AP-Traj2: Transformer-based Trajectory Prediction with Graph-Enhanced Attention Mechanism

Ana-Paula Galarreta[1], Hugo Alatrista-Salas[1], Miguel Nunez-del-Prado[2], Vincent Gauthier[3]
[1]Pontificia Universidad Católica del Peru
[2]Universidad Peruana de Ciencias Aplicadas, Peru
[3]SAMOVAR, Telecom SudParis, Institut Polytechnique de Paris, France
E-mail: a.galarreta@pucp.pe, halatrista@pucp.pe, miguel.nunez@upc.pe, vincent.gauthier@telecom-sudparis.eu

Keywords: Trajectory prediction, deep learning, graph, transformers, whereabout prediction, neural network

Received: February 27, 2024

Trajectory prediction is essential for understanding human mobility patterns, with applications such as itinerary recommendation and urban planning. It involves analyzing sequences of visited locations to forecast the user's next destination. Traditional approaches have often relied on Markov chains or recurrent neural networks (RNNs). More recently, Transformer neural networks have gained attention for sequential prediction tasks due to their superior parallelization and training efficiency. In this study, we propose AP-Traj2 (Attention and Possible directions for TRAJectory prediction 2), a model designed to enhance prediction accuracy by leveraging attention mechanisms and graph-based movement modeling. AP-Traj2 employs self-attention to capture dependencies among visited locations, explores feasible next steps through a graph of possible directions, and incorporates contextual information via location embeddings. Experiments conducted on GPS, CDR, and WiFi datasets demonstrate that AP-Traj2 improves the average match ratio by approximately 50% *over state-of-the-art methods. Moreover, it achieves significantly faster training times, with reductions of up to* 72% *in the best-case scenario. Unlike existing approaches that focus primarily on neural network architecture, this work emphasizes the importance of data preprocessing and filtering, highlighting their substantial impact on model performance.*

Povzetek: Študija predstavi AP-Traj2, ki z združitvijo samo-pozornosti, grafa možnih smeri in vgrajevanj lokacij izboljša natančnost in učinkovitost napovedovanja naslednje destinacije iz zaporedij lokacij.

1 Introduction

Spatio-temporal data provide valuable insights into complex phenomena involving both spatial and temporal dynamics. Today, the availability of GPS data allows us to study a wide variety of trajectories. In addition, other sources such as Call Detail Records (CDRs) and Wi-Fi data could also be used to generate trajectories and make predictions about the whereabouts of specific devices [1, 2]. Previous research has focused on predicting the future positions of various entities such as vehicles, animals, and humans [3, 4, 5, 6, 7].

Traditionally, RNN [8, 9, 10, 11] models have been used for trajectory analysis, where the sequence of positions is considered. Some works have addressed the issue of the time required to generate models for predicting the trajectories of moving objects (see Table 1). Efficient training helps to reduce costs and enables organizations to take on more significant challenges associated with the study of moving objects.

This paper aims to predict the next discrete locations with improved accuracy and reduced training times when compared to the state-of-the-art. Hence, the present effort presents the Attention and Possible directions for TRAJectory prediction 2 (AP-Traj2), a model designed to predict the next position in a discrete geographic system. In this approach, a user's trajectory coordinates are first transformed into a sequence of nodes on a directed graph. The prediction of the next nodes a user is likely to visit is based on the directed graph, the sequence of previously visited nodes and a sequence of previous directions taken. In our study, we used datasets from different sources and extracted location information in different ways. Specifically, for GPS data, the nodes in the graph correspond to road intersections. In the case of CDR data, each node is associated with the region surrounding a particular antenna, *i.e.* lower granularity is used compared to GPS. Similarly, for Wi-Fi data, each node represents a specific building. Using this information, AP-Traj2 uses three modules to predict a user's future whereabouts: a node self-attention module, a possible directions module, and a location embedding module. More in detail, we try to understand how does adding a self-attention mechanism improve trajectory prediction over existing RNN-based approaches?

The node self-attention module uses the self-attention mechanism in the Transformer's encoder to capture the dependencies between nodes. This allows the model to effectively capture the contextual relationships between vis-

ited locations and use them for trajectory prediction. The possible directions module uses the graph representation to determine the feasible paths from an original node to its neighboring nodes, facilitating the prediction of the user's movement. Finally, the location embedding module captures information about the general area previously visited by the user, providing additional context for trajectory prediction.

Experiments conducted on GPS, CDR, and WiFi datasets demonstrate that AP-Traj2 improves the average match ratio by approximately 50% over state-of-the-art methods. Moreover, it achieves significantly faster training times, with reductions of up to 72% in the best-case scenario. Our approach offers the advantage of faster training, making it a suitable solution for trajectory prediction tasks in the context of discrete geographic systems.

The following sections of this article are organized as follows. Section 2 provides a comprehensive review of scientific studies on trajectory prediction. Section 3 provides a detailed explanation of the proposed model, AP-Traj2. Section 4 describes the methodology used in this study. Section 5 details the experimental setup and results. Finally, Section 7 concludes with a summary of the findings and suggestions for future research directions.

2 State of the art

Next whereabouts prediction has been widely studied in the literature. However, capturing and modeling spatial dependency remains an unsolved challenge [20]. Several models have been proposed for predicting the next item in a sequence. Table 1 provides a summary of such models, which mainly rely on deep learning methods and show varying performance and limitations, as discussed below. In particular, convolutional neural network (CNN) models [14], recurrent neural network (RNN) models [8, 9, 10, 12, 13, 11], and Transformer-based models [15, 16, 17, 18, 19] have been used.

It is worth noting that most studies rely on GPS datasets, with some using Automatic Identification System (AIS) data [19, 12] or sensor data (e.g., highway sensors) [15]. Additionally, the evaluation metrics used vary significantly across studies, making direct comparisons challenging. Moreover, training times are rarely reported, despite their relevance for practical applications. In this work, we aim to improve prediction accuracy while maintaining training efficiency, which we will explicitly measure.

Since Transformers have demonstrated to give good results while achieving faster training times, our architecture employs the Transformer's encoder and a graph of the visited area to predict the next location that a user is going to visit.

While Table 1 shows steady advances in next-place prediction, it also highlights some key limitations. First, many existing models do not explicitly integrate both spatial and temporal constraints, treating them as separate signals or ignoring one altogether. Second, most rely on architectures that require extensive training time or lack personalization, limiting their scalability and adaptability. Third, although some models incorporate graph structures or attention mechanisms, these are not always aligned with user-specific trajectories. These gaps motivate the design of **AP-Traj2**, which combines spatial and temporal restricted sequential patterns with the efficiency of Transformer encoders and contextual graph representations, to achieve accurate, efficient, and personalized predictions.

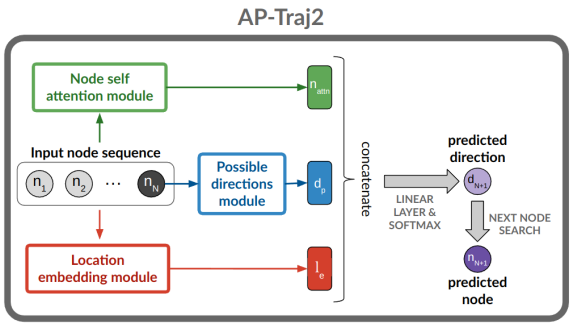

Figure 1: Attention and Possible directions for Trajectory prediction 2 (AP-Traj2). It contains three modules: Node self-attention module, Possible directions module and Location embedding module. Each of them produces an output vector, and these vectors are then concatenated and processed to predict the user's next visited node.

3 The AP-Traj2 model

The present section details *AP-Traj2*, the new extension for the **A**ttention and **P**ossible directions for **Traj**ectory prediction model (AP-Traj) [21]. One key improvement is how location information is represented. While AP-Traj uses the embedding of the last node in the input sequence, AP-Traj2 incorporates a dedicated *Location embedding module*, designed to provide a richer representation of the user's trajectory.

