future internet

MDPI

Article

Dynamic Graph Representation Learning for Passenger Behavior Prediction

Mingxuan Xie [1], Tao Zou [1], Junchen Ye [2,*], Bowen Du [1,2,3] and Runhe Huang [4]

[1] CCSE Lab, Beihang University, Beijing 100191, China; xie_mingxuan@buaa.edu.cn (M.X.); zoutao@buaa.edu.cn (T.Z.); dubowen@buaa.edu.cn (B.D.)
[2] School of Transportation Science and Engineering, Beihang University, Beijing 100191, China
[3] Zhongguancun Lab, Beijing 100190, China
[4] Faculty of Computer and Information Science, Hosei University, Tokyo 102-8160, Japan; rhuang@hosei.ac.jp
* Correspondence: junchenye@buaa.edu.cn

Abstract: Passenger behavior prediction aims to track passenger travel patterns through historical boarding and alighting data, enabling the analysis of urban station passenger flow and timely risk management. This is crucial for smart city development and public transportation planning. Existing research primarily relies on statistical methods and sequential models to learn from individual historical interactions, which ignores the correlations between passengers and stations. To address these issues, this paper proposes DyGPP, which leverages dynamic graphs to capture the intricate evolution of passenger behavior. First, we formalize passengers and stations as heterogeneous vertices in a dynamic graph, with connections between vertices representing interactions between passengers and stations. Then, we sample the historical interaction sequences for passengers and stations separately. We capture the temporal patterns from individual sequences and correlate the temporal behavior between the two sequences. Finally, we use an MLP-based encoder to learn the temporal patterns in the interactions and generate real-time representations of passengers and stations. Experiments on real-world datasets confirmed that DyGPP outperformed current models in the behavior prediction task, demonstrating the superiority of our model.

Keywords: traffic data mining; passenger behavior prediction; graph neural networks; dynamic graph learning

check for updates

Citation: Xie, M.; Zou, T.; Ye, J.; Du, B.; Huang, R. Dynamic Graph Representation Learning for Passenger Behavior Prediction. *Future Internet* **2024**, *16*, 295. https://doi.org/10.3390/fi16080295

Academic Editors: Stefano Rinaldi and Alan Oliveira De Sá

Received: 20 July 2024
Revised: 12 August 2024
Accepted: 13 August 2024
Published: 15 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Nowadays, with the rapid development of transportation technology, subways play a significant role in alleviating traffic pressure in large cities and improving the efficiency of residents' travel, thereby accumulating a large amount of passenger travel records. In intelligent transportation city systems, managing the risk of the passenger flow at stations is crucial for enhancing urban safety and maintaining the stability of infrastructure operations. Passenger behavior prediction (PP) models use passengers' historical boarding and alighting information to reflect their travel habits and preferences. The results can be applied to various downstream tasks, such as analyzing future passenger flow trends at stations based on group travel behavior or managing risks at key urban stations based on crowd travel patterns.

The traditional PP models initially relied on statistical methods and conventional time series prediction analysis. Recently, the application of deep learning techniques to PP tasks has demonstrated higher prediction accuracy and better user learning capabilities. The diffusion convolutional gated recurrent unit (DCGRU) model [1] first generates subgraphs for traffic conditions, learns subgraph messages through a deep convolutional neural network (DCNN), and then captures user behavior habits through a gate recurrent unit (GRU). A multiple temporal units neural network (MTUNN) [2] uses multiple time

units to process information of different time lengths, while a PENN integrates different structural information using various models and then makes the final prediction. A long short-term memory neural network (LSTM_NN) [3] models the long-term patterns of passengers by applying LSTM and also explores the significant impact of weather on passenger flow. Additionally, many traditional methods [4–6] predict temporal features using an autoregressive integrated moving average model (ARIMA), analyze passenger flow with a support vector machine (SVM), or forecast short-term changes using a CNN or other methods.

However, the current works overlook three important features of PP tasks:

Firstly, passenger travel records are continuously growing. Current PP models typically only model the existing historical interaction sequences of passengers to make single-step predictions and evaluations for the next moment. In the real world, passenger travel records are continuously increasing, so prediction tasks should incorporate the latest interaction information in a timely manner, and predictions should be multi-step, with the model being promptly updated based on changes to travel records. Sequence-based solutions in PP models cannot adequately adapt to real-life applications.

Secondly, passenger travel behavior encompasses both long-term and short-term characteristics. Human behavior exhibits periodicity, showing regular patterns over the long term but potentially undergoing abrupt changes in the short term. For example, a passenger might typically commute between home and work from Monday to Friday but stay home or visit a shopping center on weekends. Long-term periodic patterns reveal inherent characteristics of passengers, such as identifying them as workers based on their regular commute between work and home. Short-term variability, however, can be influenced by factors such as weather conditions and the passenger's mood. The existing methods often capture the long-term characteristics of passengers but struggle to effectively extract short-term variability features.

Thirdly, passenger travel behavior involves complex interactions between passengers and stations. A passenger may have entered and exited multiple different stations before a certain time, and a station may have had multiple passengers enter and exit before that time. Therefore, the relationship between a passenger and a station is intricate and complex. Relying solely on statistical methods to determine the probability of a passenger interacting with a station at a given time can overlook the influence of passenger density at the station on the passenger's destination choice.

To address these issues, a new method called **dy**namic **g**raph representation learning for **p**assenger behavior **p**rediction (DyGPP) is proposed as shown in Figure 1. This method aims to track the dynamic travel behavior characteristics of passengers using continuous-time dynamic graphs (CTDG). In this dynamic graph, passengers and stations are represented as heterogeneous nodes, and the entire graph can be viewed as a bipartite graph. The goal is to predict the connection between specific passenger nodes and station nodes at particular times, indicating whether the passenger will interact with the corresponding station at a certain time. This novel design can adapt to the continuously growing interaction records between passengers and stations, while modeling passengers' long-term periodic features and short-term abrupt features using their historical interaction sequences. Experiments on the Beijing subway travel dataset were conducted, and the results demonstrated the effectiveness of our method.

The contributions can be summarized as follows:

- A continuous-time dynamic graph model is proposed for the passenger behavior prediction problem. Unlike previous sequential models, DyGPP treats fine-grained timestamped events as a dynamic graph and learns temporal patterns for both passengers and stations from their historical sampling sequences.
- A temporal pattern encoding module is designed to simultaneously capture the individual temporal patterns for passengers and stations and correlate temporal behaviors between passengers and stations.

