key components of TrajGPT, including joint modeling and GMM-based temporal distribution learning, plays a crucial role in achieving high prediction accuracy. Removing or replacing these components leads to a substantial decrease in performance, further confirming their necessity in capturing the complex spatiotemporal patterns of human mobility trajectory data.

6 CONCLUSION

In this paper we introduced the novel problem of "controlled" synthetic trajectory generation, addressing the need to fill gaps in visit sequences with specific constraints on locations and times. Filling gaps is useful for imputing missing data and for generating synthetic visit sequences that have some preordained visits. The task is challenging because the filled-in visits, along with travel times, must fill the gap exactly, and the visit locations and durations must be realistic.

As a solution we presented TrajGPT, a transformer-based, multi-task, joint spatiotemporal generative model. TrajGPT leverages the transformer architecture to predict locations while separately approximating the visit duration and travel time between visits using a Gaussian mixture model. This innovative approach ensures that TrajGPT can generate sequences that are both spatially and temporally realistic by adhering to the statistical dependencies between visit locations, visit durations, and travel times.

Dataset	Method	Region				Arrival Time			Departure Time		
		Acc@1	Acc@5	Acc@10	Acc@20	$P_{\pm5}$	$P_{\pm10}$	$P_{\pm20}$	$P_{\pm5}$	$P_{\pm10}$	$P_{\pm20}$
GeoLife	GETNext*	7.64	36.87	**46.83**	51.87	0.16	0.38	0.73	1.41	2.87	7.91
	TrajGPT	**19.87**	**40.57**	44.78	**52.21**	**65.72**	**75.53**	**85.31**	**36.08**	**50.81**	**64.23**

Table 3: Comparison of TrajGPT for the infilling task with GETNext*, which we adapted from GETNext, showing the effectiveness of TrajGPT. The best results are highlighted in bold.

Dataset	Method	Region				Arrival Time			Departure Time		
		Acc@1	Acc@5	Acc@10	Acc@20	$P_{\pm5}$	$P_{\pm10}$	$P_{\pm20}$	$P_{\pm5}$	$P_{\pm10}$	$P_{\pm20}$
GeoLife	STAR-HiT	17.92	40.78	48.98	56.40	-	-	-	-	-	-
	GETNext	**38.74**	**66.38**	**72.78**	77.13	2.39	4.27	8.62	-	-	-
	TrajGPT	35.06	62.08	70.77	**77.63**	**64.28**	**71.70**	**80.02**	**35.01**	**48.40**	**60.82**
MobilitySim	STAR-HiT	42.79	62.87	70.25	75.74	-	-	-	-	-	-
	GETNext	51.46	**80.91**	91.59	94.50	1.29	2.91	5.50	-	-	-
	TrajGPT	**54.53**	80.26	**92.88**	**94.66**	**89.33**	**94.01**	**98.07**	**52.57**	**62.05**	**71.20**

Table 4: Comparison of TrajGPT with baseline models for the next visit prediction task. The best results are highlighted in bold.

Dataset	Method	Region				Arrival Time			Departure Time		
		Acc@1	Acc@5	Acc@10	Acc@20	$P_{\pm5}$	$P_{\pm10}$	$P_{\pm20}$	$P_{\pm5}$	$P_{\pm10}$	$P_{\pm20}$
MobilitySim	TrajGPT w Independence	39.71	79.52	83.21	85.80	68.77	81.91	90.54	31.85	40.48	46.33
	TrajGPT w Regression	43.61	80.87	**86.31**	**90.40**	31.49	56.91	78.16	1.53	3.21	6.42
	TrajGPT	**44.15**	**81.04**	86.11	89.71	**73.71**	**84.40**	**91.67**	**42.65**	**50.45**	**57.70**

Table 5: Comparison of TrajGPT with its variants for the infilling task, demonstrating the effectiveness of TrajGPT's design. The best results are highlighted in bold.

