sensors

MDPI

Article

Exploring Transformer and Graph Convolutional Networks for Human Mobility Modeling

Riccardo Corrias, Martin Gjoreski * and Marc Langheinrich

Computer Systems Institute, Faculty of Informatics, Università della Svizzera italiana (USI), 6900 Lugano, Switzerland; riccardo.corrias@usi.ch (R.C.); marc.langheinrich@usi.ch (M.L.)
* Correspondence: martin.gjoreski@usi.ch

Abstract: The estimation of human mobility patterns is essential for many components of developed societies, including the planning and management of urbanization, pollution, and disease spread. One important type of mobility estimator is the next-place predictors, which use previous mobility observations to anticipate an individual's subsequent location. So far, such predictors have not yet made use of the latest advancements in artificial intelligence methods, such as General Purpose Transformers (GPT) and Graph Convolutional Networks (GCNs), which have already achieved outstanding results in image analysis and natural language processing. This study explores the use of GPT- and GCN-based models for next-place prediction. We developed the models based on more general time series forecasting architectures and evaluated them using two sparse datasets (based on check-ins) and one dense dataset (based on continuous GPS data). The experiments showed that GPT-based models slightly outperformed the GCN-based models with a difference in accuracy of 1.0 to 3.2 percentage points (p.p.). Furthermore, Flashback-LSTM—a state-of-the-art model specifically designed for next-place prediction on sparse datasets—slightly outperformed the GPT-based and GCN-based models on the sparse datasets (1.0 to 3.5 p.p. difference in accuracy). However, all three approaches performed similarly on the dense dataset. Given that future use cases will likely involve dense datasets provided by GPS-enabled, always-connected devices (e.g., smartphones), the slight advantage of Flashback on the sparse datasets may become increasingly irrelevant. Given that the performance of the relatively unexplored GPT- and GCN-based solutions was on par with state-of-the-art mobility prediction models, we see a significant potential for them to soon surpass today's state-of-the-art approaches.

Keywords: machine learning; deep learning; graph convolutional networks; transformers; mobility modeling

Citation: Corrias, R.; Gjoreski, M.; Langheinrich, M. Exploring Transformer and Graph Convolutional Networks for Human Mobility Modeling. *Sensors* **2023**, *23*, 4803. https://doi.org/10.3390/s23104803

Academic Editor: Antonio Corradi

Received: 17 March 2023
Revised: 5 May 2023
Accepted: 15 May 2023
Published: 16 May 2023

1. Introduction

The estimation of human mobility patterns is widely recognized as being vital for the provision of future services and solutions [1]. Example application domains include digital solutions for more effective advertising [2], enhanced security [3], improved urban services [4], and tracking the spread of a contagious disease (e.g., COVID-19) [5]. Mobility modeling is also essential for behavioral tracking and interventions related to mental health and well-being [6,7], given that different symptoms (e.g., increased vs. decreased mobility and social interactions) can be connected to mental health problems, such as depression [8].

Mobility modeling is a well-established research domain. In the past, the focus has been on utilizing Markov models to analyze frequent mobility patterns [9–11]. Machine learning (ML) techniques—including Decision Trees [12], Support Vector Machines [13], and Random Forest [14]—are some of the ML algorithms that have been used in addition to Markov-based approaches to predict future mobility patterns. The first ML methods relied on features (e.g., time- and frequency-based descriptors) that were taken from the mobility trajectories [15]. The more recent ML methods replaced the more traditional feature- and

Markov-based approaches by utilizing raw trajectory data in combination with end-to-end learning (e.g., Recurrent Neural Networks—RNNs [16,17], and Long Short-term Memory networks—LSTMs [18]).

Unlike the existing mobility modeling methods based on RNNs and LSTMs, this study explores how two recent Deep Learning methods—General Purpose Transformer (GPT) and Graph Convolutional Networks (GCNs)—perform on next-place prediction tasks. GPT-based and GCN-based models have recently achieved outstanding results in image processing (e.g., diffusion-based models [19]) and natural language processing (e.g., ChatGPT [20]). Our reason for applying GPT-based models to mobility data is the similarity between sentences and movement trajectories. GPTs were initially applied in the NLP domain, where the words in each sentence are processed as input sequences. Similarly, locations that constitute the movement trajectories can be processed as "input words" to a GPT model. Additionally, the task of next-place prediction is quite similar to the task of next-word prediction, which is often used for the pre-training of Large Language Models (LLMs) [20]. On the other hand, the task of next-place prediction can be viewed as a graph problem where each node represents the places visited at a specific moment. By using a GCN-based approach to model this data, we try to model hidden interdependencies between the nodes. However, it is unclear how GPT- and GCN-based models will perform in the context of next-place prediction. Some specific challenges explored in this study are the model behavior on a small vs. a big dataset and the behavior on a sparse vs. a dense dataset. Our study contributions thus include the following:

- Developing GPT- and GCN-based architectures for next-place prediction.
- Adapting the popular Flashback-LSTM model for next-place prediction to our experimental setup to serve as a state-of-the-art baseline.
- Pre-processing three real-life datasets for mobility modeling (both sparse and dense) and performing a comparative analysis between GPT, GCN, and Flashback-LSTM by taking into account the behavior of the models with respect to the size of the datasets (e.g., small vs. big datasets) and the behavior of the models on a sparse vs. a dense dataset. The code for our experiments is publicly available (https://github.com/corrir/Transformers-and-Graph-Convolutional-Networks-for-Human-Mobility-Modeling (accessed on 16 March 2023)).

The rest of this paper is structured as follows. Section 2 summarizes the relevant related work on human mobility models. Section 3 describes the three datasets used in the study, including the pre-processing steps for each dataset. Section 4 details the three methods used in our experiment (a GPT-based model, a GCN-based model, and Flashback-LSTM). The experimental setup and results are presented in Section 5. We discuss our findings and conclude the study in Section 6.

2. Related Work

When reviewing the state-of-the-art deep learning models for mobility tasks, Luca et al. [21] proposed a two-level taxonomy (see Figure 1). At the first level, mobility tasks are divided into *generative tasks* and *predictive tasks*. At the second level, the *generative* tasks are split into *flow generation* and *trajectory generation*, while the *predictive* tasks are split into *crowd-flow prediction* and *next-location prediction*. Next-location prediction focuses on estimating an individual's future movements given the historical movement of that individual. We present a more detailed analysis of this task in the following subsection. On the other hand, crowd flow prediction estimates aggregated traffic flows [22] and the demand for shared transportation tools (e.g., city parking [23] and bike-sharing [24]).

Next-Place Prediction

Human mobility models aim to predict future places visited by individuals or groups. Next-place predictors focus on individual predictions and use previous observations to anticipate an individual's future location. Next-place prediction can be helpful for various purposes, such as monitoring public health [25], well-being [26], as well as improving

travel recommendations, geomarketing, and social network link predictions [27]. These predictions can also be used by authorities to enhance public transportation, urban planners to plan for the future growth of a city, and transportation companies to provide better traffic management services.