- Experiments on real-world datasets demonstrated the effectiveness of our method. Compared to the baseline models, our approach achieved higher average precision and AUC values, demonstrating its superiority.

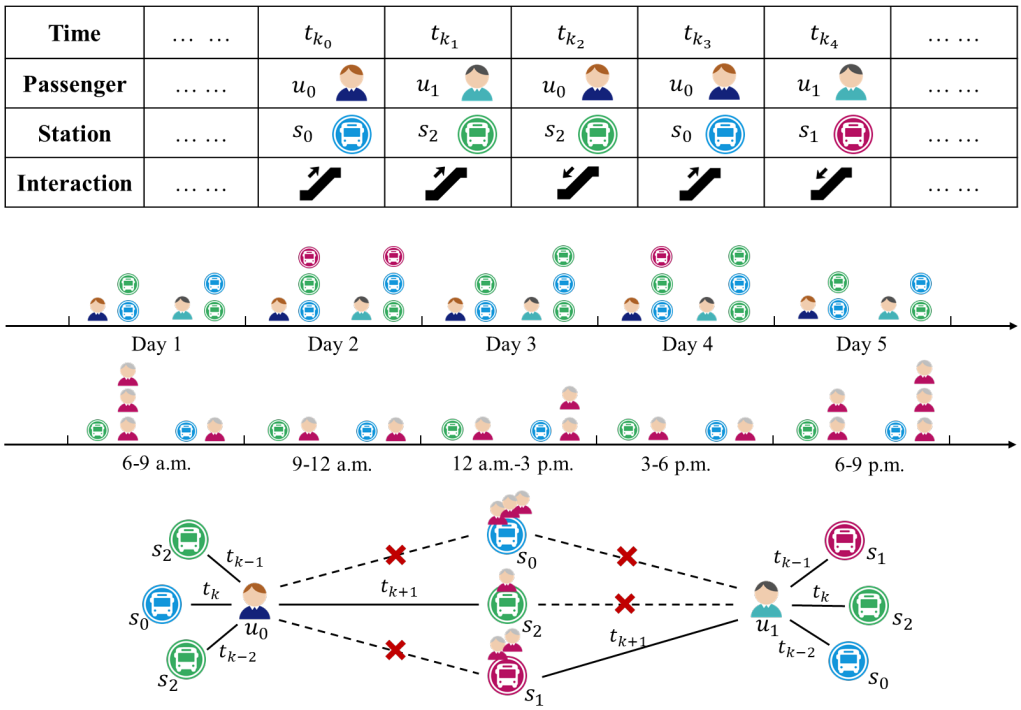

Figure 1. Passenger behavior prediction with a dynamic graph to trace passengers' travel patterns. This task has three dynamics: (1) travel records are infinitely growing; (2) periodic patterns over long intervals and abrupt changes over short intervals. (3) evolving relationships between passengers and stations.

2. Related Works

2.1. Passenger Behavior Prediction

Passenger behavior prediction has garnered significant attention in urban data analysis. Early research [7] proposed a multidimensional analysis of travel behavior to understand people's travel demands. In recent years, travel behavior analysis has been commonly used in urban function perception and smart transportation development. In machine learning methods, neural network models have been effective in solving time series prediction problems [1]. Tsai et al. [2] considered temporal features in neural network models to predict station-based passenger flow. Wei and Chen [8] used passenger flow data from the six time steps before the prediction period as input, combining empirical mode decomposition and NN models to predict passenger flow. Menon and Lee [9] combined a non-homogeneous Poisson process with a single-layer neural network model to predict short-term public transit passenger flow. Zhai et al. [10] proposed a new hierarchical hybrid model for short-term passenger flow prediction, based on time series models, deep belief networks, and an improved incremental extreme learning machine. Liu et al. [3] applied LSTM networks to develop an hourly subway passenger flow prediction model. Yao et al. [4] used convolutional neural networks to predict short-term passenger flow during special events. The work in [5] proposed a short-term passenger flow prediction method for subways based on ARIMA, incorporating weather conditions. Gu et al. [6] proposed a trajectory sequence encoding method and applied it to an interpretable StTP model with adaptive location awareness. Hao et al. [11] proposed an end-to-end framework for multi-step prediction, using an attention mechanism to predict the number of passengers alighting at all stations. Please see Table 1 for a review summary.

Table 1. Summary of related works in passenger behavior prediction.

Ref	Time Series Prediction	Passenger Flow Prediction	Solution
[1]	✓	✗	Neural network models
[2]	✗	✓	Neural network models with temporal features
[3]	✗	✓	LSTM models
[4]	✗	✓	CNN models
[5]	✓	✓	ARIMA model with weather conditions
[6]	✗	✗	StTP model with adaptive location awareness
[8]	✗	✓	NN models with six time steps before prediction
[9]	✗	✓	Non-homogeneous Poisson process
[10]	✓	✓	Hierarchical hybrid model based on time series models
[11]	✓	✗	End-to-end framework for multi-step prediction

2.2. Dynamic Graph Learning

Dynamic graphs describe entities as nodes and represent their interactions as edges with timestamps. In recent years, extensive research [12,13] on dynamic graph representation learning has helped simulate real-world network scenarios more effectively. Research on dynamic graphs can be divided into two types: one decomposes the changing graph structure at time intervals and takes snapshots at the final moment of decomposition [14,15]. The dynamics of the graph are reflected in the changes in these snapshots. However, this method cannot effectively represent all interaction changes in a dynamic graph; it can only show the overall trend. To analyze finer-grained changes in graph structure, some methods add time-based random walks [16,17] to the graph structure to represent node information, while others preserve temporal information in node features [18–20]. Additionally, some approaches use sequence models to capture long-term temporal features in dynamic graph structures [21,22]. Jodie [19] uses an RNN-based encoder to model the evolution of nodes. TGAT [23] extracts and encodes local temporal subgraphs to represent the current state of nodes. TGN [24] extends Jodie with a graph convolutional layer to aggregate information from local neighbors and updates node representations with the aggregated temporal information.