We validated our approach on a public and private dataset, comparing against state-of-the-art methods for predicting next locations. We observed that TrajGPT not only demonstrates proficiency in gap filling but also surpasses competing methods in predicting the next visit. On average, TrajGPT achieves a remarkable 26-fold enhancement in temporal prediction accuracy while preserving over 98% of the spatial accuracy achieved by state-of-the-art approach.

ACKNOWLEDGMENTS

Research supported by the Intelligence Advanced Research Projects Activity (IARPA) via the Department of Interior/Interior Business Center (DOI/IBC) contract number 140D0423C0033. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA or the U.S. Government. This research project has benefited from the Microsoft Accelerate Foundation Models Research (AFMR) grant program through which leading foundation models hosted by Microsoft Azure along with access to Azure credits were provided to conduct the research.

REFERENCES

[1] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. 1877–1901.

[2] Yile Chen, Cheng Long, Gao Cong, and Chenliang Li. 2020. Context-aware deep model for joint mobility and time prediction. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 106–114.

[3] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 2011. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1082–1090.

[4] Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling Language Models to Fill in the Blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2492–2501.

[5] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems* 32 (2019).

[6] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General Language Model Pretraining with Autoregressive Blank Infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 320–335.

[7] Nathan Eagle and Alex Pentland. 2006. Reality mining: sensing complex social systems. *Personal and ubiquitous computing* 10 (2006), 255–268.

[8] Jun Feng, Xiang Li, Ji Zhang, Changqing Zhang, Jun Han, and Ke Li. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *Proceedings of the 2018 World Wide Web Conference*. ACM, 1459–1468.

[9] Foursquare. 2018. Foursquare Check-in Dataset. https://sites.google.com/site/yangdingqi/home/foursquare-dataset Accessed: 2024-06-04.

[10] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. 2008. Understanding individual human mobility patterns. *nature* 453, 7196 (2008), 779–782.