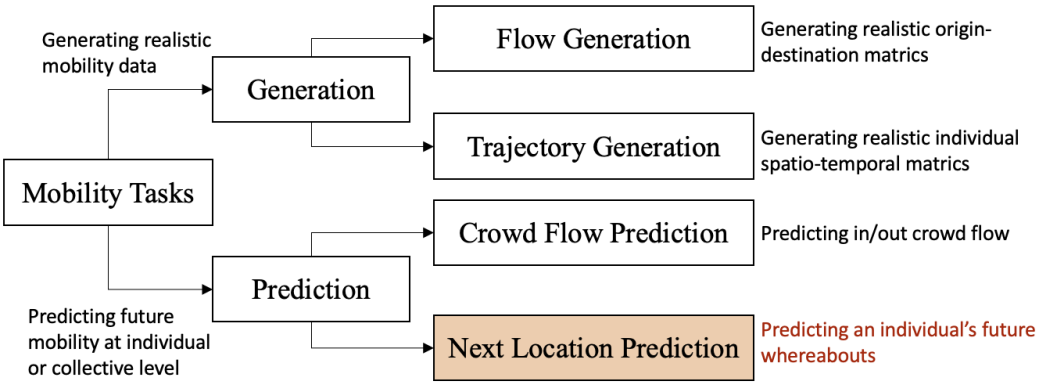

Figure 1. Mobility-task taxonomy. Based on the work by Luca et al. [21].

Next-place prediction is a complex task, because it requires the understanding of an individual's unique routine and the consideration of various factors influencing human mobility. Luca et al. [21] defined it as follows:

> "Next-location predictors forecast an individual's future whereabouts, based on their historical observations... Formally, let u be a user, T_u their trajectory, and $p_t \in T_u$ u's current location, next-location prediction aims at predicting u's next destination p_{t+1}. This problem is treated in two ways: (i) as a multiclass classification task, in which we have as many classes as locations and we aim at predicting the next visited location p_{t+1}; or (ii) as a regression task, predicting $p_{t+1} = (x_{t+1}, y_{t+1})$, where x_{t+1} and y_{t+1} are the next location's geographic coordinates. A variant of next-location prediction aims at forecasting the next Point Of Interest (POI) p_{t+1} an individual u will visit given their trajectory T_u. Regardless of the specific definition, next-location predictors output a ranking of the probability of each location to be u's next destination."

The initial methods for next-place prediction were based on probability or patterns. These methods could handle datasets of moderate size [28] but required significant effort in feature engineering and could not capture long-term temporal and geographical correlations. One example of a probability-based method combined multiple features of trajectories, such as the spatiotemporal, semantic characteristics of the modeled places and the distance traveled [29]. Other methods use stochastic models to cluster geographical coordinates into relevant areas [9], or create a Mobility Markov Chain in which each Markov state represents a modeled place, and the transitions between the Markov states correspond to movements between POIs [30]. Another example, the *Wherenext* system [12], uses patterns to characterize mobility as a sequence of frequently visited places, considering the duration of the journey.

More recent approaches utilize deep learning techniques to overcome traditional methods' limitations by capturing temporal, geographical, and social–geographic patterns in large datasets. These approaches use mechanisms such as RNNs, LSTM, gated recurrent units (GRUs), fully connected layers (FCs), attention mechanisms, and Convolutional Neural Networks (CNNs). For example, DeepMove [16], RNN+SAtl [17], and Flashback [18], are deep learning architectures that learn time and location embeddings and apply RNNs, including LSTMs, to predict the future places that will be visited by a user, i.e., next-place prediction.

Unlike the existing mobility modeling methods based on RNNs and LSTMs, this study explores GPTs and GCNs for next-place prediction. A recent review study by Luca et al. [21] analyzed the existing deep learning approaches for human mobility modeling. Their

analysis showed that, out of the 240 studies, only three were related to GPT- and GCN-based modeling. From those three studies, the MVGCN (Multiview Graph Convolutional Network) [31] and SI-GCN (Spatial Interaction GCN) [32] are two GCN-based networks used for flow generation. Given that flow generation is a task for estimating the flows between a group of geographic places based on those locations' data (such as population, POIs, land use, and distance to other sites), it is a separate task from next-place prediction. Finally, only one study presented a transformer-like architecture, the DWSTTN (Deep Wide Spatio Temporal Transformer Network) [33]. The DWSTTN uses historical pick-up and drop-off data from taxi companies in Porto and Manhattan to predict a taxi's next destination. The scarcity of GPT- and GCN-based modeling in the related work is not surprising, given that these approaches are relatively new. Thus, to the best of our knowledge, we present the first study to compare GPT-based and GCN-based models for next-place prediction.

3. Datasets

Mobility datasets can be collected using various methods, including GPS tracking, mobile phone data, or surveys. The data can be collected at different levels of granularity, ranging from the detailed tracking of individual movements to more aggregated data on the movements of groups or populations. In this study, we worked with three datasets: *Foursquare*, *Gowalla*, and *Breadcrumbs*. All three datasets provide the following information:

- User ID: This feature is a unique identifier for each individual or entity represented in the dataset.
- POI ID: This feature is a unique identifier for each Point of Interest (POI) represented in the dataset.
- Latitude: This feature represents a value to identify the position of the POI. More particularly, it represents the North–South position (vertical).
- Longitude: This feature represents a value to identify the position of the POI. More particularly, it represents the East–West position (horizontal).
- Timestamp: This feature answers the question of *when* a specific user is in a certain POI.

3.1. Foursquare Dataset

Foursquare was a company that provided location-based services and data through its mobile and web applications. The *Foursquare* dataset [34,35] includes data on the locations of millions of businesses, venues, and points of interest around the world, as well as data on the movements and activities of users who have checked in at these locations using the Foursquare services. This dataset contains Foursquare check-in information gathered over an extended period (about 18 months, from April 2012 to September 2013). It includes check-ins totaling 33,278,683 from 266,909 people in 3,680,126 locations in 415 cities across 77 countries. These 415 cities have at least 10,000 check-ins apiece, making them the 415 most frequented towns across the globe. Figure 2 depicts the distribution of user locations available in the dataset.

3.2. Gowalla Dataset

Similar to the Foursquare dataset, the *Gowalla* dataset was extracted from another location-based service (Gowalla). From February 2009 to October 2010, 6,442,890 check-ins from Gowalla users were gathered using the service's API [36]. Compared to the *Foursquare* dataset, *Gowalla* has a smaller number of check-ins.

3.3. Breadcrumbs Dataset

The *Breadcrumbs* dataset was collected in a 12-week study by Moro et al. [37] from 81 users in Lausanne, Switzerland using several mobile phone sensors (GPS, WiFi, Bluetooth). Compared with the other two datasets (Foursquare and Gowalla), Breadcrumbs is based on continuous mobile sensing (e.g., 1 Hz GPS data), which is different than the sparse, check-in-based data available in Foursquare and Gowalla.