To predict the next location, the proposed model takes a sequence of N nodes as input and consists of three modules, as shown in Figure 1. The first module, in green, is the *Node self-attention module*, which uses the Transformer's encoder to capture the importance of the input nodes. The second module, in blue, is the *Possible directions module*, which is responsible for identifying the possible directions that can be taken from the last node of the input sequence nodes using an annotated graph that contains information about the area traveled. Finally, the *Location embedding module*, in red, combines the embeddings of the input node sequence, generating a vector that encapsulates information about the location of the user's trajectory. Each module generates an output vector, which is combined and processed to predict the user's next visited node. These three modules are described in detail in the following sections.

Table 1: Summary of models, datasets, and performance metrics

	Name	Model	Dataset	Dataset size	Technology	Metric	Value	Training time
[8]	DeepMove	LSTM	Foursquare check-in NYC Foursquare check-in Tokio Mobile application location Call Detail Records	82K records 537K records 15M records 491K records	GPS GPS GPS CDR	Top-1 PA* Top-1 PA Top-1 PA Top-1 PA	0.167 0.206 0.694 0.593	not reported
[9]	DeNavi	LSTM + EWMA	Gowalla check-in BrightKite check-in	6.4 M interactions 4.5 M interactions	GPS GPS	ACC@20 ACC@20	0.2460 0.5831	not reported
[10]	CLNN	LSTM + MLP	Porto taxi trajectory Geolife	1.7 million records 18,670 trajectories	GPS GPS	MHD error MHD error	1.6864 2.9401	not reported
[12]	—	GRU	AIS data - China	7.5M points	AIS	accuracy	0.98	135.97 (units unknown)
[13]	—	LSTM+GCNN	Didi - China	2M records	GPS	MAE	1.201	not reported
[11]	NetTraj	LSTM + ATTN	T-Drive Shanghai Trajectories	15M records used 4M records	GPS GPS	AMR AMR	69.7 65.8	3,61 h/iter 2,75 h/iter
[14]	—	CNN	HRI Driving Dataset	43 hours	GPS	Precision	0.94	
[15]	Space timeformer	Transformer	Metr-LA Traffic Pems-Bay Traffic NY-TX Weather AL Solar	34k time steps 52k time steps 570K time steps 52k time steps	sensor sensor sensor sensor	MSE MSE MSE MSE	38.27 13.99 12.49 7.75	not reported
[16]	Forecaster	Transformer + graph	NYC taxi dataset	65.4M datapoints	GPS	RMSE	5.19	not reported
[17]	PreCLN	Transformer	Porto taxi trajectory T-Drive	1.7 million records 17M records	GPS GPS	MAE MAE	193.15 55.90	not reported
[18]	—	Transformer	Green Class Yumuv	139 participants 498 participants	GPS GPS	ACC@10 ACC@10	73.50 66.92	not reported
[19]	TrAISformer	Transformer	AIS data - Denmark	712M records	AIS	mean error (nmi)	0.48	60 min/10 days of data

3.1 Node self-attention module

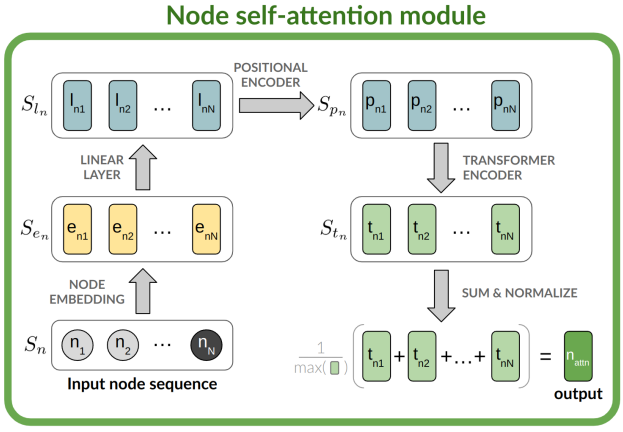

Figure 2: Node self-attention module schema

As shown in Figure 2, this module takes as input a sequence of nodes $S_n = \{n_1, ..., n_N\}$ and generates the vector $n_{attn} \in \mathbb{R}^{1\times d}$, which captures the importance of each node in the input sequence.

To this aim, the first step is to compute the embedding for each unique node, shown as node embedding in Figure 2. This step is done by learning a parameter matrix $W_{n_e} \in \mathbb{R}^{|N_T|\times d}$, where $|N_T|$ is the total number of nodes present in the dataset and d denotes the dimensions of the output vector n_{attn}. The result is a sequence of embeddings denoted by $S_{e_n} = \{e_{n_1}, e_{n_2}, ..., e_{n_N}\} \in \mathbb{R}^{N\times d}$, where N is the number of nodes in the input sequence. Note that N and $|N_T|$ usually have different values. For example, the input sequence might consist of $N = 10$ nodes, while there might be $|N_T| = 50$ different nodes in the dataset used to obtain the sequences.

Next, a linear layer is applied to S_{e_n} to adjust the magnitude of each vector within it. This is done by learning a parameter matrix $W_{l_n} \in \mathbb{R}^{d\times d}$, which ensures that the vectors are not heavily influenced by the subsequent positional encoding. This results in the sequence $S_{l_n} = \{l_{n_1}, l_{n_2}, ..., l_{n_N}\} \in \mathbb{R}^{N\times d}$.

Then, following the approach introduced by Vaswani et al. [22], a positional encoder is used to preserve the sequential order information of the input sequence. This involves augmenting each l_{n_i} with a corresponding position vector, resulting in the sequence $S_{p_n} = \{p_{n_1}, p_{n_2}, ..., p_{n_N}\} \in \mathbb{R}^{N\times d}$. Then, the sequence S_{p_n} is passed through the Transformer's encoder. This encoder uses a self-attention mechanism [23] that establishes relationships between different positions within the same sequence to generate a meaningful representation of that sequence. This encoder is constructed as a stack of identical layers, with each layer consisting of two subcomponents: a multi-head attention mechanism and a simple, position-wise, fully connected feedforward network. To improve the training of deep networks, a residual connection is built around each of these subcomponents. This residual connection acts as a shortcut, allowing the network to bypass one or more layers in both forward and backward passes. Then, a normalization layer is applied, resulting in a transformed representation denoted as $S_{t_n} = \{t_{n_1}, t_{n_2}, ..., t_{n_N}\} \in \mathbb{R}^{N\times d}$.

Finally, the outputs of the Transformer's encoder are

added and normalized to produce the output vector:

$$n_{attn} = \frac{1}{max(t_{n_i})} \sum_{i=1}^{N} t_{n_i}, n_{attn} \in \mathbb{R}^{1\times d}, \quad (1)$$

where n_{attn} is a vector that represents the significance of the nodes from input sentence.

3.2 Possible directions module

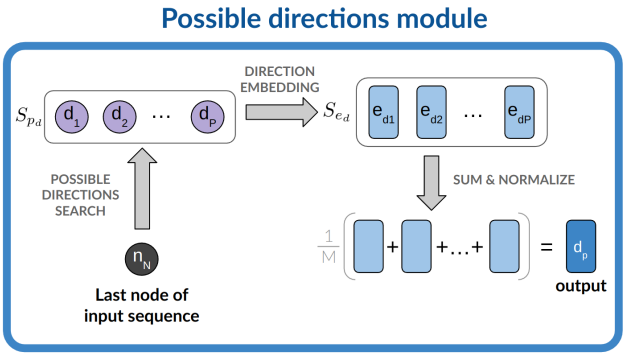

Figure 3: Possible directions module schema

This module, illustrated in Figure 3, focuses on the last node of the input sequence (n_N) and generates $S_{p_d} \in \mathbb{R}^{1\times d}$, which encapsulates information about the possible directions to follow from the last node.

To achieve this, an annotated directed graph $G_a = (V, E_a)$ is used, where:

$$V = \{n_1, n_2, \ldots, n_k\}$$
$$E_a = \{(n_1, n_2, dir_{12}), (n_2, n_3, dir_{23}), \ldots\}$$

Here, V is the set of nodes representing the elements in the input sequence and E_a is the set of directed edges, where each edge, has a direction dir_{ij} represented by an integer number representing the direction ranging from node n_i to node n_j. This direction dir_{ij} is obtained by following [11], where the angle θ_{ij} between the coordinates of nodes n_i and n_j is calculated. Since this angle can take any continuous value in the range $[0, 360[$, it has been discretized by dividing the space into K equal intervals, so that each has the same width $W = 360/K$.