[11] Hanshin Expressway Company Limited. 2024. Zen Traffic Data. https://zen-traffic-data.net/english/ Accessed: 2024-06-04.
[12] Wenchong He, Zhe Jiang, Tingsong Xiao, Zelin Xu, Shigang Chen, Ronald Fick, Miles Medina, and Christine Angelini. 2024. A hierarchical spatial transformer for massive point samples in continuous space. *Advances in Neural Information Processing Systems* 36 (2024).
[13] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751* (2019).
[14] Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. 2019. Time2vec: Learning a vector representation of time. *arXiv preprint arXiv:1907.05321* (2019).
[15] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.
[16] Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha Laurila. 2010. Towards rich mobile phone datasets: Lausanne data collection campaign. *Proc. ICPS, Berlin* 68, 7 (2010).
[17] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.
[18] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*. 1–10.
[19] Haowen Lin, Yao-Yi Chiang, Li Xiong, and Cyrus Shahabi. 2024. Unified Modeling and Clustering of Mobility Trajectories with Spatiotemporal Point Processes. In *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 625–633.
[20] Haowen Lin, John Krumm, Cyrus Shahabi, and Li Xiong. 2024. Controllable Visit Trajectory Generation with Spatiotemporal Constraints. In *2024 IEEE International Conference on Data Mining (ICDM)*. IEEE.
[21] Xin Liu, Xiangnan He, Bingzhe Tian, Jianhui Wang, and Tat-Seng Chua. 2019. STAN: Spatio-Temporal Attention Network for Next Location Recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. ACM, 2193–2196.
[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[23] Gengchen Mai, Krzysztof Janowicz, Bo Yan, Rui Zhu, Ling Cai, and Ni Lao. 2020. Multi-Scale Representation Learning for Spatial Feature Distributions using Grid Cells. In *8th International Conference on Learning Representations, ICLR 2020*.
[24] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 336–343.
[25] Peng Si Ow and Thomas E Morton. 1988. Filtered beam search in scheduling. *The International Journal Of Production Research* 26, 1 (1988), 35–62.
[26] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2022. CRAWDAD epfl/mobility. https://doi.org/10.15783/C7J010
[27] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
[28] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.
[29] Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. 2020. Blank Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5186–5198.
[30] Inc. Uber Technologies. 2023. H3: A Hexagonal Hierarchical Spatial Index. https://h3geo.org/ Accessed: 2024-06-04.
[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[32] Xinglei Wang, Meng Fang, Zichao Zeng, and Tao Cheng. 2023. Where would i go next? large language models as human mobility predictors. *arXiv preprint arXiv:2308.15197* (2023).
[33] Yu Wang, Jun Feng, Zhicheng Liu, Xiang Wang, Tat-Seng Chua, and Xiangnan He. 2021. STGN: Spatio-Temporal Gated Network for Human Mobility Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[34] Jiayi Xie and Zhenzhong Chen. 2023. Hierarchical transformer with spatio-temporal context aggregation for next point-of-interest recommendation. *ACM Transactions on Information Systems* 42, 2 (2023), 1–30.
[35] Hao Xue, Flora Salim, Yongli Ren, and Nuria Oliver. 2021. MobTCast: Leveraging auxiliary trajectory forecasting for human mobility prediction. *Advances in Neural Information Processing Systems* 34 (2021), 30380–30391.
[36] Song Yang, Jiamou Liu, and Kaiqi Zhao. 2022. GETNext: trajectory flow map enhanced transformer for next POI recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on research and development in information retrieval*. 1144–1153.
[37] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).
[38] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering* 25, 1 (2011), 220–232.
[39] Kangzhi Zhao, Yong Zhang, Hongzhi Yin, Jin Wang, Kai Zheng, Xiaofang Zhou, and Chunxiao Xing. 2020. Discovering subsequence patterns for next POI recommendation.. In *IJCAI*, Vol. 2020. 3216–3222.
[40] Pengpeng Zhao, Anjing Luo, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. 2020. Where to go next: A spatio-temporal gated network for next poi recommendation. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2512–2524.
[41] Yu Zheng, Xing Xie, Wei-Ying Ma, et al. 2010. GeoLife: A collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* 33, 2 (2010), 32–39.
[42] Zihao Zhou, Xingyi Yang, Ryan Rossi, Handong Zhao, and Rose Yu. 2022. Neural point process for learning spatiotemporal event dynamics. In *Learning for Dynamics and Control Conference*. PMLR, 777–789.

A EXPERIMENTAL SETTINGS

For all experiments with TrajGPT, the following settings remain consistent for both GeoLife and MobilitySim: We implement TrajGPT in PyTorch and train it with an AMD EPYC 7V13 64-core CPU and an NVIDIA A100 80GB GPU. The number of scales for Space2Vec is 64. The largest scale of Space2Vec is set to the diameter of the region of interest, and the smallest scale is set to 1 meter. The dropout in the transformer is set to 0.1. The epsilon of layer normalization for the transformer is set to 1e-5. The learning rate is set to 1e-4. The patience for early stopping is set to 10 epochs. The random seed is set to 0.

We determine the rest of the hyperparameters of TrajGPT through grid search, using the validation loss as the selection criterion. For experiments on GeoLife, we set the number of layers for all transformer encoders to 2, the number of attention heads for all multi-head attention modules to 8, and the feedforward dimension to 32. GMM contains 3 components. Region and special token embeddings are each 32 dimensions. Each training batch contains 64 instances. For experiments on MobilitySim, we utilize 4-layer transformer encoders with a feedforward dimension of 256. The number of heads is 2 for all multi-head attention modules. GMM contains 5 components. Embedding size is 64. The batch size is 128.

For both GETNext and STAR-HiT, we employ the implementations from their respective repositories, which are linked to in their papers, and set all hyperparameters according to the papers.