2.3. Time-Based Batch

When calculating the loss for gradient descent to update weights [25–27], we can use all the samples to compute the loss; this method is called batch gradient descent (BGD). Alternatively, we can randomly select a single sample to compute the loss and then the gradient; this method is called stochastic gradient descent (SGD). To balance the advantages of BGD and SGD, mini-batch gradient descent (MBGD) was developed. The Adam optimizer [28] further enhances the MBGD method by introducing first moment and second moment estimates to control the learning rate. However, when it comes to dynamic link prediction problems, the effectiveness of the aforementioned methods may decrease as the batch size increases. Lampert [29] proposed dynamic link forecasting, which maintains prediction accuracy by dividing batches using fixed time intervals. This approach enhances the robustness of the model.

3. Preliminaries

In this section, we first present some necessary definitions and then formalize the studied problem. The notations used and their meanings are shown in the Table 2.

Table 2. Notations and their meanings.

Notation	Meaning
u_i	The ith passenger
s_i	The ith station
$\mathcal{U}$	Collections of passengers
$\mathcal{S}$	Collections of stations
l_k	The kth label, which is 0 when the passenger gets on the train and 1 when they get off the train
t_k	The kth timestamp
e_k	The kth interaction
$\mathcal{G}^t$	The dynamic graph at time t
x_{u_i}	Feature of the passenger u_i
x_{s_i}	Feature of the station s_i
$\hat{p}^t_{u,s}$	The probability of interaction between passenger u and stations s at time t
FC	Fully connected layer
ReLU	Linear rectification function, a type of activation function in neural networks
Sigmoid	A common S-shaped function in biology, it also acts as the activation function for neural networks, mapping variables between 0 and 1

3.1. Definitions

Let $\mathcal{U} = \{u_i \mid i = 1, 2, \ldots, n\}$ and $\mathcal{S} = \{s_j \mid j = 1, 2, \ldots, m\}$ represent the collections of n passengers and m stations. Specifically, u_i represents the i -th passenger and s_j represents the j-th station.

Definition 1. ***Passenger Behavior.*** *The passenger behavior is defined as an interaction $e_k = (u_k, s_k, l_k, t_k)$ where the passenger u_k enters or leaves the subway station s_k at timestamp t_k.*

Definition 2. ***Dynamic Graph.*** *A dynamic graph can be represented as a monotonically non-decreasing sequence of interactions denoted by $\mathcal{G}^t = \{e_1, \cdots, e_k\}$, where $0 < t_1 \leq \cdots \leq t_k \leq t$. In the graph, each passenger node u_i is associated with a node feature $x_{u_i} \in \mathbb{R}^{d_N}$, and the feature of each station node $x_{s_j} \in \mathbb{R}^{\mathbb{N}}$ also carries the same meaning. Additionally, we defined corresponding features for the edges $x^{t_k}_{e_k} \in \mathbb{R}^{d_E}$, where d_N and d_E denote the dimensions of the node feature and the edge feature.*

3.2. Problem Formalization

Passenger Behavior Prediction with a dynamic graph aims to predict the subway station that passenger $u_i \in \mathcal{U}$ will enter or leave in the future. Specifically, given the passenger node u_i, and the historical passenger–station interaction sequence before t_k, i.e., $\mathcal{G}^{t_k}$, we solve such a problem by predicting whether u_i will pass through station $s_j \in \mathcal{S}$ at timestamp t_k. The process can be formalized as below

$$\hat{p}^{t_k}_{u_i,s_j} = f(\mathcal{G}^{t_k})$$

where $\hat{p}^{t_k}_{u_i,s_j}$ is the probability that passenger u_i will pass through the target interaction station s_j at time t_k.

4. Methodology

The overall architecture of our method is illustrated in Figure 2. The model for learning passenger travel behavior can be divided into three components: temporal sequence construction, dynamic representation learning, and future behavior prediction. Specifically, given a predicted interaction e_k at time t_k, we first fetch the recent historical interaction sequence and capture the temporal features of passenger u_k and station s_k via the first component. The second component learns the dynamic behavior pattern for the passenger and passenger flow trend for the station before time t. Then, a prediction layer is utilized

to estimate the probability that passenger u_k would pass through station s_k based on their dynamic representations. Finally, we present a time-batch algorithm to enhance the model efficiency by creating batches based on time intervals, ensuring consistency with the passenger flow over time.

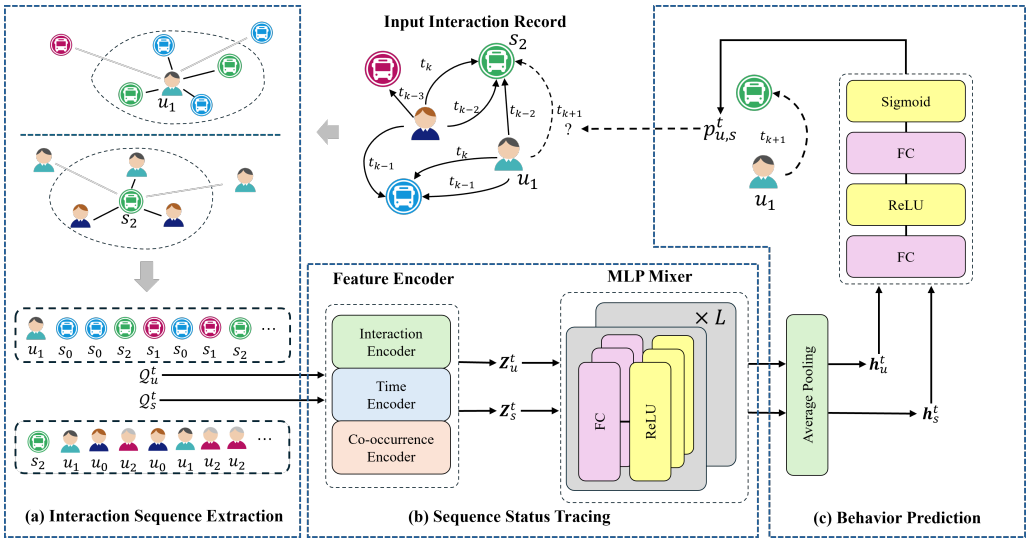

Figure 2. Framework of the DyGPP model.

4.1. Temporal Sequence Construction

Passenger travel behavior tends to be regular in traffic systems, particularly for metro systems [30,31]. To capture the temporal patterns, we first sample the historical first-hop neighbors of passengers and stations. Then, we encode the temporal information and interaction behavior between two nodes via different feature encoders.