Hence, for nodes $n_i = \{x_i, y_i\}$ and $n_j = \{x_j, y_j\}$:

$$\theta_{ij} = tan^{-1}\left(\frac{y_j - y_i}{x_j - x_i}\right) \times \frac{180}{\pi}$$
$$dir_{ij} = \lfloor \frac{\theta_{ij}}{360/K} \rfloor$$

For example, if $K = 8$, there are 8 possible directions, where 0 represents a direction between the east and the northeast.

Using G_a, the first step is to search for the edges connected to the last node to identify all possible directions that can be taken from it. This way, we obtain $S_{p_d} = \{d_1, d_2, ..., d_P\}$, where P is the number of possible directions.

Then, the embeddings of the possible directions are computed using a parameter matrix $W_{d_e} \in \mathbb{R}^{K\times d}$. As a result, for any given node we obtain a sequence of direction embeddings denoted by $S_{e_d} = \{e_{d1}, e_{d_2}, ..., e_{dP}\} \in \mathbb{R}^{P\times d}$.

The number of possible directions P varies from node to node, as it represents the number of neighboring nodes of each node.

Finally, the resulting vectors are summed and normalized to produce the output vector:

$$d_p = \frac{1}{max(e_{di})} \sum_{i=1}^{P} e_{d_i}, d_p \in \mathbb{R}^{1\times d}, \quad (2)$$

where e_{d_i} is the embedding of each direction and d_p consolidates all direction embeddings into a single vector.

3.3 Location embedding module

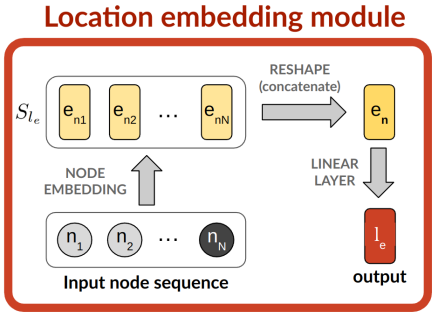

Figure 4: Location embedding module schema

A third module produces the vector $l_e \in \mathbb{R}^{1\times d}$ to store information about the area where the user moves. To obtain this vector, we use the same parameter matrix $W_{n_e} \in \mathbb{R}^{|N_T|\times d}$ used in the *Node self-attention module*. This matrix is utilized to compute the embeddings of all the input nodes to produce a sequence of location embeddings $S_{l_e} \in \mathbb{R}^{N\times d}$, as shown in Figure 4. Then, these embeddings are concatenated to form the vector $e_n \in \mathbb{R}^{1\times(N\times d)}$. Finally, a linear layer is applied to e_n to obtain the vector $l_e \in \mathbb{R}^{1\times d}$ by learning a parameter matrix $W_{l_l} \in \mathbb{R}^{(N\times d)\times d}$).

3.4 Next node prediction

After obtaining the outputs from the three modules, as shown in Figure 1, the resulting vectors are concatenated:

$$d_C = [n_{attn}; d_P; l_e], \quad d_C \in \mathbb{R}^{1\times(3d)} \quad (3)$$

The resulting vector d_C is then passed through a linear layer (by learning a parameter matrix $W_f \in \mathbb{R}^{3d\times K}$), followed by the application of the softmax function. This operation yields the probabilities of the K possible directions from the input sequence's last node (n_N).

Once the direction with the highest probability (d_{prob}) is computed, it is compared to the possible directions from n_N. This is done by searching the edges connected to this node in the graph G_a to find the closest match based on the difference between d_{prob} and all the possible directions. This process identifies the predicted direction (d_{N+1}). To break ties, the one with the lower value is selected. For example, if $d_{prob} = 2$ and the possible directions are 1 and 3, the predicted direction would be $d_{N+1} = 1$ because 1 has the lowest value ($1 < 3$). Finally, the predicted node (n_{N+1}) is obtained by using the predicted direction (d_{N+1}) and G_a.

3.5 Prediction of subsequent nodes

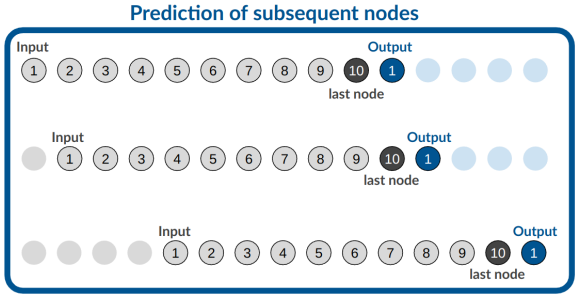

Figure 5: Prediction of subsequent nodes. When a node is predicted, it is concatenated with the last nodes of the original sequence to create a new input sequence.

To extend the prediction to multiple nodes instead of just one, a sliding window method is used. After predicting the node n_{N+1} using the input sequence S_{n_1}, a new input sequence S_{n_2} is constructed:

$$S_{n_1} = \{n_1, n_2, ..., n_N\} \quad (4)$$

$$S_{n_2} = \{n_2, n_3, ..., n_{N+1}\} \quad (5)$$

The sequence S_{n_2} is then fed into the above architecture, and the node n_{N+2} is predicted. This process is repeated for M iterations, where M is the length of the output sequence and the number of nodes predicted.

An example can be seen in Figure 5, where an input sequence of 10 nodes is used to predict a single output node. Then, the last 9 nodes of the original sequence are concatenated with the first predicted node to produce a new input sequence. This prediction process is then repeated to obtain the subsequent nodes.

4 Methodology

The proposed methodology, shown in Figure 6, transforms GPS, Wi-Fi, or CDR data into a format suitable for the AP-Traj2 model to be trained. It consists of three main stages: graph construction and annotation, sequence preprocessing, and model training. The **graph construction and annotation** phase generates an annotated directed graph $G_a = (V, E_a)$ that captures the connectivity between all the nodes the users can visit. This graph is then used in the **sequence preprocessing** stage, where the sequences of locations (represented as local or absolute coordinates) are transformed into sequences of nodes and directions. Next, this information is used in the **model training** stage to predict the subsequent nodes to be visited by a user.

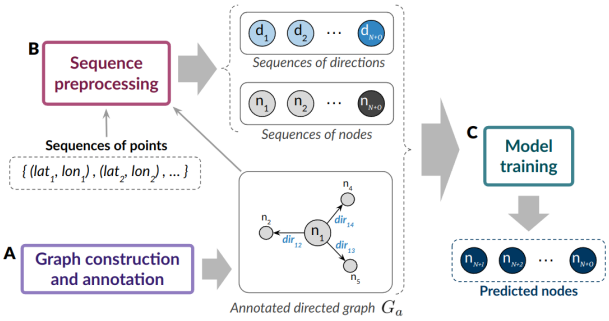

Figure 6: Proposed methodology for transforming input data and training the AP-Traj2 model. It has three main stages: graph construction, sequence preprocessing, and model training.

The following paragraphs describe this process in detail.

4.1 Graph construction and annotation

The first step of the preprocessing phase is to obtain an annotated and directed graph $G_a = (V, E_a)$, which contains information about the connections between all the locations available in each dataset. The method used to construct the graph depends on the characteristics of the dataset. However, the procedure used to annotate it is the same for all of them.

This step consists in obtaining a directed graph $G = (V, E)$, where:

$$V = \{n_1, n_2, \ldots, n_k\}$$
$$E = \{(n_1, n_2), (n_2, n_3), \ldots\}$$

In the case of GPS data, the graph G, representing the road network, can be obtained directly from an external source like *Open Street Maps* (OSM), with road intersections as nodes and road segments as edges. However, for other trajectory datasets, such as those containing Wi-Fi or CDR data, where it is necessary to construct the graph G, from the data itself, showing the connections between each of the specific locations visited (e.g., an administrative division or the coverage area of an antenna). In this case, we will construct the graph G taking into account the proximity between these locations. To achieve this, given a set of locations with their coordinates, Voronoi regions are obtained by using the Python library *GeoVoronoi* [24], which computes the region around each location that is closer to that location than any other one. Then, an unweighted graph is constructed using *NetworkX* [25], a Python package for

network analysis. To achieve this, the adjacency between regions is defined by Queen contiguity, which means that two regions are considered neighbors if they share at least one edge or one vertex. Once the adjacency relationships between the regions are defined, an unweighted graph is created by representing each region as a node and connecting adjacent regions with edges. In this case, edges are considered to be bidirectional (*c.f.*, Figure 7).