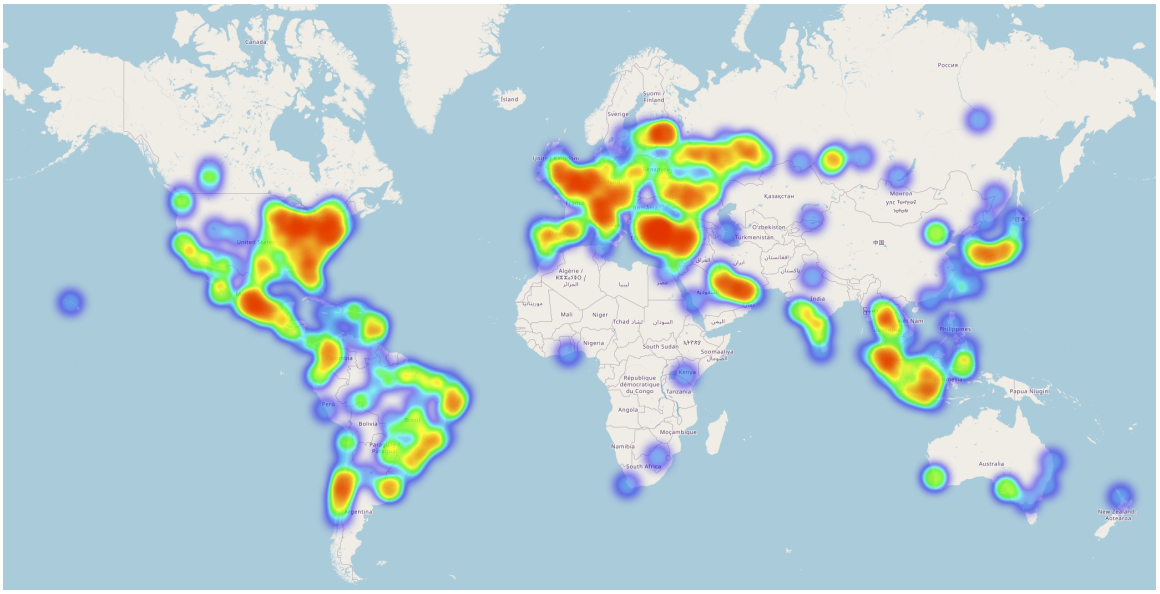

Figure 2. Distribution of user locations available in the Foursquare dataset. Red represents higher density and blue represents lower density.

In our earlier study on mobility modeling, we used the hierarchical spatiotemporal DBSCAN, a clustering algorithm, to determine POIs in the Breadcrumbs dataset, i.e., the shared locations where users spend the majority of their time [38]. The algorithm employed the following parameters: a maximum cluster radius of 250 m and a waiting time of 5 min (stop-points with a shorter waiting time were ignored). After POI extraction, we subsampled the dataset at a $15-min rate (i.e., we would get check-in information for each user every 15 min) to make it more comparable to the check-in-based datasets. This led to 15,898 check-ins from 81 users in 40 different places.

3.4. Pre-Processing

We performed several pre-processing steps for each of the three datasets, *Foursquare*, *Gowalla*, and *Breadcrumb*, in order to bring them to a similar data format.

- **Cleaning and standardization**:
  We noticed that not all timestamps followed the same format. Since there can be a variety of Timestamp (Date), we decided to transform all Timestamp fields into the following format:

$$yyyy - MM - ddTHH : mm : ssZ$$

$$2013 - 03 - 12T20 : 34 : 32Z$$

  Another step done for data cleaning was the elimination of lower-quality data. Following the related work [18], we decided to remove all the users with less than 100 check-ins and POIs that were visited less than 50 times from *Foursquare* and *Gowalla*. The last step regarding data transformation was standardizing all the *userIDs*, i.e., we mapped all string-based *userIDs* into unique integers.
- **Additional information**: To give the models more information for each check-in and improve the predictions' accuracy, we added a feature to identify the day of the week on which a specific check-in was registered.

4. Machine Learning Methods

4.1. Flashback-LSTM

Flashback-LSTM was introduced by Yang et al. [18] to model sparse user mobility data by exploiting flashbacks on hidden states in RNNs. Sparse user mobility is an important challenge for check-in-based datasets, given that vanilla RNNs have been created to learn

from continuous and equidistant sequences (e.g., sentences), and hence cannot handle sparse data easily. To address the challenge, Flashback uses the spatiotemporal context by searching for previous hidden states with strong predictive potential similar to the current environment, i.e., the instance to be predicted, given a user's mobility trace represented as a sequence of POIs:

$$\{\cdots, P_{i-3}, P_{i-2}, P_{i-1}, P_i, P_{i+1}, \cdots\}$$

The temporal and spatial difference between two check-ins, P_i and P_j, is denoted as $\Delta T_{i,j}$ and $\Delta D_{i,j}$. To predict the subsequent position p_{i+1} Flashback LSTM leverages the spatiotemporal context by computing the weighted average of the historical hidden states with the weight $W(\Delta T_{i,j}, \Delta D_{i,j})$ as an aggregated hidden state to seek prior hidden states rather than relying on the present hidden state h_i to predict the subsequent position p_{i+1}. Flashback-LSTM also learns user-specific embedding vectors to represent their preferences. This vector is then concatenated with the combined hidden state and put into a fully connected layer for location prediction.

In our experiments, Flashback-LSTM served as a state-of-the-art baseline model. For the training procedure, we used the Cross-Entropy Loss Function in order to adjust the model weights. As an optimization algorithm, we used Adam, and the learning rate used throughout the experiments was dynamically adjusted, starting from a value of 0.01. Every 20 epochs, a gamma value of 0.2 was applied to achieve better convergence to the prediction. The whole training phase was done for 100 epochs.

4.2. Multivariate Time Series Graph Neural Network (MTGCN)

Multivariate time series forecasting can be viewed as a graph problem, where in our specific task of next-place prediction, the variables in the time series are represented as nodes. In other words, each node represented the POIs visited at a specific moment. The assumption is that the nodes are connected through hidden interdependencies. By using graph neural networks to model this data, it is possible to preserve the temporal trajectory of the time series while also leveraging the interdependencies between the variables.

Our GCN model was based on the Multivariate Time Series Graph Neural Network (MTGNN) proposed by Wu et al. [39]. GNNs are particularly adept at handling dependencies between entities. However, GNNs need a clear graph structure for information propagation to perform effectively. This means they cannot be used directly on multivariate time series data, where the dependencies between variables are not known in advance. The suggested method automatically extracts the unidirected relationships between variables through a graph learning module. External knowledge, such as variable qualities, may be included in the graph learning module. A mix-hop propagation layer and a dilated inception layer are suggested as two new layers to capture spatial and temporal relationships in the time series data. These layers are collectively taught inside an end-to-end architecture, together with the graph learning, graph convolution, and temporal convolution modules.

When using the MTGNN (Figure 3) for next-location prediction, the sequence of visited POIs is initially projected onto a latent space using a 1×1 standard convolution. To capture geographical and temporal relationships, graph convolution modules and temporal convolution modules are interleaved with one another. The hidden graph adjacency matrix, which is used by the graph convolution modules, is learned by the graph learning layer through the use of similarity methods between a portion of the different check-ins for each user in each minibatch. Finally, to obtain the predicted next location, the output module projects hidden characteristics to the specified dimension.

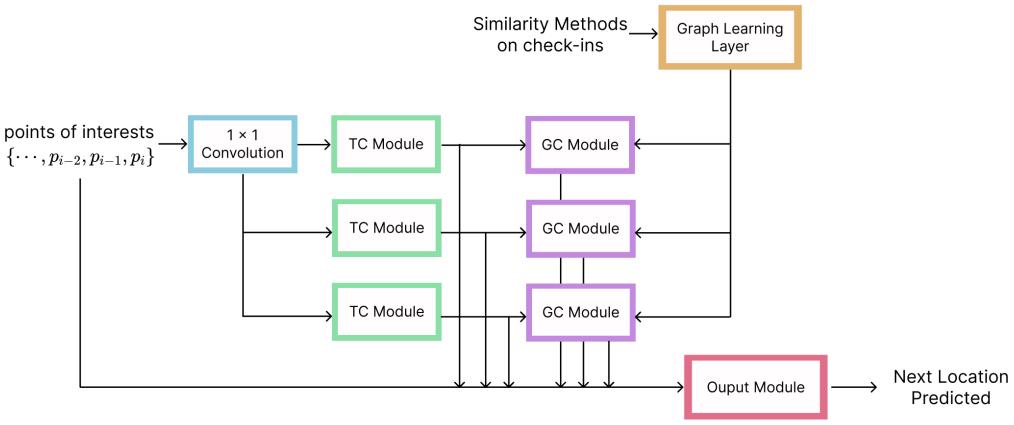

Figure 3. Architecture overview. Multivariate Time Series Graph Neural Network for next-POI prediction.