4.1.1. Extracting Historical Sequences

The sequence of historical neighbors always reflects the temporal patterns of nodes in the dynamic graphs [18,21,23]. To capture the evolving patterns for passengers and stations, we extract a fixed number of first-order neighbors from an infinitely long historical interaction sequence and construct a corresponding interaction sequence. Formally, this can be expressed as follows: Given a potential interaction $e = (u, s, l, t)$ at time t, we extract the corresponding interaction sequences $\mathcal{Q}_u^t = \{(u, s', l', t') \mid t' < t\} \cup \{(u, u, 0, t)\}$ and $\mathcal{Q}_s^t = \{(u', s, l', t') \mid t' < t\} \cup \{(s, s, 0, t)\}$ for the passenger node u and the station node s with label l before time t. Specifically, we consider that the state of passengers and stations at time t is related to their characteristics, so we treat the node itself as a neighbor and include it in the model. To accurately represent the current node features, we sort the interactions by time and select the N recently interacted neighbors up to time t. For nodes with fewer neighbors, we use a zero-padding technique to expand the historical neighbors, ensuring that each node has N neighbors. This method improves the efficiency of the model without sacrificing accuracy and maintains the stability of model training.

4.1.2. Feature Encoder

For each interaction $e = (u, v, l, t)$, we first obtain the embeddings for each sequence with interaction features, time interval information, and temporal pattern behaviors. Then, we generate the fused representations for each passenger and station by aggregating these embeddings.

Interaction Encoder. Boarding and alighting, as key attributes of passenger–station interactions, are encoded into the interaction features. We use different labels to distinguish between boarding and alighting and write them into the raw features of the edges. Next,

we use a replication padding method to expand the raw feature matrix of the edges, which serves as $\mathbf{X}_E$ in the feature fusion.

Time Encoder. Passenger boarding and alighting behaviors usually exhibit strong periodicity and are highly correlated with time. We developed a time encoder to effectively utilize the time features in historical interaction information. Let the time encoding dimension be d, the predicted timestamp be t, and for passenger u's neighbor s_i interacted at time t_i, we use the $\cos(\cdot)$ to transform the time interval $\Delta t = t - t_i$ into a high-dimensional vector, and then represent the time interval using learnable parameters, followed by [32]. The formula is as follows:

$$\mathbf{x}^t_{s_i,time} = \sqrt{\frac{1}{d_T}}\left[\cos(\omega_1 \Delta t), \cos(\omega_2 \Delta t), \ldots, \cos(\omega_{d_T} \Delta t)\right], \tag{1}$$

where $\mathbf{x}^t_{s_i,time} \in \mathbb{R}^{d_T}$ and $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_{d_T}]^T$ represent the trainable parameters. Hence, the time interval embeddings for the sequence of the passenger and station are denoted as $\boldsymbol{X}^t_{u,T}$ and $\boldsymbol{X}^t_{s,T}$, respectively.

Temporal Pattern Encoding. The existing methods usually calculate the feature information of the passenger sequence and the station sequence separately, ignoring the correlation between the two nodes. We designed a module to consider both separate and cross-temporal patterns in these two sequences. It is worth noting that passenger nodes and station nodes are two different types of node. The historical first-order neighbors of a passenger node are always station nodes, and the historical first-order neighbors of a station node are always passenger nodes. In other words, the historical neighbor set of a passenger node can be denoted as $\mathcal{N}^t_u = \{u\} \cup \{s^{t_1}_1, \ldots, s^{t_n}_n \mid t_n < t\}$, and the station nodes' neighbor set can be denoted as $\mathcal{N}^t_s = \{s\} \cup \{u^{t_1}_1, \ldots, u^{t_n}_n \mid t^{t_n} < t\}$.

On the one hand, to capture the separate temporal patterns, we calculate the interaction frequency of each neighbor in the individual sequence, which aims to learn the long-term temporal behaviors of passengers and stations. On the other hand, we learn the repeated interaction frequency between the passenger and target station in all other sequences. Specifically, we calculate the frequency at which passenger nodes appear in the historical neighbors of station nodes and the frequency at which station nodes appear in the historical neighbors of passenger nodes. The higher the frequency, the stronger the correlation between the two. In other words, if a station node u appears frequently in the historical neighbors of a passenger $\mathcal{N}^t_u$, it is more likely that the passenger will interact with the station at time t.

Formally speaking, given the historical interaction sequence $\mathcal{Q}^t_u$ for passengers and $\mathcal{Q}^t_s$ for stations, we calculate the correlation between the two sequences, denoted as $\mathbf{Co^t_u}$ and $\mathbf{Co^t_s}$. For example, let us assume the historical neighbor sequence of u_1 as $\mathcal{Q}^t_{u_1} = \{u_1, s_1, s_2, s_1, s_3\}$, and the historical neighbor sequence of s_2 as $\mathcal{Q}^t_{s_2} = \{s_2, u_2, u_1, u_1\}$ as shown in Figure 3. The frequency at which u_1 appears in the $\mathcal{Q}^t_{u_1}$, $\mathcal{Q}^t_{s_2}$ sequence is $[1,2]$, then the correlation can be denoted as $\mathbf{Co}^t_{u_1} = \{[1,2],[2,0],[1,1],[2,0],[1,0]\}$. Similarly, $\mathbf{Co}^t_{s_2}$ can be denoted as $\mathbf{Co}^t_{s_2} = \{[1,1],[1,0],[2,1],[2,1]\}$.

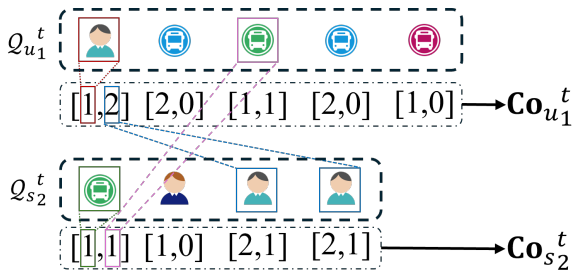

Figure 3. Example of co-occurrence features.

Then, we apply a function f to generate the temporally correlated representation by

$$\mathbf{X}^t_{*,Co} = f\left(\mathbf{Co}^{\mathbf{t}}_*[:,\mathbf{0}]\right) + f\left(\mathbf{Co}^{\mathbf{t}}_*[:,\mathbf{1}]\right), \tag{2}$$

where $*$ represents u or s.