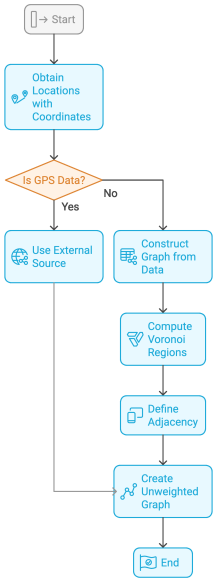

Figure 7: Graph construction flowchart

The graph $G = (V, E)$ is annotated to create the graph $G_a = (V, E_a)$. In G_a each edge e_{ij} stores the direction dir_{ij} taken when moving from node n_i to node n_j. As previously explained in Section 3.2, these directions are determined by calculating the angle between connected nodes and then discretizing this angle with a constant K parameter.

It is important to clarify that the edges in the annotated graph $G_a = (V, E_a)$ carry directional weights based on the discretized movement directions between nodes, which are used as features during model training. These direction values effectively serve as edge weights.

Additionally, for all datasets considered, the constructed graph fully covers all locations present in the trajectory data. There are no missing nodes, as each observed location corresponds to a node in the graph. For GPS data, nodes correspond to intersections obtained from external road network sources. For Wi-Fi and CDR datasets, nodes represent coverage areas or regions constructed via Voronoi tessellation, ensuring complete representation of all visited locations.

4.2 Sequence preprocessing

As shown in Figure 6, once we have obtained the graph G_a, we preprocess the sequences of locations to collect a sequence of nodes for each trajectory and its corresponding sequence of directions. This phase consists of four steps: Trajectory identification, Sample selection, Edge matching and Direction search.

The specific procedures used to collect this information depend on the characteristics of the dataset and the data available. In the case of GPS data, the inputs consist of a sequence of points represented as absolute coordinates and a graph representing road segments. For other types of data, however, the inputs consist of a sequence of points represented by specific regions, a table containing the absolute or relative coordinates associated with each region, and a graph showing the connections between the regions based on their proximity. Each of the four steps for sequence processing will be described in the following paragraphs.

4.2.1 Trajectory identification

The process has two stages. In the first stage, the sequences of points are broken down into smaller sequences. In the second stage, trajectories are selected based on whether they are located within a specific geographic region.

a. Sequence split. To identify stops and trajectories for GPS data, the Infostop algorithm [26] is used. It works by first dividing the data into segments and determining whether they represent a stop or just a temporary halt. To classify the points as trips or stays, Infostop uses three main parameters (1) the maximum roaming distance allowed between two points within the same stay (r_1), (2) the minimum time difference between two consecutive records for them to be considered within the same stay (t_{min}), and (3) the maximum time difference between two consecutive records to be considered within the same stay (t_{max}).

For example, given a sequence of points $seq = \{p_1, p_2, p_3, p_4, p_5\}$. Each point is tagged as a trip tr or a stay st $\{p_1 \rightarrow tr, p_2 \rightarrow tr, p_3 \rightarrow tr, p_4 \rightarrow st, p_5 \rightarrow tr\}$ In this case, for the first 5 points we would have two trajectories $traj_1 = \{p_1, p_2, p_3, p_4\}$ and $traj_2 = \{p_4, p_5\}$, which are sequences of points that represent the change in position of an object.

Note that we keep the stationary points (points tagged as stays) and use them as the start or end point of each trajectory. This is why the stationary point p_4 appears twice in the previous example (once in each trajectory).

Furthermore, since the experiments are performed with an input sequence of size N and an output sequence of size M, only the sequences of size $N + M$ are kept. This is done to preserve a larger amount of information associated with the original sequence.

b. Trajectory filtering. Some datasets contain trajectories that span over very large geographic regions. This can happen, for example, if the points within the dataset come from two different cities. In this case, the density of points for a given area is greatly reduced,

which reduces the quality of the training process. To avoid this, if the obtained trajectories come from different geographic regions, only those that are within a certain boundary are selected. This boundary could be a specific geographic region, such as a city.

4.2.2 Sample selection

To increase the number of experiments with each dataset and to test more hyperparameters during the tuning phase, a representative subset is selected by creating bounding boxes that cover the geographic region where the trajectories are located, such as a particular city. Next, the samples located within each bounding box are extracted and their inter-event distance and time interval distributions are compared to the entire population. The representativeness of the sample is then tested using the Chi-square goodness-of-fit test, which determines whether the sample is a good fit to the population. Acceptance of the null hypothesis indicates that the sample is representative. Finally, if multiple samples meet the conditions, the one with the highest number of data points is selected.

4.2.3 Edge matching

The AP-Traj2 method consists of predicting subsequent nodes, given a user's trajectory. This means that each sequence of points must be matched with a specific sequence of nodes. Two different methods are used to achieve this, depending on how the input trajectory is represented:

a. Points represented by absolute coordinates. If the input is a sequence of points represented as absolute coordinates, each point is matched to an edge of G_a using the ST-Matching tool [27] to transform each sequence of points into a sequence of road segments:

$$S_e = \{e_{01}, e_{12}, ..., e_{(N-1)N}\} \tag{6}$$

Then, considering that we will train the model with an input sequence of size N and an output sequence of size M, all sequences containing less than $N + M$ nodes are discarded to ensure a sufficient amount of information for the training stage.

b. Points represented by a region. If the input trajectory consists of a sequence of points represented by a region related to certain absolute or relative coordinates, each of these points is considered a node. Then, successive nodes are identified as stops, and the sequences are split into trajectories. Finally, Dijkstra's algorithm is used to find the shortest path between the nodes by using G_a, thus determining the user's trajectory.

It is worth noting that in cases where the user's location data is incomplete, our approach can still generate a trajectory that includes neighboring nodes. This can be achieved using either the ST-Matching tool or Dijkstra's algorithm.

4.2.4 Direction search

To increase the efficiency of the proposed model by limiting the number of possible predictions, we predict only one of the K available directions that can be taken from a given node.

Using G_a, all the information about a user's trajectory is represented as a sequence of nodes (S_n) and a sequence of directions (S_d) in the following way:

$$S_n = \{n_0, n_1, n_2, ..., n_N\} \tag{7}$$

$$S_d = \{dir_{01}, dir_{12}, ..., dir_{(N-1)N}\} \tag{8}$$

Note that S_n has one more element than S_d, since $dir_{(i-1)i}$ indicates the direction taken to go from n_{i-1} to n_i.

4.3 Model training

To generate the input and output sequences for the training phase, the initial node in S_n is excluded. Then, the S_n and S_d sequences are partitioned into shorter sequences of length of $N + M$, where N is the input size and M is the output size. For example, if the input size is $N = 10$ and the output size is $M = 5$, this means that 10 nodes and their corresponding directions are going to be used to predict the subsequent 5 nodes and directions. Hence, the original sequences must be partitioned into sequences of length $N + M = 15$.

Then, the generated sequences and the graph G_a are used to train the **AP-Traj2** model. The following section provides a detailed description of the experiments performed.

5 Experiments

Dataset characteristics, graph generation, sequence preprocessing, evaluation metrics, and model training are described in this section.

5.1 Datasets

In the present effort, the datasets for the experiments are: (1) **T-Drive**, which contains GPS trajectories of 9,657 taxis in Beijing, China for six days in February 2008 [28]. A total of 128,074 individual sequences representing trajectories were extracted. (2) **Geolife**, having 18,670 GPS sequences representing the trajectories of 182 users from April 2007 to October 2011 in China [29]. (3) **D4D Ivory Coast** CDR dataset, consisting of 500,000 randomly selected trajectories in Ivory Coast between December 1, 2011 and April 28, 2012. For user privacy, location is reported at the subprefecture level, reducing spatial accuracy [30]. The total number of sequences representing trajectories is 4,860,992. (4) **D4D Senegal**, comprising country level CDR data for 2013 in Senegal. For privacy reasons, only a 2-week period of randomly selected users is reported [31]. In our study, only a subset of 319,508 trajectories of unique users was

used. (5) **KTH wireless (Stockholm)** dataset, which contains records of authenticated user connections to the wireless network at the KTH Royal Institute of Technology in Stockholm during 2014 and 2015 [32]. This dataset contains information on 109,183 users, and a total of 483,004 sequences representing trajectories were extracted.