More precisely, the problem can be formulated as follows:

Let $z_t \in \mathbb{R}^n$ denote the value of a multivariate variable of dimension N at time step t, where $z_t[i] \in \mathbb{R}$ denotes the value of the ith variable at time step t. Given a sequence of historical P time steps of observations on a multivariate variable, we have

$$X = \{z_{t_1}, z_{t_2}, \cdots, z_{t_P}\}$$

The goal is to predict the value:

$$Y = z_{t_{P+1}}$$

The input signals (e.g., historical check-ins) can be combined with other information, such as the day of the week and the time of day. We view variables in multivariate time series as nodes in graphs from a graph-based approach. The graph adjacency matrix, which is generated by the Graph Learning Layer in MTGNN, defines the connections between the nodes. The adjacency matrix may be defined as follows:

$$A \in \mathbb{R}^{N \times N}$$

with:

$$A_{ij} = c > 0 \text{ if } (v_i, v_j) \text{ are connected by an edge}$$
$$A_{ij} = 0 \text{ if } (v_i, v_j) \text{ are not connected by an edge}$$

A Graph Learning Layer, a Graph Convolution Module, Temporal Convolution Module, and an Output Module are the particular parts that comprise the MTGNN. The Graph Learning Layer computes a graph adjacency matrix, which is subsequently utilized as input for the Graph Convolution Module to find hidden relationships between the nodes. These modules are interleaved with the Temporal Convolution Module, which captures temporal relationships. To have a better and clear understanding of how the MTGNN works, we can go through the different layers and modules:

- **Graph Learning Layer**: Different from many Graph Neural Networks that can only operate with a known graph structure (which is not the case with time series forecasting because the structure is unknown before the training procedure), the MTGNN uses this layer to extract a sparse graph adjacency matrix depending on the data. Previous research has measured the similarity between pairs of nodes using distance metrics, such as the dot product and Euclidean distance, in order to construct a network from data. However, the cost of this method increases quadratically with the size of the network and can be computation and memory heavy, with a time and space complexity of $O(N^2)$. This restriction may be overcome using a sampling strategy, whereby only the connections between a portion of the nodes in each minibatch are learned.

- **Graph Convolution Module**: This module aims to describe spatial relationships in a network by incorporating knowledge from a node's neighbors. It comprises two mix-hop propagation layers that handle the entrance and outflow of data independently through each node. The outputs of these two levels are combined to provide the net inflow information. The Graph Convolution Module aims to connect a node's information with its neighbors. In our case, since the nodes represent the POIs visited at that specific moment in time, and the neighbors of each node describe the next location visited starting from the previous node, this fuse operation should improve the understanding of the model on how the locations are distributed and the general trajectory of the users.
- **Temporal Convolution Module**: To extract high-level temporal information, this module uses a collection of common dilated 1D convolution filters. There are two dilated inception layers in this module. A filtering tangent hyperbolic activation function follows the first dilated inception layer. The second dilated inception layer is followed by a sigmoid activation function, which serves as a gate to regulate how much data the filter may propagate to the subsequent module.
- **Output Module**: Composed by two convolution layers, transforming the channel dimension of the inputs to the desired output dimension, which for next location prediction, corresponds to setting the desired output dimension to one.

We used the same training procedure that was used for training the Flashback-LSTM models, i.e., the modes were trained for 100 epochs using the Cross-Entropy Loss Function. The Adam optimizer was used with a dynamic learning rate (modified every 20 epochs).

4.3. Temporal Fusion Transformer (TFT)

Our GPT model was based on the Temporal Fusion Transformer (TFT) architecture proposed by Lim et al. [40]—initially developed for multihorizon time series forecasting. The architecture includes a sequence-to-sequence layer to process known and observed inputs locally, static covariate encoders to encode context vectors for use elsewhere in the network, sample-dependent variable selection to reduce the impact of irrelevant inputs, and a temporal self-attention decoder to learn any long-term dependencies present in the dataset.

A high-level overview of the TFT architecture for next-POI prediction is presented in Figure 4. There are three types of inputs for the model, Static Metadata, Past Inputs (which correspond to previous POIs), and Known Future Inputs (that can be extracted from the dataset in order to help with the predictions). The Known Future Inputs can be the day of the week and the time of the day.

Other components included in the TFT architecture are the following:

- **Gating Mechanism**: Used to skip over any parts of the architecture that are not being used by the model, offering adjustable depth and network complexity to support a variety of datasets and scenarios. The gating mechanism is implemented using a Gated Residual Network.
- **Variable Selection Networks**: TFT is a model that can manage several variables, but it is frequently uncertain what each variable specifically contributes to the result and how relevant it is. TFT uses Variable Selection Networks to select pertinent variables for static and time-dependent covariates individually for each occurrence. In addition to assisting with the identification of factors that are most important for the prediction task, this enables TFT to exclude irrelevant inputs that could have a negative effect on performance. This component is based on the Gated Residual Network and an external context vector obtained from a Static Covariate Encoder.
- **Static Covariate Encoder**: The TFT uses context vectors to encode static metadata. The vectors are produced by encoders based on Graph Recurrent Networks. These context vectors are a crucial component of the temporal fusion decoder's processing of static variables.

- **Multihead Attention**: The TFT uses a self-attention mechanism, which is adapted from multihead attention in transformer-based architectures to learn long-term associations across various time steps. This component learns attention weights to estimate the significance of the different inputs.
- **Temporal Fusion Decoder**: This component learns the temporal connections contained in the dataset through a succession of layers.
  - *Locality Enhancement with the Sequence-to-Sequence Layer*: This layer aims to estimate informative segments (e.g., check-ins) in the time series data. Depending on the data, informative data points can be abnormalities, change points, or repetitive patterns.
  - *Static Enrichment Layer*: Because static covariates often substantially impact temporal dynamics, this layer incorporates static metadata and enhances temporal features.
  - *Temporal Self-Attention Layer*: After static enrichment, self-attention is applied, followed by a multihead attention mechanism at each time step. Decoder masking is then added to the multi-head attention layer to guarantee that each temporal dimension focuses on features that are in the past (and not in the future). The self-attention layer enables TFT to recognize long-range relationships that would be difficult for RNN-based systems to learn while maintaining causal information flow via masking.
  - *Position-Wise Feed-Forward Layer*: Applies a nonlinearity to the temporal self-attention layer's outputs. Additionally, it contains a gated residual connection that provides a direct connection to the sequence-to-sequence layer by skipping over the full transformer block.
- **Output layer**: In order to generate the next-location predictions, we used a modified version of quantile forecasts. The outputs were generated using linear transformation of the output from the temporal fusion decoder.