Feature Fusion. After extracting the sequence information, we obtain the historical interactions that occurred before time t for both passengers and stations. These interactions can be used to represent the current node features. We further introduce learnable features $\mathbf{X}^U_N \in \mathbb{R}^{n\times d_N}, \mathbf{X}^S_N \in \mathbb{R}^{m\times d_N}$ for passenger nodes and station nodes to represent their inherent attributes, which will be gradually learned during the model training process. Hence, we denote $\mathbf{X}^t_{u,N}, \mathbf{X}^t_{s,N}$ as the learnable node feature sequences for passenger u's neighbors and station s's neighbors. Additionally, for historical interaction information e, the boarding and alighting attributes of passengers are important features for representing this interaction information. Therefore, we represent the fused features as follows:

$$\mathbf{Z}^t_* = Proj\left(\mathbf{X}^t_{*,N}||\mathbf{X}^t_{*,E}||\mathbf{X}^t_{*,T}||\mathbf{X}^t_{*,Co}\right), \tag{3}$$

where $*$ denotes u or s. $Proj(\cdot)$ represents the projection mapping function, which is implemented using a linear layer in this model. This is used to integrate the temporal feature information in the sequence, thereby generating the embedding $\mathbf{Z}^t_* \in \mathbb{R}^{N\times d_E}$ of the corresponding node at time t.

4.2. Dynamic Representation Learning

The passenger representations and station representations processed by the projection function have the same dimensions. This reflects the historical boarding and alighting behaviors of passengers, as well as their state (whether they are on or off the train) during the current interaction, over time. Additionally, we incorporate the inherent attributes of the station and the passenger flow changes over time. After obtaining the serialized representations, our goal is to use these representations to track changes in the boarding and alighting states of passengers.

Temporal Information Aggregation. To achieve this goal, we use an MLP block composed of stacked feedforward neural networks (FFN) and activation functions. Considering the issue of potential overfitting due to the small number of boarding and alighting events for a single passenger, we use dropout to address this problem. The processed representations $\mathbf{Z}^t_*$ are then passed through the stacked MLP modules. The entire process is as follows:

$$\mathbf{r}^l \sim \text{Bernoulli}(p), \tag{4}$$

$$\text{Dropout}(\mathbf{h}^l) = \hat{\mathbf{h}}^l = \mathbf{h}^l \odot \mathbf{r}^l, \tag{5}$$

$$\text{FFN}(\mathbf{Z}^{\mathbf{t}}_*, \mathbf{W}, \mathbf{b}) = \text{Dropout}(\text{ReLU}(\mathbf{Z}^{\mathbf{t}}_*\mathbf{W} + \mathbf{b})), \tag{6}$$

$$\mathbf{H}^t_* = \text{FFN}(\mathbf{Z}^{\mathbf{t}}_*, \mathbf{W}, \mathbf{b}). \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{d_E\times d_E}$, $\mathbf{b} \in \mathbb{R}^{d_E}$. After processing by the MLP model, the input features $\mathbf{Z}^{t,0}_u$, $\mathbf{Z}^{t,0}_s$ are transformed into output features $\mathbf{H}^t_u$ and $\mathbf{H}^t_s$. This transformed feature can be used for subsequent downstream prediction tasks.

Node Embedding Representation The representations of passenger nodes and station nodes at time t can be denoted as $\mathbf{h}^t_p$ and $\mathbf{h}^t_s$. These are obtained by averaging the representations $\mathbf{H}^t_*$ and then passing them through a linear output layer. The specific process is as follows:

$$\mathbf{h}^t_u = \left(\frac{1}{N}\sum_{k=1}^{N}\mathbf{H}^t_u[k,:]\right)\mathbf{W}_{out} + \mathbf{b}_{out}, \tag{8}$$

$$\mathbf{h}^t_s = \left(\frac{1}{N}\sum_{k=1}^{N}\mathbf{H}^t_s[k,:]\right)\mathbf{W}_{out} + \mathbf{b}_{out}. \tag{9}$$

where $\mathbf{W}_{out} \in \mathbb{R}^{d_E \times d_{out}}$ and $\mathbf{b}_{out} \in \mathbb{R}^{d_{out}}$ are trainable weights, and $\mathbb{R}^{d_{out}}$ represents the output embedding dimension.

4.3. Future Behavior Prediction

We consider the boarding or alighting relationship between passengers and stations as the creation of an edge in a dynamic graph. Therefore, the problem can be transformed into a link prediction problem in a dynamic graph. The representations of passengers and stations are fed into a fully connected network to calculate the probability of a link being formed between them at time t.

$$\mathbf{Z}^t = \mathbf{h}_u^t || \mathbf{h}_s^t, \tag{10}$$

$$p_{u,s}^t = (\mathrm{Relu}(\mathbf{Z}^t \mathbf{W}_1) + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2. \tag{11}$$

The dimension of $\mathbf{Z}$ is $2 \times d_{out}$, the dimension of $\mathbf{W}_1$ and $\mathbf{b}_1$ are $2 \times d_{out}$, while $\mathbf{W}_2 \in \mathbb{R}^1$, $\mathbf{b}_2 \in \mathbb{R}^1$. p_t represents the probability of a new link being formed between the passenger node and the station node at time t.

4.4. Parameter Optimization Algorithm

In our approach, the model training process begins with the construction of a neighbor sampler. For each training iteration, we first set a random seed to randomize the parameters and then load the model and start training. For each batch, we first calculate the most recent first-order neighbors for all passengers and stations in the batch. Next, we pad the extended first-order interaction sequences to generate historical interaction sequences of the same length. We then calculate the correlation of the sequences and use an MLP to capture the temporal characteristics between the sequences. Finally, we estimate the probability of creating new links between passengers and stations and update the parameters using the gradient descent method. The complete algorithm workflow is outlined in Algorithm 1.