The main characteristics of these datasets can be observed in Table 2, which shows the technology used to obtain the data, the number of users, the number of total sequences, the number of days observed, the average sequence length and its standard deviation.

5.2 Graph construction and annotation

As it was described in section 4.1, an annotated directed graph G_a containing information about the connections between all the locations in each dataset is needed. The construction and subsequent annotation of this graph is explained in the following lines.

5.2.1 Graph construction

In the case of the GPS datasets (T-Drive and Geolife), the Beijing road network obtained from OSM through OSMnx [33], was used. The resulting graph G contains 46,805 nodes and 110,268 edges, where each node represents a road intersection and each edge represents a road segment.

However, since the D4D Ivory Coast, D4D Senegal, and KTH wireless contain CDR and Wi-Fi data, each graph was constructed based on the proximity between the antenna location. For more details on this procedure, see section 4.1. The graph G created for the D4D Ivory Coast has 255 nodes and 2,824 edges, the one for D4D Senegal has 1,666 nodes and 9,630 edges, and the KTH wireless graph has 59 nodes and 648 edges.

5.2.2 Graph annotation

For all five datasets, the graphs G were annotated according to the procedure described in Section 4.1, in order to obtain G_a. A value of $K = 8$ was used for the T-Drive and Geolife datasets, while $K = 12$ was used for the D4D Ivory Coast, D4D Senegal, and KTH wireless datasets. These values were chosen based on the maximum number of directions that can be taken from a given node in each dataset. Our choice ensured that direction embeddings captured sufficient spatial variance without over-fragmenting the space. It's important to note that if a single node yields the same discretized direction for two neighbors, the duplicated direction obtained in second place is replaced with the next closest direction.

5.3 Sequence preprocessing

In this step, the obtained graph G_a and a sequence of points are processed to obtain the sequences of nodes and directions needed to train the model. As previously described in Section 4.2, the procedure varies depending on the dataset. In the following paragraphs, we will explain the exact steps taken for each individual dataset.

5.3.1 Trajectory identification

This process, described in Section 4.2.1, includes the sequence split and trajectory filtering steps:

a. Sequence split. This allows the identification of stops and trajectories on GPS datasets. This step was only performed on the T-Drive dataset, because it consists of taxi trajectories, but it does not contain an occupied flag, *i.e.* we have no information regarding whether a taxi driver is driving a passenger to a specific location or not. In total, 128,074 sequences representing taxi trajectories were extracted. However, this step was considered unnecessary for the Geolife dataset, which also contains GPS data, because the individual user trajectories were already separated. Finally, since the other three datasets used do not contain GPS data, this procedure was not performed on them.

b. Trajectory filtering. Considering that both the T-Drive and Geolife datasets cover a very large geographic region, mostly consisting on the Beijing area in China and its surroundings, the point density is reduced significantly, which decreases the quality of the training process. To avoid this, only points within the Beijing city area were considered. For the T-Drive dataset, about 94.5% of the trajectories were retained, while for the Geolife dataset, about 90% of the trajectories were retained. It is worth noting that this specific procedure was not applied to the other three datasets.

5.3.2 Sample selection

As it was explained previously in Section 4.2.2, to increase the number of experiments for the hyperparameter tuning, a representative subset of each dataset was selected based on how its inter-event distance and time interval distributions compare to those of the general population.

For the T-Drive, Geolife, D4D Ivory Coast and D4D Senegal datasets, a sample selection was performed based on the results of an inter-event distance analysis. However, due to the limited number of sequences in the KTH wireless dataset, this analysis and the subsequent sample selection were unnecessary, as all the sequences were used in the preprocessing steps.

The results of the inter-event analysis can be seen in Table 3, where we can see that the inter-event time and distance vary greatly in the analyzed datasets.

Based on the obtained distributions, we selected samples from the T-Drive, Geolife, and D4D Senegal datasets. The bounding boxes used to extract these samples are [39.855, 39.980, 116.342, 116.467] for T-Drive, [39.917, 40.017, 116.254, 116.354] for Geolife and [14.658, 14.788, -17.456, -17.326] for D4D Senegal.

Table 2: Dataset statistics

Metrics	T-Drive	Geolife	D4D Senegal	D4D Ivory Coast	KTH wireless
technology	GPS	GPS	CDR	CDR	Wi-Fi
num users	9,657	182	319,508	500,000	109,183
num seq	128,074	18,670	319,508	4,860,992	483,004
num days	7	1,645	365	150	713
seq len avg	126	1,332	139	152	193
seq len std	710	2,821	167	288	348

Table 3: Inter-event analysis

Metrics	T-Drive	Geolife	D4D Senegal	D4D Ivory Coast
time avg. (s)	441	7	8,166	7,567
time std. dev. (s)	3,699	191	21,425	25,683
distance avg. (m)	3,815	52	1853	1056
distance std. dev. (m)	183,632	5847	13,565	9,387

It is important to note that for the D4D Ivory Coast dataset, we initially performed an inter-event distance analysis, but ultimately decided not to select a sample. Instead, we decided to use the entire dataset for hyperparameters fine-tuning. This decision was driven by the small number of sequences obtained after the subsequent edge and node matching step.

5.3.3 Edge and node matching

As explained in Section 4.2.3, two different approaches can be used to transform a sequence of points into a sequence of nodes by using G_a. In the case of the Geolife and T-Drive datasets, the points are represented by absolute coordinates, so the ST-Matching tool was used to transform each sequence of points into a sequence of road segments.

On the other hand, since the points in the D4D Ivory Coast, D4D Senegal, and KTH wireless datasets are represented by regions, consecutive nodes were identified as stops. In cases where the raw data contained incomplete trajectories—i.e., users leaped between non-consecutive nodes in the constructed graph—Dijkstra's algorithm was applied to estimate the shortest path between observed locations and reconstruct a plausible trajectory.

5.3.4 Direction search

This process was performed for all datasets, using their corresponding sequences of edges S_e and annotated directed graphs G_a.

5.4 Evaluation metrics

To evaluate the effectiveness of AP-Traj2, the Match Ratio (MR) and the Average Match Ratio (AMR) were used. The Match Ratio in Equation 10 is calculated as the average of the matches between the predicted values and the ground truth:

$$MR(k) = \frac{1}{k}\sum_{i=1}^{k} match(y, \hat{y}), \tag{9}$$

where:

$$match(y, \hat{y}) = \begin{cases} 1, & \text{when } y = \hat{y} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

On the other hand, the Average Match Ratio is defined by Equation 11, where O represents the number of predicted nodes and $MR(k)$ is the Match Ratio up to the k-th predicted node.

$$AMR = \frac{1}{O}\sum_{k=1}^{O} MR(k) \tag{11}$$

5.5 Model training

5.5.1 Hyperparameter tuning

In the case of the T-Drive, Geolife, and D4D Senegal datasets, representative sample bounding boxes were used to explore multiple hyperparameter values, as indicated in section 5.3.2. However, for the D4D Ivory Coast and KTH wireless datasets, the entire population was used.

Based on [11] the sequences for all datasets were divided into segments of length $N + M = 15$. Each segment had an input size of $N = 10$ nodes and their corresponding directions, and an output size of $M = 5$ nodes and directions. Although N is a tunable parameter, it was fixed to 10 during training to focus on tuning model-specific hyperparameters. The impact of varying N is analyzed separately in Section 5.5.2. Then, the obtained sequences were randomly divided into training, validation, and test sets. The number of sequences in each split for every dataset is shown in Table 4. All datasets were split using an 80% training, 10% validation, and 10% test ratio.