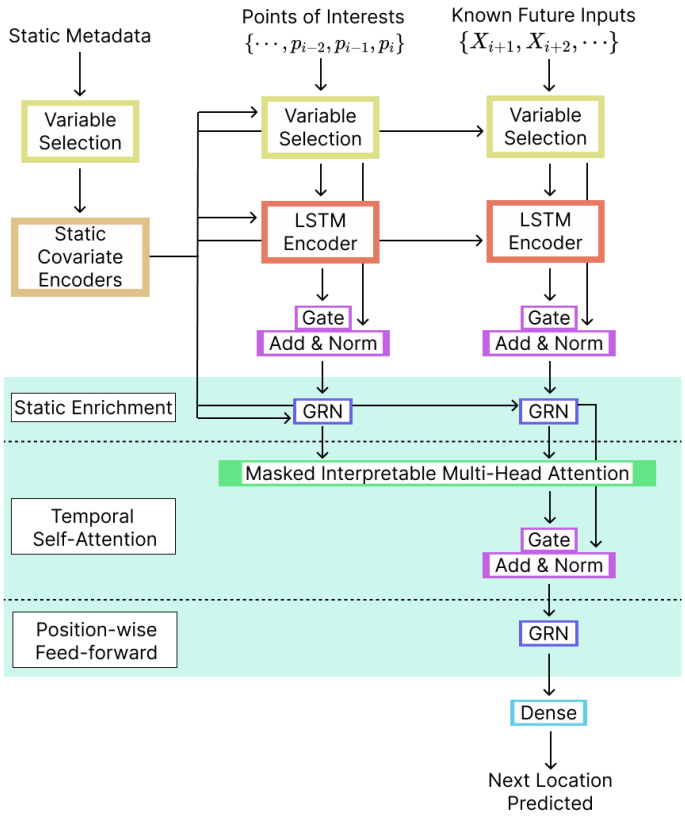

Figure 4. Architecture overview. Temporal Fusion Transformer for next-POI prediction.

We used the same training procedure that was used for training the other model types, i.e., the modes were trained for 100 epochs using the Cross-Entropy Loss Function. The Adam optimizer was used with a dynamic learning rate (modified every 20 epochs).

5. Experiments

5.1. Experimental Setup

5.1.1. Hardware

All experiments were performed on the Ubuntu 22.04.1 server equipped with two GPU units, the NVIDIA GeForce RTX 3080 with 10 GB of memory, and the NVIDIA RTX A5000 with 24 GB of memory.

5.1.2. Dataset Splits

It is important to note that since we were working with spatiotemporal data, we could not randomly shuffle the datasets. Otherwise, the models would be trained with events that are happening in the future. Consequently, we first sorted each dataset based on the timestamps, and then we created the following three splits:

- *Training set*: Used to train the predictive models and corresponds to the first 70% of the sorted datasets.
- *Test set*: Used to perform final model evaluation. This subset corresponds to the final 10% of the sorted datasets.
- *Validation set*: Used to validate the predictive models during training. This subset corresponds to 20% of the overall data. In temporal terms, 20% of the data were selected from the period between the training set and the testing set, i.e., from 70% to 90%.

In addition to the typical training/testing/validation splits, we also performed experiments with several subsets extracted from the original datasets. Both datasets (especially the Foursquare dataset) cover diverse populations (e.g., from Figure 2 it can be seen that Foursquare users come from all around the world). To investigate the behavior of the models for the different population-based subgroups, we first split the datasets into population-based datasets (e.g., based on users' country, city, gender, and number of Twitter followers) and then split the resulting subdatasets into training/testing/validation datasets using the initial procedure (70%–10%–20% splits).

Section 5.2 presents the results for the overall datasets. Section 5.3 presents the results for the Foursquare subdataset, and Section 5.4 presents the results for the Gowalla subdatasets.

5.1.3. Model Evaluation

For all models, the number of training epochs was set to 100 epochs. The final models were evaluated using the top-k Accuracy score (ACC@k). The ACC@k represents the ratio of the number of times the correct location was among the top-k output locations (ordered from most likely to least likely) divided by the total number of predictions. For example, for the ACC@3 metric, the target (correct location) is checked against a vector of the top 3 most likely output locations. The prediction is accurate (or a true positive) if the target is one of the top 3 vector elements. In the end, the total number of predictions is divided by the number of true positives.

5.2. Results—Full Datasets

Table 1 presents the results for next-place prediction for the three models and for each of the three datasets. The table shows that regardless of the type of model, the accuracy scores highly depend on the dataset. On the Breadcrumbs dataset, all models achieved an accuracy of higher than 0.94 (e.g., ACC@1). However, for the Foursquare and Gowalla datasets, the ACC@1 was lower than 0.27. These results are in line with related work [18,41]. Data sparsity is the main reason for the difference in accuracy scores between the sparse datasets (Foursquare and Gowalla) and the dense dataset (Breadcrumbs). This is because the Breadcrumbs dataset has a 15-min sampling frequency, i.e., the models provide a

prediction for every 15-min slot, and there is a high probability that the users are at the same place. On the other hand, Foursquare and Gowalla are check-in-based datasets, and the users are rarely at the sample place twice in a row.

Table 1. Top-k Accuracy scores (k = 1, 5, and 10) for the three models for each of the three datasets (Foursquare, Gowalla, and Breadcrumbs).

	Flashback-LSTM			**MTGCN**			**TFT**		
	ACC@1	**ACC@5**	**ACC@10**	**ACC@1**	**ACC@5**	**ACC@10**	**ACC@1**	**ACC@5**	**ACC@10**
Foursquare	0.2678	0.5536	0.6326	0.2356	0.4928	0.6251	0.2577	0.5389	0.6212
Gowalla	0.1281	0.2935	0.3637	0.0932	0.2628	0.3421	0.1076	0.271	0.3572
Breadcrumbs	0.9477	0.9839	0.9931	0.9442	0.9786	0.992	0.947	0.9821	0.9928

Regarding the accuracy score for each model, the Flashback-LSTM model—a state-of-the-art model from the related work specifically designed for next-place prediction on sparse datasets [18]—slightly outperformed the MTGCN and the TFT models. More specifically, on the Foursquare dataset, Flashback-LSTM achieved 0.2678 ACC@1, MTGCN achieved 0.2356 ACC@1, and TFT achieved 0.2577 ACC@1. Thus, the difference was less than 0.0322 (or 3.22 percentage points—p.p.). On the Gowalla dataset, the difference was less than 3.49 p.p. On the Breadcrumbs dataset, all models performed similarly, achieving an ACC@1 score of 0.94.

To provide more insight into the training process of the models, we present the learning curves for the three models on the Foursquare dataset: Figure 5 shows the Flashback-LSTM model; Figure 6 shows the MTGCN model; and Figure 7 shows the TFT model. From the figures, it can be seen that (i) all models have similar learning patterns, i.e., the loss values (categorical cross-entropy) decreased significantly in the first twenty learning epochs. After that, the loss values decreased steadily at a lower rate; (ii) there were no abrupt variations in the learning curves, which signifies a stable learning process and consequently stable models; and (iii) the training loss values were close to the validation loss values throughout the overall training process, which signifies that all models avoided overtraining.

5.3. Results—Foursquare Subdatasets

We performed additional experiments on the Foursquare dataset to evaluate the models more thoroughly. The Foursquare dataset contains user profile data in addition to check-in data. Using the user profile information, we created seven subdatasets based on the users' country, city, gender, and number of Twitter followers. Information about the size of the subdatasets is presented in Table 2:

1. *_US*: Data from US-based users.
2. *_NYC*: Data from New York City users.
3. *_TKY*: Data from Tokyo users.
4. *_NYC_gender*: Data from male users from New York City. We chose males because the number of male users was higher than the number of female users.
5. *_TKY_gender*: Data from male users from Tokyo. We chose males because the number of male users was higher than the number of female users.
6. *_NYC_gender*: Data from the top 1000 New York City users ranked by the number of Twitter followers.
7. *_TKY_gender*: Data from the top 1000 Tokyo users ranked by the number of Twitter followers.