Algorithm 1: Training process of DyGPP

Input : Collection of passenger $\mathcal{U}$ and collection of stations $\mathcal{S}$, a sequence of historical interaction events $\mathcal{G}^t = \{e_1, \ldots, e_k\}$ with $0 \leq t_1 \leq \ldots \leq t_k \leq t$, maximum number of training epochs $MaxEpoch$;

Output: The model parameters Θ after training;

Initialize the parameters in DyGPP with random weights Θ and set $Epochs \leftarrow 1$;
while *not converged and* $Epochs \leq MaxEpoch$ **do**
  for *batch* $B_b \in \mathcal{B}$ **do**
    $\mathcal{N}_{u_i}^t \leftarrow$ Sample latest one-hop neighbors of $u_i^t \in \mathcal{U}$;
    $\mathcal{N}_{s_i}^t \leftarrow$ Sample latest one-hop neighbors of $s_i^t \in \mathcal{S}$;
    $\mathcal{S}_{u,i}^t \leftarrow$ Create the interactions of passengers' neighbor set $\mathcal{N}_{u_i}^t$;
    $\mathcal{Q}_{s_i}^t \leftarrow$ Create the interactions of stations' neighbor set $\mathcal{U}_{s_i}^t$;
    $\mathcal{Q}_{u_i,pad}^t, \mathcal{Q}_{s,i,pad}^t \leftarrow$ Pad sequences to a regular size;
    $\mathbf{x}_{u_i,Co}^t, \mathbf{x}_{s_i,Co}^t \leftarrow$ Calculate correlations between $\mathcal{Q}_{u_i,pad}^t$ and $\mathcal{Q}_{s,i,pad}^t$;
    $\mathbf{x}_{*_i,N}^t, \mathbf{x}_{*_i,E}^t, \mathbf{x}_{*_i,T}^t \leftarrow$ Get raw feature embedding from the origin input;
    $\mathbf{X}_{*_i}^t \leftarrow$ Generate co-occurrence feature through projection layer;
    $\mathbf{Z}_{*,i}^t \leftarrow$ Encode passenger embedding and station embedding via MLP network;
    $\hat{p}_{u,s}^t \leftarrow$ Compute the probabilities by fusing sequence embedding $\mathbf{Z}_{u,i}^t, \mathbf{Z}_{s,i}^t$ Optimize the model parameters θ by back propagation;
  $Epochs \leftarrow Epochs + 1$;

Batch partitioning is an important method to enhance the parallel training capability of a model and shorten the training time. It should be noted that our approach to batch partitioning differs from commonly used methods. Traditional methods usually determine

the batch size based on the number of nodes in each batch, which often works well for time-insensitive data. In our model, the interactions between passengers and stations are strongly time-correlated. Factors such as commuting times and daily traffic flow cycles cause interactions to often concentrate during specific hours of the day. To ensure that the time intervals between data points in each batch are consistent, we adopted a method of partitioning batches based on time intervals.

Formally, given a dynamic graph $\mathcal{G}^t$, batches can be partitioned as $\mathcal{G}^t = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$, $\mathcal{B}_i = \{(u_{i,1}, s_{i,1}, l_{i,1}, t_{i,1}), \ldots, (u_{i,k}, s_{i,k}, l_{i,k}, t_{i,k}) \mid t_{i,1} \leq \ldots \leq t_{i,k}, t_{i,k} - t_{i,1} \leq \delta t\}$, where δt is a hyperparameter partitioned to represent the maximum time interval of interactions within a batch. This means that the length of each batch may vary. During periods of high passenger flow, a batch may contain a large amount of interaction information, while during periods of low passenger flow, a batch may contain very little interaction information. This method of batch partitioning ensures the stability of the number of batches across different data scales and provides good temporal information within batches. It also prevents the loss of temporal information, which could occur when multiple batches are within the same time period due to a large number of interactions.

5. Experiments

5.1. Dataset

In our experiments, we used two real-world datasets, both were collected from the Beijing subway system. The datasets contain all passenger entry and exit data from 1 June 2017 to 9 September 2017. The statistical information of the data is shown in Table 3.

Table 3. Statistics of all datasets.

Dataset	Passenger	Station	Interaction	Interaction Per Passenger
BJSubway-40K	459	392	44,852	97
BJSubway-1M	7491	392	958,076	127

We analyzed the distribution of interaction frequencies between different passengers and stations in the two datasets, as shown in Figure 4. The results indicated that most passengers interacted with stations (including boarding and alighting) fewer than 200 times. In the 40K dataset, the median was 54, the Q1 (first quartile) was 30, and the Q3 (third quartile) was 127. In the 1M dataset, the median was 72, the Q1 was 34, and the Q3 was 202.

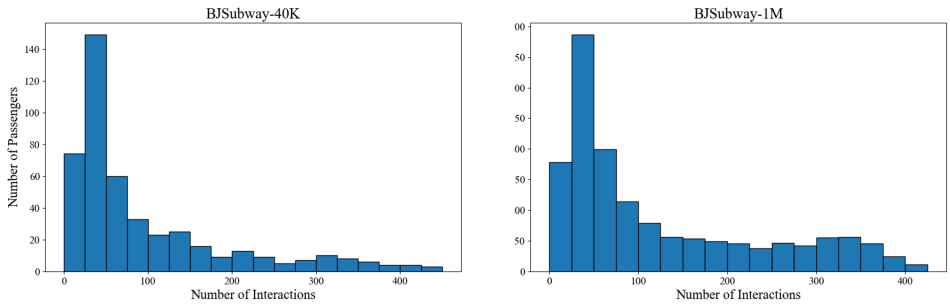

Figure 4. Distribution of Interaction Frequencies Between Passengers and Stations.

5.2. Baselines

In our experiments, we compared our model with seven other commonly used models. These models encompass a variety of techniques and implementation methods, including statistical methods (i.e., TOP and Personal TOP), sequence model methods (i.e., LSTM and GRU), graph convolutional methods (i.e., TGAT), memory network models (i.e., TGN), and MLP-based methods (i.e., GraphMixer).

TOP. The TOP method [33] is a statistical method based on the travel patterns of all users. First, the training data are fed into the model for statistical analysis. Then, for each

station, two sequences are generated: one for all alighting stations when the station is the boarding station, and one for all boarding stations when the station is the alighting station. After completing the statistics, the corresponding station is determined based on the passenger's last travel status, and the station with the highest occurrence frequency in the corresponding sequence is selected as the predicted station.

Personal TOP. The personal TOP [33] method is similar to the TOP method but generates unique sequences for each specific passenger for boarding and alighting stations. During testing, for passengers with existing sequence records, the predicted station is directly generated based on their unique sequences. For passengers without sequence records, the predicted station is determined based on the overall sequence.

LSTM. LSTM [34] is a commonly used method for processing sequential data. Compared to traditional RNN models, it introduces input gates, forget gates, output gates, and a cell state, which help prevent the problems of vanishing or exploding gradients. This allows LSTM to better handle long-term dependencies in sequences.