The main hyperparameters tuned for AP-Traj2 are the following:

Table 4: Hyperparameter tuning: number of sequences

Dataset	Training	Test	Validation
T-Drive	20,376	2,547	2,547
Geolife	24,219	3,027	3,027
D4D Senegal	24,478	3,060	3,060
D4D Ivory Coast	11,650	1,450	1,450
KTH wireless	24,510	3,065	3,065

Table 5: Hyperparameter tuning: search space

Parameter	Tested Values
Embedding dimension (d)	$\{64, 128, 256\}$
Feedforward dimension	$\{256, 512, 1024\}$
Number of heads	$\{2, 4\}$
Number of layers	$\{2, 4\}$
Dropout rate	$\{0.1, 0.2\}$
Initialization range	$\{0.1, 0.2\}$

- *dim embedding*. This parameter determines the embedding dimensions for the nodes and directions (d). It also corresponds to the number of expected features in the input of the Transformer's encoder.
- *dim feedforward*. This parameter is related to the Transformer's encoder.
- *num heads*. Returns the number of heads in the multi-head attention model.
- *num layers*. This is the number of sub-encoder layers in the Transformer's encoder.
- *dropout*. Corresponds to the dropout rate used to prevent overfitting. Two values were tested: 0.1 and 0.2, which correspond to deactivating 10% and 20% of the neurons during training, respectively.
- *init range*. A uniform distribution with a range from *-init range* to *init range* has been used used for weight initialization.

The embedding dimensions, Transformer encoder settings (number of layers, number of heads, and feedforward size), dropout rate, and initialization range were selected based on common practices. Specifically, they are either equal to or less than the values used in [22]. The specific values used for tuning are summarized in Table 5.

This decision was made to balance model performance and resource requirements, as larger models would require more memory and processing power. In addition, we conducted experiments with different dropout and initial range values to further fine-tune the hyperparameters of our model for each specific dataset. This allowed us to optimize the model's performance based on the data's characteristics.

The training was done using a Stochastic Gradient Descent (SGD) optimizer with a *batch size = 20*, *maximum number of iterations = 100* and early stopping with *patience = 1*, meaning that the training stops the first time the validation loss increases. The loss function used during training was the Cross-Entropy Loss, a common choice used not only in our baseline [11] but also in several other studies, including [8], [17], and [18]. It is important to note that the Cross-Entropy Loss was calculated by comparing the predicted direction to the ground truth, and not the predicted node. Although activation functions are not explicitly defined in custom layers, we have used PyTorch's implementation of `TransformerEncoderLayer`, which applies ReLU as the default activation function in the feed-forward sub-layer. No additional activation functions were added outside this component.

Table 6 shows the combinations of values that gave the best results when comparing the AMR with $k = M = 5$. It is important to note that each parameter combination was tested 10 times, resulting in 1440 trials for each dataset, for a cumulative total of 7200 trials.

5.5.2 Optimal input length

After tuning the hyperparameters, a larger sample was extracted from the T-Drive, Geolife, and D4D Senegal datasets. The bounding boxes for this larger sample are [39.817, 39.992, 116.304, 116.479] for Drive, [39.740, 40.050, 116.140, 116.600] for Geolife and [14.658, 14.808, -17.481, -17.331] for D4D Senegal. Five different lengths were tested using this larger sample to determine the optimal input length, along with the full D4D Ivory Coast and KTH wireless datasets.

To ensure consistency, the number of training, test, and validation sequences was controlled for each dataset. When testing different input lengths, the data was initially segmented using an input length of $L = 10$. To evaluate shorter input lengths (e.g., $L = 8$ or $L = 6$), nodes were removed from the beginning of these segments rather than reprocessing the entire dataset. This allowed us to keep the same number of training, validation, and test sequences across different values of L, ensuring a fair comparison while controlling for dataset size. The trials for each input length were run 10 times on each dataset. The number of training, test and validation sequences for each dataset is given in Table 7. Here, the number of D4D Ivory Coast and KTH Wireless sequences remains the same compared to Table 4 because all available sequences were used in the fine-tuning phase for these two datasets. However, there is a significant increase in the number of sequences for the T-Drive, Geolife, and D4D Senegal datasets.

No trade-off between training time and accuracy was observed in our experiments. All models were tuned to prioritize prediction accuracy, and training times remained competitive. While additional iterations could potentially improve results further, this was not required to reach the reported performance.

The results shown in Figure 8 indicate that the optimal input sequence length (L) for predicting future locations varies depending on the dataset, given the specific hyperparameters and training data size used. In the case of the

Table 6: Hyperparameter tuning: best values

Parameter	T-Drive	Geolife	D4D Senegal	D4D Ivory Coast	KTH wireless
dim embed.	256	256	256	256	256
dim feedf.	256	256	256	1024	512
num heads	4	2	4	2	4
num layers	4	4	2	4	2
dropout	0.2	0.2	0.2	0.1	0.2
init range	0.2	0.2	0.2	0.2	0.2

Table 7: Optimal input length: number of sequences

Dataset	Training	Test	Validation
T-Drive	134,967	16,871	16,871
Geolife	80,915	10,115	10,115
D4D Senegal	93,216	11,652	11,652
D4D Ivory Coast	11,650	1,450	1,450
KTH wireless	24,510	3,065	3,065

T-Drive dataset, we observe a consistent increase in AMR as the input sequence length increases, leading to the best performance when $L = 10$. However, the other datasets do not show the same pattern. For the Geolife dataset, the optimal performance is achieved with $L = 8$, while for the Stockholm dataset, it is achieved with $L = 6$. On the other hand, both the D4D Senegal and D4D Ivory Coast datasets give the best results when $L = 4$. Therefore, these results do not allow us to conclude that a larger input sequence consistently leads to better AMR, except in the case of the T-Drive dataset.

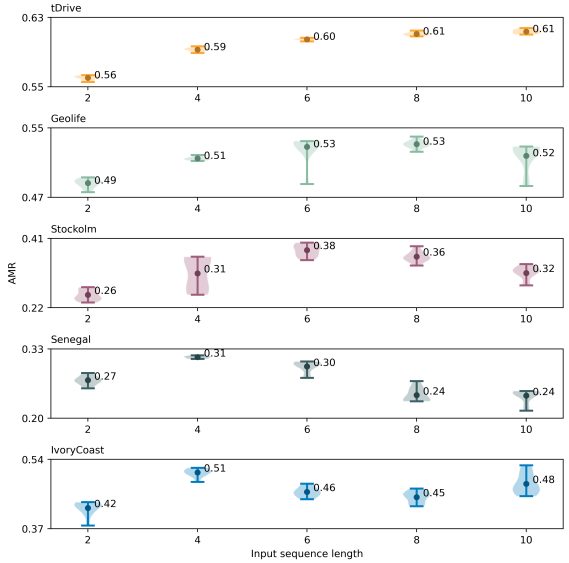

Figure 8: Average match ratio (AMR) versus input sequence length

5.5.3 Baseline comparison

To evaluate the effectiveness of AP-Traj2, a comparison with the NetTraj method [11] was performed. The NetTraj method was implemented to the best of our ability, without the use of contextual information (*i.e.*, weather conditions, season, etc.) and using the same hyperparameters as specified by the authors, except for the input sequence length. Instead, we used the previously determined optimal input lengths. It is important to note that for the NetTraj method, an inverse sigmoid decay schedule was used, and the decay rate was fine-tuned in the range of 1 to 12 with increments of 0.5. The results in Figure 9 show the significant performance advantage of the AP-Traj2 method over the baseline, especially for the GPS datasets such as T-Drive and Geolife. Also, we can see that the AP-Traj2 method has significantly faster training times compared to the baseline for the GPS datasets. Furthermore, the AP-Traj2 method also demonstrates shorter training times for the Wi-Fi and CDR datasets, which can potentially reduce computational costs.

In summary, across all five datasets, AP-Traj2 has superior AMR performance compared to Net-Traj, while achieving faster training times.

6 Discussion

The superior performance of AP-Traj2 across all five datasets, in terms of both prediction accuracy (AMR) and training time, highlights the advantages of the proposed architecture. While we compare directly to NetTraj, it is worth noting that NetTraj has itself been benchmarked against a wide array of state-of-the-art models, including LSTM encoder-decoder [34], Caser [35], AT-LSTM [36], ATST-LSTM [37], SASRec [38], and GeoSAN [39]. By outperforming NetTraj, AP-Traj2 demonstrates competitive performance relative to these methods as well.