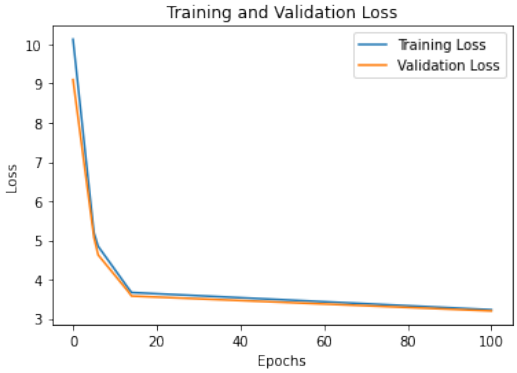

Figure 5. Flashback-LSTM model loss curves. Foursquare dataset.

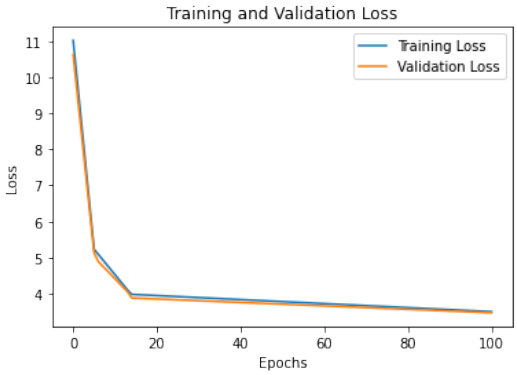

Figure 6. MTGCN model loss curves. Foursquare dataset.

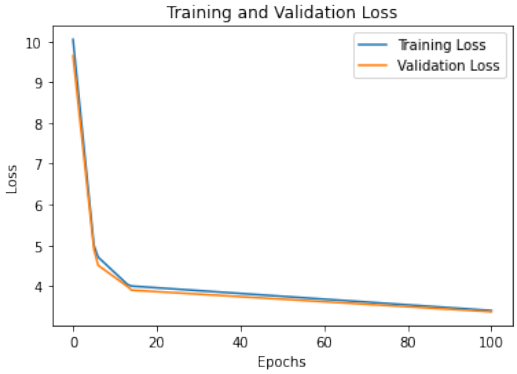

Figure 7. TFT model loss curves. Foursquare dataset.

Table 2. Size of the Foursquare dataset and the corresponding subdatasets based on the city, gender, and number of Twitter followers.

Info	4sq	_US	_NYC	_TKY	_NYC_gender	_TKY_gender	_NYC_twitt	_TKY_twitt
Checkins	8,180,051	678,663	281,374	802,344	196,223	45,510	49,803	217,381
Users	65,340	11,116	4214	4059	2876	480	1000	1000
POIs	61,794	7183	2956	4180	2104	392	545	1362

These subdatasets enabled us to evaluate the behavior of the models with a varying number of users (from 480 up to 65,340), a varying number of POIs (from 392 up to 61,794), and a varying number of training instances/check-ins (from 45,510 up to 8,180,051).

The results of the experiments are presented in Table 3. From the table, it can be seen that the results follow a similar trend as the results from the previous subsections, i.e., the full dataset experiments (Table 1): (i) Regardless of the type of model, the accuracy scores depend strongly on the dataset; (ii) the Flashback-LSTM model slightly outperformed

the MTGCN and the TFT models. More specifically, if we look at ACC@1 scores, the difference between Flashback-LSTM and TFT was 0 to 2.5 p.p, and the difference between Flashback-LSTM and MTGCN was between 1 and 3.5 p.p.; and (iii) the TFT model slightly outperformed the MTGCN in all experiments.

Table 3. Top-k Accuracy scores (k = 1, 5, and 10) for the three models on seven Foursquare subdatasets based on the city, gender, and number of Twitter followers.

	Flashback LSTM			**MTGCN**			**TFT**		
	ACC@1	**ACC@5**	**ACC@10**	**ACC@1**	**ACC@5**	**ACC@10**	**ACC@1**	**ACC@5**	**ACC@10**
4sq	0.2678	0.5536	0.6326	0.2356	0.4928	0.6251	0.2577	0.5389	0.6212
4sq_US	0.3946	0.8016	0.8723	0.3531	0.7884	0.861	0.3756	0.7692	0.8711
4sq_NYC	0.4031	0.8112	0.8802	0.3588	0.7979	0.8752	0.3841	0.8013	0.8749
4sq_NYC_gender	0.4065	0.8304	0.9062	0.361	0.8072	0.8824	0.3858	0.8136	0.8909
4sq_NYC_tweet	0.4495	0.855	0.9385	0.4139	0.8221	0.9264	0.4243	0.8348	0.9215
4sq_TKY	0.2245	0.5301	0.6284	0.2062	0.5059	0.5928	0.2228	0.5284	0.6193
4sq_TKY_gender	0.3483	0.7378	0.8521	0.3229	0.7091	0.8133	0.3401	0.7226	0.8432
4sq_TKY_tweet	0.2589	0.6032	0.7089	0.2483	0.5812	0.6817	0.2498	0.5941	0.6916

5.4. Results—Gowalla Subdatasets

With the Gowalla dataset, we were not able to create diverse subdatasets (as we did with the Foursquare subdatasets), because Gowalla does not contain user profile data, except for the number of friends for each user. Thus, we created a Gowalla subdataset by taking the top 1000 users ranked by their number of friends. The resulting subdataset (*Gowalla_firends*) is presented in Table 4.

The results of the experiments are presented in Table 5. On the *Gowalla_firends* subdataset, Flashback-LSTM achieved the highest ACC@1 score of 0.4, whereas both the MTGCN and the TFT model achieved an ACC@1 score of 0.36.

Table 4. Size of the Gowalla dataset and the corresponding subdataset based on the number of friends each user has.

Info	**Gowalla**	**Gowalla_friends**
Checkins	6,442,890	196,591
Users	196,591	1000
POIs	35,639	504

Table 5. Top-k Accuracy scores (k = 1, 5, and 10) for the three models on the Gowalla subdataset.

	Flashback LSTM			**MTGCN**			**TFT**		
	ACC@1	**ACC@5**	**ACC@10**	**ACC@1**	**ACC@5**	**ACC@10**	**ACC@1**	**ACC@5**	**ACC@10**
Gowalla	0.1281	0.2935	0.3637	0.0932	0.2628	0.3421	0.1076	0.271	0.3572
Gowalla_friends	0.4022	0.7966	0.871	0.3597	0.6923	0.8015	0.3616	0.7052	0.8124

6. Discussion and Conclusions

6.1. Discussion of the Results

We presented a study that explored deep learning models based on GPT and GCNs for mobility modeling. More specifically, we developed models for next-place prediction, and we evaluated them using two sparse datasets (Foursquare and Gowalla) and one dense dataset (Breadcrumbs). Including the eight subdatasets we created from the Foursquare and the Gowalla datasets, we performed experiments on eleven different datasets. Some general findings that were confirmed in all eleven experiments are the following:

- Regardless of the type of model, the accuracy scores were highly dependent on the dataset. Data sparsity was the main reason for the difference in accuracy scores between the sparse datasets (Foursquare and Gowalla) and the dense dataset (Breadcrumbs).
- For the sparse datasets, Flashback-LSTM—specifically designed for next-place prediction on sparse datasets—slightly outperformed the GPT-based and GCN-based models. This indicates that some of the mechanisms that are specific to the Flashback-LSTM could be useful to improve the quality of the other models. For instance, Flashback-LSTM explicitly takes advantage of spatiotemporal context in the RNN layers to seek past hidden states with good predictive potential. By performing "flashbacks" on the RNN's hidden states, the model may assess how relevant the current input is in light of analogous prior scenarios.
- For the dense dataset, all the models achieved equal evaluation scores (see Table 1).
- The TFT model slightly outperformed the MTGCN in all experiments.