GRU. GRU [35] can be seen as a variant of LSTM. It uses a reset gate to determine how to combine new input information with previous memory, and an update gate to define the amount of previous memory to retain for the current time step. GRU is capable of preserving information from long sequences, without clearing it over time or removing it if it is unrelated to the prediction.

TGAT. TGAT [23] employs a self-attention mechanism to generate corresponding representations by aggregating the temporal topological features of neighbors for each passenger node and station node. Additionally, it uses a time encoder to capture temporal characteristics.

TGN. TGN [18] maintains an updated memory node for each passenger node and station node and updates its memory when the node is observed. This is achieved through interaction with a message function, a message aggregator, and a memory updater. This model generates temporal embeddings for both passenger nodes and station nodes.

GraphMixer GraphMixer [32] demonstrates that a fixed-time encoding function outperforms the trainable version. It incorporates this fixed function into a link encoder based on MLP-Mixer to learn from temporal links. A node encoder using neighbor mean-pooling is implemented to summarize node features.

5.3. Evaluation Metrics

Regarding evaluation metrics, we used the average precision (AP) and area under the receiver operating characteristic curve (AUC) to assess the performance of all methods in predicting binary future connections between passengers and stations. AP measures the proportion of correct predictions among all responses, while AUC reflects the model's ability to distinguish between positive and negative responses.

$$\mathrm{AP} = \frac{1}{n}\sum_{k=1}^{n}\frac{\mathrm{TP}_k}{\mathrm{TP}_k + \mathrm{FP}_k}, \tag{12}$$

where TP_k represents the cases in the k-th prediction where the predicted value matches the true value and is positive (i.e., true positive); FP represents the cases in the k-th prediction where the predicted value is positive but the true value is negative (i.e., false positive).

Additionally, we introduced an inductive sampling method to test the accuracy of predicting interactions with stations that passengers have never encountered before. This sampling method involved hiding the station nodes to be used for inductive testing during the initial training phase, and then providing these test station nodes during the testing phase. The proportions used for the training, validation, and test sets were 0.7, 0.15, and 0.15, respectively.

5.4. Implementation Details

To ensure consistency and effectiveness in the comparison, the parameter settings for the baseline models were as shown in Table 4. We applied the Adam optimizer, which uses the Adam algorithm for first-order gradient-based optimization of stochastic objective functions [28], and set the early stopping patience to 20. For all datasets, we set the maximum time interval in a batch to 1000. For the TOP and personal TOP methods, we constructed an overall station counter and a personal station counter, respectively, to calculate the probability of interactions between passengers and stations. In the LSTM and GRU methods, we input all station interaction records of the passenger to be predicted as a sequence, divide the batch according to time intervals for training, and finally obtained the interaction probability with the target station. In graph-based methods (i.e., TGAT, TGN, GraphMixer, and DyGPP), we first constructed a dynamic graph of all user–station interactions. Then, we divided the node change sequences according to the set time intervals and finally proceeded with the subsequent tasks. For all models, the dimensions of the passenger nodes and station nodes were set to 172, and the dimensions of the time encoding nodes were set to 100. The other model settings were consistent with those described in the original articles. The experiments were conducted on an Ubuntu machine featuring two Intel(R) Xeon(R) Gold 6130 CPUs @ 2.10GHz with 16 physical cores. The GPU device utilized was an NVIDIA Tesla T4 with 15GB memory.

Table 4. Parameters of the experiment.

Parameter	Value
Time gap for batch	1000 s
Number of neighbors	20
Neighbor sampling strategy	Most recent neighbor
Negative sample strategy	Random
Time scaling factor	0.000001
Number of heads	2
Number of encoder model layers	1
Channel embedding dimension	50
Learning rate	0.0001
Max input sequence	32

5.5. Performance Comparison

Table 5 presents the prediction results of whether a passenger interacted with the next station in the two datasets, compared with seven other methods. The results shown in the table are averages obtained after five independent training rounds. The best results are highlighted in bold, and the second-best results are marked with an underline.

Table 5. AP and AUC for passenger behavior prediction.

Datasets	BJSubway-40K		BJSubway-1M	
Metrics	**AP**	**AUC**	**AP**	**AUC**
TOP	0.7880 ± 0.0000	0.7831 ± 0.0000	0.8501 ± 0.0000	0.8204 ± 0.0000
Personal TOP	0.7545 ± 0.0000	0.7701 ± 0.0000	0.7901 ± 0.0000	0.8067 ± 0.0000
LSTM	0.9026 ± 0.0039	0.8752 ± 0.0041	0.9395 ± 0.0015	0.9336 ± 0.0015
GRU	0.9119 ± 0.0059	0.8860 ± 0.0062	0.9418 ± 0.0016	0.9357 ± 0.0018
TGN	0.9390 ± 0.0032	0.9215 ± 0.0040	0.9430 ± 0.0017	0.9365 ± 0.0023
GraphMixer	0.8565 ± 0.0078	0.8281 ± 0.0079	0.7591 ± 0.0015	0.7716 ± 0.0018
TGAT	0.7094 ± 0.0095	0.6410 ± 0.0116	0.6563 ± 0.0107	0.6420 ± 0.0180
DyGPP	**0.9699 ± 0.0008**	**0.9591 ± 0.0015**	**0.9697 ± 0.0002**	**0.9647 ± 0.0002**

The DyGPP model achieved the highest accuracy and the best AUC score among all models. This indicates that dynamic graph models can efficiently handle passenger behavior prediction tasks using dynamic graph structures, and these structures can be

effectively extended to other sequential models. Other key observations from the table are as follows:

(i) Statistical methods (i.e., TOP and personal TOP), performed poorly in comparison. The statistical methods only considered the historical boarding and alighting counts of passengers, without analyzing specific times, and they did not account for the temporal sequence and potential changes in passenger travel. Compared to the personal TOP method, the TOP method performed better because most individuals had fewer than 100 interactions, making it difficult to effectively represent individual travel patterns given the significantly larger number of stations compared to the number of interactions.

(ii) The sequential methods (i.e., LSTM and GRU) performed well in the prediction task. These methods predicted future passenger behavior by combining boarding and alighting patterns, fully learning the travel habits of passengers. However, the sequential methods did not consider the impact of station conditions on passenger travel choices and had insufficient predictive ability for travel behavior fluctuations caused by small-scale special circumstances.