Several architectural choices contribute to AP-Traj2's improvements. First, the incorporation of spatial and temporal restricted sequential patterns enables the model to capture fine-grained local dependencies that traditional RNN or attention-only architectures may overlook. Second, the use of a Transformer encoder improves efficiency and scalability, particularly in long-range sequence modeling, while avoiding the training overhead associated with recurrent layers. Finally, the integration of a contextual

graph constructed from the visited area provides additional structure and spatial awareness, which supports more informed predictions and helps reduce overfitting in data-sparse contexts.

Although we did not conduct an ablation study specifically for AP-Traj2, our previous work on AP-Traj [21] included such an analysis. That study demonstrated that the last node embedding, node self-attention, and possible directions modules each contribute significantly to performance, with the last node embedding having a particularly strong effect on prediction quality. Motivated by these findings, we focused on enhancing the last node embedding module in this paper, which led to the development of AP-Traj2.

We observe that the model performs better on GPS datasets compared to Wi-Fi and CDR datasets. This may be due to higher spatial granularity and denser sampling in GPS data, which allows the model to learn more informative patterns. In contrast, CDR and Wi-Fi data are sparser and use coarser spatial units (e.g., antennas or access points), potentially limiting prediction accuracy.

We also note that datasets with finer granularity benefit more from longer input sequences (Figure 8). This suggests that when the information per time step is more precise, the model can leverage additional temporal context to improve predictions.

We have not explicitly studied the impact of data sparsity—i.e., how performance changes when fewer data points or trajectories are available. Investigating AP-Traj2's robustness to sparse data, possibly through data augmentation or regularization techniques, is an important direction for future work.

The model's reliance on historical trajectories with consistent spatial patterns may hinder its performance in highly irregular or noisy datasets. Additionally, the graph construction and pattern extraction steps require offline preprocessing, which may limit applicability in real-time scenarios or systems with constrained latency requirements. Future work may explore strategies to incrementally update graph representations and adapt to unseen locations more effectively.

Although we have not explicitly evaluated AP-Traj2 on datasets larger than T-Drive, the approach's design principles and its performance on existing data indicate potential for scalability. Nevertheless, further research is needed to assess its generalizability and efficiency in real-time or large-scale applications.

While our model showed strong results across datasets with varying spatial resolutions and densities, further evaluation is needed to assess its behavior in extremely large geographic areas or with denser trajectory data. Although our approach constrains candidate movements through a fixed number of discretized directions, which helps maintain efficiency even at scale, graph annotation and node retrieval times may increase proportionally with the number of nodes. Future work may explore optimization strategies for preprocessing and search to better accommodate larger graphs without impacting overall performance.

While AP-Traj2 achieves competitive accuracy without sacrificing training efficiency, we did not explicitly explore the trade-off between training time and accuracy. Our hyperparameter tuning prioritized performance, and no simplifications were made for faster training. It remains possible that longer training or further optimization could improve results even further.

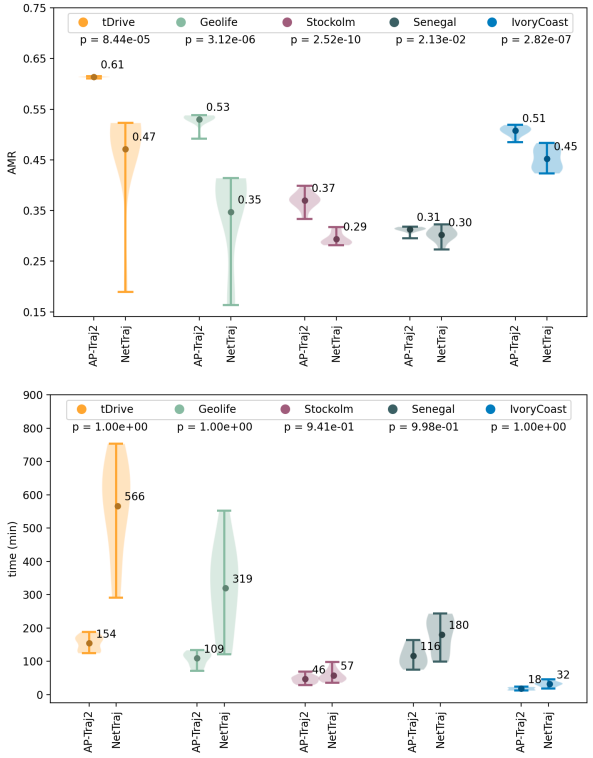

Figure 9: Average match ratio (AMR) and total training time in minutes (min) for Net-Traj and AP-Traj2

7 Conclusion

This research paper presents AP-Traj2 (Attention and directions for TRAJectory prediction 2), a novel model capable of working with location data issued from different technologies such as GPS, CDR, and WI-FI for predicting future whereabouts based on a graph representation. AP-Traj2 includes a self attention module, a possible directions module, and a location embedding module to predict the next location a user is likely to visit. In addition, this study emphasizes the importance of graph preprocessing stages to improve the overall performance of the model. Unlike existing approaches in the field of ANN-based techniques, which primarily focus on the structure of the structure of the neural network, we address some crucial aspects of the data preprocessing and filtering steps, recognizing their impact on model accuracy, as shown by previous research [40]. Through experiments on five real datasets (see Figure 9), we show that AP-Traj2 outperforms the state-of-the-art

model NetTraj in terms of achieving a higher 50% Average Match Ratio (AMR) while exhibiting 72% lower training times.

The contribution of the present effort beyond the performance, AP-Traj2 incorporates a discretized and annotated directed graph of the area traversed by users (road intersections for GPS, coverage regions for CDR and Wi-Fi data). This graph helps the model limit next-location predictions to feasible spatial directions, improving prediction accuracy. AP-Traj2 shows robust performance with different types of spatio-temporal data sources: GPS, Call Detail Records (CDR), and Wi-Fi. However, results indicate better predictions for high-granularity, dense GPS data compared to the sparser CDR and Wi-Fi datasets. Experiments reveal that the optimal input sequence length varies by dataset, related to the spatial granularity and sampling frequency. Thus, for T-Drive and CDR data the best length sequences were ten and four, respectively. We showed that data preprocessing, such as graph construction, sequence identification, and trajectory filtering, affect significantly prediction quality. Additionally, the architectural design and constrained output space theoretically suggests that AP-Traj2 offers efficient inference with low latency due to parallelizable Transformer layers and limited directional choices.

Looking ahead, future research can investigate the integration of external information sources, such as weather data or vacation schedules, to assess their potential impact on prediction accuracy. While these factors may influence movement patterns, further study is needed to quantify their effectiveness. Additionally, the methodology proposed in this work could be easily extended to predict the next whereabouts in other domains, such as forecasting the movement of various vehicles beyond taxis and vessels, or even tracking animal movements, given its generalizability.

Acknowledgments

This project was funded by the *Pontificia Universidad Católica del Perú* (PUCP) and the Peruvian National Program for Scientific Research and Advanced Studies (*ProCiencia*) through the Doctoral Scholarship in Engineering. Additionally, it received support from PUCP through the Marco Polo Fund.