The learning curves of the three models revealed that all the models have similar, stable learning processes, and all models avoided overtraining, i.e., there were no substantial differences between training and the validation learning curves of the models.

The experimental results showed that GPT- and GCN-based architectures are a promising direction for future mobility modeling, especially given that future use cases will likely involve dense datasets provided by ubiquitous sensing devices.

6.2. Data Privacy

The methods presented in this study are fully centralized, i.e., we assume that the data from all users are available at one data center. However, such a centralized approach raises serious privacy concerns, given the sensitivity of mobility data. Since location and movement data are included in the definition of personal data, the EU General Data Protection Regulation (GDPR) has highlighted the significance of this information [42]. In the real world, there are currently no generally applicable methods for service providers to track users without seriously jeopardizing their right to privacy (see, for example, [1,43,44]). Promising privacy-aware approaches that can be explored in combination with GPT and GCN models include

- The use of federated learning (FL) instead of centralized learning for training the predictive models. FL allows devices to learn a shared model collaboratively while keeping all the training data on-device [45]. While offering promising privacy-aware solutions, FL is a relatively new approach with many open challenges, including scalability and federated model training and evaluation [41].
- The use of cohort-based modeling [38], whereby, instead of estimating the behavior of each user separately, cohort-based models aim to estimate the behavior of a group of similar users (cohorts). Thus, the specific users are removed from the focus of the processing pipelines. Cohort-based models treat each user's data as just one data point that is anonymously aggregated within the pool of cohort data. Such an approach provides k-anonymity by default (k is the size of the cohort) [46], enabling privacy-aware analytics.

6.3. Limitations and Future Work

State-of-the-art models, including GPT and GCN models, rely on large datasets to learn patterns and make predictions. These datasets can be biased or incomplete. For instance, the model may be less accurate in predicting outcomes for some groups if the dataset used to train the model does not include data for some specific regions or demographic categories. Additionally, factors, such as one's economic situation and educational background, may also need to be considered when estimating mobility patterns, and it is unclear how models such as GPT and GCN models would be able to include such information. Furthermore, GPT and GCN models are complex and challenging to interpret, which makes it difficult for policymakers and stakeholders to evaluate and validate the predictions made by the model. Thus, Explainable AI techniques may be required to make these approaches more

applicable (e.g., [47]). In addition to transparency and privacy, fairness is a third dimension that needs to be considered to avoid discriminatory outcomes. Finally, to capture the dynamic nature of human mobility, such models will need to be updated frequently to maintain their predictive power.

In this study, we did not perform a thorough hyperparameter optimization, which could be a promising direction for improving the evaluation scores of the GPT- and GCN-based models. Note that the Flashback architecture was optimized to work with sparse datasets in the original study. Given the large number of inter-related hyperparameters and the overall complexity of the DL architectures, a hyperparameter search based on Bayesian optimization may be an appropriate approach to solve this problem [48].

Author Contributions: Conceptualization, R.C., M.G. and M.L.; Funding acquisition, M.L.; Investigation, M.G.; Methodology, R.C. and M.G.; Project administration, M.L.; Resources, M.L.; Software, R.C.; Supervision, M.G. and M.L.; Validation, R.C.; Visualization, R.C.; Writing—original draft, R.C.; Writing—review & editing, M.G. and M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Swiss National Science Foundation, project 200021-182109—BASE: Behavioral Analytics for Smart Environments.

Data Availability Statement: The study used publicly available datasets. The details are presented in the section Datasets.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baron, B.; Musolesi, M. Where you go matters: A study on the privacy implications of continuous location tracking. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2020**, *4*, 1–32. [CrossRef]
2. Chen, G.; Cox, J.H.; Uluagac, A.S.; Copeland, J.A. In-depth survey of digital advertising technologies. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2124–2148. [CrossRef]
3. Crossler, R.E.; Johnston, A.C.; Lowry, P.B.; Hu, Q.; Warkentin, M.; Baskerville, R. Future directions for behavioral information security research. *Comput. Secur.* **2013**, *32*, 90–101. [CrossRef]
4. Rathore, M.M.; Ahmad, A.; Paul, A.; Rho, S. Urban planning and building smart cities based on the internet of things using big data analytics. *Comput. Netw.* **2016**, *101*, 63–80. [CrossRef]
5. Leung, K.; Wu, J.T.; Leung, G.M. Real-time tracking and prediction of COVID-19 infection using digital proxies of population mobility and mixing. *Nat. Commun.* **2021**, *12*, 1501. [CrossRef] [PubMed]
6. Mehrotra, A.; Hendley, R.; Musolesi, M. Towards multi-modal anticipatory monitoring of depressive states through the analysis of human-smartphone interaction. In Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, Heidelberg, Germany, 12–16 September 2016; pp. 1132–1138.
7. Lathia, N.; Pejovic, V.; Rachuri, K.K.; Mascolo, C.; Musolesi, M.; Rentfrow, P.J. Smartphones for large-scale behavior change interventions. *IEEE Pervasive Comput.* **2013**, *12*, 66–73. [CrossRef]
8. Wang, R.; Chen, F.; Chen, Z.; Li, T.; Harari, G.; Tignor, S.; Zhou, X.; Ben-Zeev, D.; Campbell, A.T. StudentLife: Assessing mental health, academic performance and behavioral trends of college students using smartphones. In Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Seattle, WA, USA, 13–17 September 2014; pp. 3–14.
9. Ashbrook, D.; Starner, T. Learning significant locations and predicting user movement with GPS. In Proceedings of the IEEE Sixth International Symposium on Wearable Computers, Seattle, WA, USA, 7–10 October 2002; pp. 101–108.
10. Ashbrook, D.; Starner, T. Using GPS to learn significant locations and predict movement across multiple users. *Pers. Ubiquitous Comput.* **2003**, *7*, 275–286. [CrossRef]
11. Song, L.; Kotz, D.; Jain, R.; He, X. Evaluating location predictors with extensive Wi-Fi mobility data. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2003**, *7*, 64–65. [CrossRef]
12. Monreale, A.; Pinelli, F.; Trasarti, R.; Giannotti, F. WhereNext: A location predictor on trajectory pattern mining. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; ACM: New York, NY, USA , 2009; pp. 637–646.
13. Prabhala, B.; La Porta, T. Spatial and temporal considerations in next place predictions. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 390–395.
14. Do, T.M.T.; Gatica-Perez, D. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive Mob. Comput.* **2014**, *12*, 79–91. [CrossRef]