References

[1] D. Bachir, G. Khodabandelou, V. Gauthier, M. El Yacoubi, and J. Puchinger, "Inferring dynamic origin-destination flows by transport mode using mobile phone data," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 254–275, 2019. [Online]. Available: https://doi.org/10.1016/j.trc.2019.02.013

[2] F. Asgari, A. Sultan, H. Xiong, V. Gauthier, and M. A. El-Yacoubi, "Ct-mapper: Mapping sparse multimodal cellular trajectories using a multilayer transportation network," *Computer Communications*, vol. 95, pp. 69–81, 2016, mobile Traffic Analytics. [Online]. Available: https://doi.org/10.1016/j.comcom.2016.04.014

[3] L. Huang, J. Zhuang, X. Cheng, R. Xu, and H. Ma, "Sti-gan: Multimodal pedestrian trajectory prediction using spatiotemporal interactions and a generative adversarial network," *IEEE Access*, vol. 9, pp. 50 846–50 856, 2021. [Online]. Available: 10.1109/ACCESS.2021.3069134

[4] Y. Pang, X. Zhao, J. Hu, H. Yan, and Y. Liu, "Bayesian spatio-temporal graph transformer network (b-star) for multi-aircraft trajectory prediction," *Knowledge-Based Systems*, p. 108998, 2022. [Online]. Available: https://doi.org/10.1016/j.knosys.2022.108998

[5] R. W. Liu, M. Liang, J. Nie, Y. Yuan, Z. Xiong, H. Yu, and N. Guizani, "Stmgcn: Mobile edge computing-empowered vessel trajectory prediction using spatio-temporal multi-graph convolutional network," *IEEE Transactions on Industrial Informatics*, 2022. [Online]. Available: 10.1109/TII.2022.3165886

[6] K. Shao, Y. Wang, Z. Zhou, X. Xie, and G. Wang, "Trajforesee: How limited detailed trajectories enhance large-scale sparse information to predict vehicle trajectories?" in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2189–2194. [Online]. Available: 10.1109/ICDE51399.2021.00222

[7] J. Bray, J. T. Feldblum, and I. C. Gilby, "Social bonds predict dominance trajectories in adult male chimpanzees," *Animal Behaviour*, vol. 179, pp. 339–354, 2021. [Online]. Available: https://doi.org/10.1016/j.anbehav.2021.06.031

[8] J. Feng, Y. Li, Z. Yang, Q. Qiu, and D. Jin, "Predicting human mobility with semantic motivation via multi-task attentional recurrent networks," *IEEE Transactions on Knowledge and Data Engineering*, 2020. [Online]. Available: 10.1109/TKDE.2020.3006048

[9] Y.-C. Chen, T. Thaipisutikul, and T. K. Shih, "A learning-based poi recommendation with spatiotemporal context awareness," *IEEE Transactions on Cybernetics*, vol. 52, no. 4, pp. 2453–2466, 2022. [Online]. Available: 10.1109/TCYB.2020.3000733

[10] J. Tang, J. Liang, T. Yu, Y. Xiong, and G. Zeng, "Trip destination prediction based on a deep integration network by fusing multiple features from taxi trajectories," *IET Intelligent Transport Systems*, vol. 15, no. 9, pp. 1131–1141, 2021. [Online]. Available: https://doi.org/10.1049/itr2.12075

[11] Y. Liang and Z. Zhao, "Nettraj: A network-based vehicle trajectory prediction model with directional representation and spatiotemporal attention mechanisms," *IEEE Transactions on Intelligent Transportation Systems*, 2021. [Online]. Available: 10.1109/TITS.2021.3129588

[12] Y. Suo, W. Chen, C. Claramunt, and S. Yang, "A ship trajectory prediction framework based on a recurrent neural network," *Sensors*, vol. 20, no. 18, p. 5133, 2020. [Online]. Available: https://doi.org/10.3390/s20185133

[13] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data," *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 62–77, 2020. [Online]. Available: https://doi.org/10.1016/j.trc.2020.01.010

[14] M. A. Rahim, S. D. Khan, S. Khan, M. Rashid, R. Ullah, H. Tariq, and S. Czapp, "A novel spatio–temporal deep learning vehicle turns detection scheme using gps-only data," *IEEE Access*, vol. 11, pp. 8727–8733, 2023. [Online]. Available: 10.1109/ACCESS.2023.3239315

[15] J. Grigsby, Z. Wang, and Y. Qi, "Long-range transformers for dynamic spatiotemporal forecasting," *arXiv preprint arXiv:2109.12218*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2109.12218

[16] Y. Li and J. M. Moura, "Forecaster: A graph transformer for forecasting spatial and time-dependent data," *arXiv preprint arXiv:1909.04019*, 2019. [Online]. Available: https://doi.org/10.48550/arXiv.1909.04019

[17] B. Yan, G. Zhao, L. Song, Y. Yu, and J. Dong, "Precln: Pretrained-based contrastive learning network for vehicle trajectory prediction," *World Wide Web*, pp. 1–23, 2022. [Online]. Available: https://doi.org/10.1007/s11280-022-01121-3

[18] Y. Hong, H. Martin, and M. Raubal, "How do you go where? improving next location prediction by learning travel mode information using transformers," *arXiv preprint arXiv:2210.04095*, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2210.04095

[19] D. Nguyen and R. Fablet, "Traisformer-a generative transformer for ais trajectory prediction," *arXiv preprint arXiv:2109.03958*, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2109.03958

[20] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert Systems with Applications*, vol. 207, p. 117921, 2022. [Online]. Available: https://doi.org/10.1016/j.eswa.2022.117921

[21] A.-P. Galarreta, H. Alatrista-Salas, and M. Nunez-del Prado, "Predicting next whereabouts using deep learning," in *International Conference on Modeling Decisions for Artificial Intelligence*. Springer, 2023, pp. 214–225. [Online]. Available: 10.1007/978-3-031-33498-6_15

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016. [Online]. Available: https://doi.org/10.48550/arXiv.1601.06733

[24] M. Konrad. (2022) geovoronoi – a package to create and plot voronoi regions inside geographic areas. Accessed: January, 2023. [Online]. Available: https://pypi.org/project/geovoronoi/

[25] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008. [Online]. Available: https://www.osti.gov/biblio/960616

[26] U. Aslak and L. Alessandretti, "Infostop: Scalable stop-location detection in multi-user mobility data," *arXiv preprint arXiv:2003.14370*, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.2003.14370

[27] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate gps trajectories," in *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, 2009, pp. 352–361. [Online]. Available: https://doi.org/10.1145/1653771.1653820

[28] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 316–324. [Online]. Available: https://doi.org/10.1145/2020408.2020462

[29] Y. Zheng, X. Xie, W.-Y. Ma *et al.*, "Geolife: A collaborative social networking service among user, location and trajectory." *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010. [Online]. Available: http://sites.computer.org/debull/A10june/geolife.pdf

[30] V. D. Blondel, M. Esch, C. Chan, F. Clérot, P. Deville, E. Huens, F. Morlot, Z. Smoreda, and C. Ziemlicki, "Data for development: the d4d challenge on mobile phone data," *arXiv preprint arXiv:1210.0137*, 2012. [Online]. Available: https://doi.org/10.48550/arXiv.1210.0137

[31] Y.-A. de Montjoye, Z. Smoreda, R. Trinquart, C. Ziemlicki, and V. D. Blondel, "D4d-senegal: the second mobile phone data for development challenge," *arXiv preprint arXiv:1407.4885*, 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1407.4885

[32] L. Pajevic, G. Karlsson, and V. Fodor, "Crawdad dataset kth/campus (v. 2019-07-01)," *CRAWDAD Wirel Network Data Arch*, vol. 10, 2019. [Online]. Available: https://doi.org/10.15783/c7-5r6x-4b46

[33] G. Boeing, "Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks," *Computers, Environment and Urban Systems*, vol. 65, pp. 126–139, 2017. [Online]. Available: https://doi.org/10.1016/j.compenvurbsys.2017.05.004

[34] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE intelligent vehicles symposium (IV)*. IEEE, 2018, pp. 1672–1678. [Online]. Available: 10.1109/IVS.2018.8500658

[35] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 565–573. [Online]. Available: 10.1145/3159652.3159656

[36] S. Capobianco, L. M. Millefiori, N. Forti, P. Braca, and P. Willett, "Deep learning methods for vessel trajectory prediction based on recurrent neural networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 57, no. 6, pp. 4329–4346, 2021. [Online]. Available: 10.1109/TAES.2021.3096873

[37] L. Huang, Y. Ma, S. Wang, and Y. Liu, "An attention-based spatiotemporal lstm network for next poi recommendation," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 1585–1597, 2019. [Online]. Available: 10.1109/TSC.2019.2918310

[38] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 197–206. [Online]. Available: 10.1109/ICDM.2018.00035

[39] D. Lian, Y. Wu, Y. Ge, X. Xie, and E. Chen, "Geography-aware sequential location recommendation," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 2009–2019. [Online]. Available: https://doi.org/10.1145/3394486.3403252

[40] Manish, U. Dohare, and S. Kumar, "A survey of vehicle trajectory prediction based on deep learning models," in *Proceedings of Third International Conference on Sustainable Expert Systems: ICSES 2022*. Springer, 2023, pp. 649–664. [Online]. Available: https://doi.org/10.1007/978-981-19-7874-6_48