15. Baumann, P.; Kleiminger, W.; Santini, S. The influence of temporal and spatial features on the performance of next-place prediction algorithms. In Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Zurich, Switzerland, 8–12 September 2013; pp. 449–458.
16. Feng, J.; Li, Y.; Zhang, C.; Sun, F.; Meng, F.; Guo, A.; Jin, D. Deepmove: Predicting human mobility with attentional recurrent networks. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1459–1468.
17. Zeng, J.; He, X.; Tang, H.; Wen, J. A next location predicting approach based on a recurrent neural network and self-attention. In Proceedings of the Collaborative Computing: Networking, Applications and Worksharing: 15th EAI International Conference, CollaborateCom 2019, London, UK, 19–22 August 2019; Proceedings 15; Springer: Berlin/Heidelberg, Germany, 2019; pp. 309–322.
18. Yang, D.; Fankhauser, B.; Rosso, P.; Cudré-Mauroux, P. Location Prediction over Sparse User Mobility Traces Using RNNs: Flashback in Hidden States! In Proceedings of the IJCAI, Twenty-Ninth International Joint Conference on Artificial Intelligence, Yokohama, Japan, 7–15 January 2021; Bessiere, C., Ed.; pp. 2184–2190 .
19. Kingma, D.; Salimans, T.; Poole, B.; Ho, J. Variational diffusion models. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 21696–21707.
20. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
21. Luca, M.; Barlacchi, G.; Lepri, B.; Pappalardo, L. A survey on deep learning for human mobility. *ACM Comput. Surv. (CSUR)* **2021**, *55*, 1–44. [CrossRef]
22. Lan, S.; Ma, Y.; Huang, W.; Wang, W.; Yang, H.; Li, P. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In Proceedings of the International Conference on Machine Learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 11906–11917.
23. Zhang, W.; Liu, H.; Liu, Y.; Zhou, J.; Xu, T.; Xiong, H. Semi-supervised city-wide parking availability prediction via hierarchical recurrent graph neural network. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 3984–3996. [CrossRef]
24. Wang, L.; Chai, D.; Liu, X.; Chen, L.; Chen, K. Exploring the generalizability of spatio-temporal traffic prediction: Meta-modeling and an analytic framework. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 3870–3884. [CrossRef]
25. Canzian, L.; Musolesi, M. Trajectories of depression: Unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. In Proceedings of the UbiComp: 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan, 7–11 September 2015; Mase, K., Langheinrich, M., Gatica-Perez, D., Gellersen, H., Choudhury, T., Yatani, K., Eds.; ACM: New York, NY, USA, 2015; pp. 1293–1304.
26. Pappalardo, L.; Vanhoof, M.; Gabrielli, L.; Smoreda, Z.; Pedreschi, D.; Giannotti, F. An analytical framework to nowcast well-being using mobile phone data. *Int. J. Data Sci. Anal.* **2016**, *2*, 75–92. [CrossRef]
27. Wu, R.; Luo, G.; Shao, J.; Tian, L.; Peng, C. Location prediction on trajectory data: A review. *Big Data Min. Anal.* **2018**, *1*, 108–127.
28. Zheng, Y. Trajectory Data Mining: An Overview. *ACM TIST* **2015**, *6*, 29 . [CrossRef]
29. Calabrese, F.; Lorenzo, G.D.; Ratti, C. Human mobility prediction based on individual and collective geographical preferences. In Proceedings of the 13th International IEEE Conference on Intelligent Transportation Systems, Funchal, Portugal, 19–22 September 2010; pp. 312–317.
30. Gambs, S.; Killijian, M.O.; del Prado Cortez, M.N. Show Me How You Move and I Will Tell You Who You Are. In Proceedings of the SPRINGL '10, Workshop on Security and Privacy in GIS and LBS, San Jose, CA, USA, 2 November 2010; Bertino, E., Damiani, M.L., Saygin, Y., Eds.; ACM: New York, NY, USA, 2010; pp. 34–41. [CrossRef]
31. Sun, J.; Zhang, J.; Li, Q.; Yi, X.; Liang, Y.; Zheng, Y. Predicting citywide crowd flows in irregular regions using multi-view graph convolutional networks. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 2348–2359. [CrossRef]
32. Yao, X.; Gao, Y.; Zhu, D.; Manley, E.; Wang, J.; Liu, Y. Spatial origin-destination flow imputation using graph convolutional networks. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 7474–7484. [CrossRef]
33. Abideen, Z.U.; Sun, H.; Yang, Z.; Ahmad, R.Z.; Iftekhar, A.; Ali, A. Deep wide spatial-temporal based transformer networks modeling for the next destination according to the taxi driver behavior prediction. *Appl. Sci.* **2020**, *11*, 17. [CrossRef]
34. Yang, D.; Zhang, D.; Zheng, V.W.; Yu, Z. Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs. *IEEE Trans. Syst. Man Cybern. Syst.* **2015**, *45*, 129–142. [CrossRef]
35. Yang, D. Dingqi YANG's Homepage, 2021. Available online: https://sites.google.com/site/yangdingqi/home (accessed on 14 May 2023).
36. Cho, E.; Myers, S.A.; Leskovec, J. 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, San Diego, CA, USA, 21–24 August 2011; Apté, C., Ghosh, J., Smyth, P., Eds.; ACM: New York, NY, USA, 2011; pp. 1082–1090.
37. Moro, A.; Kulkarni, V.; Ghiringhelli, P.A.; Chapuis, B.; Huguenin, K.; Garbinato, B. Breadcrumbs: A Rich Mobility Dataset with Point-of-Interest Annotations. In Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago, IL, USA, 5–8 November 2019; pp. 508–511.
38. Gjoreski, M.; Laporte, M.; Langheinrich, M. Toward privacy-aware federated analytics of cohorts for smart mobility. *Front. Comput. Sci.* **2022**, *4*, 891206. [CrossRef]
39. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual, 6–10 July 2020; pp. 753–763.

40. Lim, B.; Arık, S.Ö.; Loeff, N.; Pfister, T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *Int. J. Forecast.* **2021**, *37*, 1748–1764. [CrossRef]
41. Ezequiel, C.E.J.; Gjoreski, M.; Langheinrich, M. Federated Learning for Privacy-Aware Human Mobility Modeling. *Front. Artif. Intell.* **2022**, *5*, 867046. [CrossRef] [PubMed]
42. EU Publication. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation). 2016. Available online: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R067 (accessed on 20 March 2021).
43. Langheinrich, M. Privacy by design—principles of privacy-aware ubiquitous systems. In Proceedings of the Ubicomp 2001: Ubiquitous Computing: International Conference, Atlanta, GA, USA, 30 September–2 October 2001; Proceedings; Springer: Berlin/Heidelberg, Germany, 2001; pp. 273–291.
44. Lahlou, S.; Langheinrich, M.; Röcker, C. Privacy and trust issues with invisible computers. *Commun. ACM* **2005**, *48*, 59–60. [CrossRef]
45. Feng, J.; Rong, C.; Sun, F.; Guo, D.; Li, Y. PMF: A privacy-preserving human mobility prediction framework via federated learning. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2020**, *4*, 1–21. [CrossRef]
46. Sweeney, L. k-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **2002**, *10*, 557–570. [CrossRef]
47. Romashov, P.; Gjoreski, M.; Sokol, K.; Martinez, M.V.; Langheinrich, M. BayCon: Model-agnostic Bayesian Counterfactual Generator. In Proceedings of the 31st International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022.
48. Dzieżyc, M.; Gjoreski, M.; Kazienko, P.; Saganowski, S.; Gams, M. Can we ditch feature engineering? end-to-end deep learning for affect recognition from physiological sensor data. *Sensors* **2020**, *20*, 6535. [CrossRef]