(iii) The dynamic graph methods considered the relationships between nodes, thereby accounting for the mutual influence between stations and passengers, which enhanced the prediction accuracy. The TGAT method used interaction occurrence times as an important basis for learning, but due to the limited amount of available data and the coexistence of long-term periodic patterns and short-term abrupt events in passenger travel behavior, TGAT's learning capability was insufficient. The TGN transmitted interaction information between nodes through message aggregation and stored it in the node's memory, effectively learning the features of passengers and stations, thus performing the best among the baselines. However, the dynamic graph methods did not effectively utilize the historical interaction information between passengers and stations, and their temporal characteristics were lost due to the constraints of the graph structure during message passing.

(iv) The DyGPP model performed the best among all the results. DyGPP used serialized historical interaction information to capture the temporal characteristics of user travel habits and employed learnable node features to depict the attributes of passengers and stations. Building on this foundation, DyGPP integrated a time encoder to incorporate the temporal features of interactions, further enhancing the accuracy of the passenger behavior prediction task.

5.6. Ablation Study

An ablation study validated the effectiveness of various parts of the model by systematically removing specific modules and analyzing the impact on the overall performance [36]. We designed ablation experiments to further verify the effectiveness of each module in DyGPP. This experiment included examining the effectiveness of the edge encoder (eE), the time encoder (tE), the neighbor co-occurrence encoder (coE), the self-attention part of the encoder (cosE), and the cross-attention part (cocE). We removed each module separately, and refer to these as w/o eE, w/o tE, w/o coE, w/o cosE, and w/o cocE. We evaluated the prediction performance of the different variants on the BJSubway-40K dataset.

Table 6 shows that DyGPP exhibited the best performance. The following conclusions were confirmed in the experiments: (1) Temporal information played a significant role in the passenger behavior prediction task, as evidenced by the comparison with the w/o tE model. This demonstrated DyGPP's ability to track the temporal information of passenger boarding and alighting. (2) The boarding and alighting attributes of passengers proved to be important, which aligns with common sense, as a passenger's boarding behavior is always paired with their alighting behavior. (3) The co-occurrence module we designed also showed a significant impact on performance. The self-attention part of the co-occurrence module represents the passenger's preference for different stations over a period of time, while the cross-attention part directly represents the association between passengers and the stations they attempt to interact with.

Table 6. Effects of different components in DyGPP.

Methods	AP	AUC
w/o eE	0.9681 ± 0.0005	0.9562 ± 0.0005
w/o tE	0.9685 ± 0.0010	0.9570 ± 0.0016
w/o coE	0.8877 ± 0.0072	0.8601 ± 0.0085
w/o cosE	0.9672 ± 0.0014	0.9543 ± 0.0022
w/o cocE	0.8824 ± 0.0114	0.8533 ± 0.0135
DyGPP	**0.9699 ± 0.0008**	**0.9591 ± 0.0015**

5.7. Temporal Encoder Analysis

We further conducted experiments to verify the effectiveness of the MLP temporal encoder, and we replaced this encoder with Transformer, LSTM, and Mamba, which are common sequence modeling methods. We evaluated the runtime of the different model variants on the BJSubway-40K dataset, and the results are shown in Table 7. The time represents the total time taken for the model to run (including training, validation, and testing).

In the tests for the temporal information encoder, Mamba included a more complex structure and more parameters; LSTM required step-by-step calculations when processing sequence data, making it difficult to parallelize; Transformer achieved more efficient parallel computation through the self-attention mechanism but had higher theoretical complexity; while MLP, with its simple structure, allowed for fully parallel computation. The AP and AUC results show that MLP performed better than the other encoders, demonstrating the effectiveness of using MLP for temporal learning.

Table 7. Performance on different temporal encoders.

Methods	AP	AUC	Time Cost (s)
Transformer	0.9692 ± 0.0006	0.9578 ± 0.0010	4707.03
LSTM	0.9693 ± 0.0009	0.9580 ± 0.0018	4008.01
DyGPP-mamba	0.9690 ± 0.0010	0.9581 ± 0.0015	5641.16
DyGPP	**0.9699 ± 0.0008**	**0.9591 ± 0.0015**	**3604.43**

5.8. Parameter Sensitivity

The historical interaction sequences included the attributes of the nodes themselves and the related information from their neighbors. These sequences had a significant impact on the subgraph construction and node feature representation. Therefore, the length of the interaction sequence was a key factor in determining the prediction accuracy. We tested the parameter sensitivity of DyGPP's prediction results under different sequence lengths in the BJSubway-40K dataset, and the results are depicted in Figure 5. As seen from the curve, the accuracy tended to improve with increasing sequence length and gradually stabilized. This is because longer sequences provided more information for tracking passenger and station interactions. However, excessively long interaction sequences exceeded the maximum interaction value for passengers, resulting in no further increase in accuracy once a threshold had been reached.

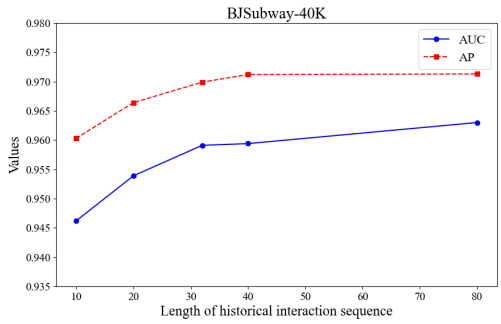

Figure 5. Sequence length sensitivity in predicting.

6. Discussion

In this study, we used the DyGPP model to conduct an in-depth analysis of passenger travel behavior patterns. The results showed that DyGPP outperformed all existing methods in predicting the next station for passengers. This outcome indicates that comprehensively considering the historical interaction information between passengers and stations is crucial in PP tasks. The DyGPP model significantly improved the prediction accuracy of PP tasks, while maintaining a relatively simple architecture and achieving efficient predictions in a short time. This lays a solid foundation for the model's application in real-world production.

Experimental analysis revealed that the neighbor co-occurrence encoder significantly impacted the model's prediction accuracy, with the cross-attention part playing a critical role. Since the model uses first-order neighbors as the basis for sequence construction, and the first-order neighbors of a station are typically passengers, a small amount of non-repetitive passenger information may be insufficient to fully represent the characteristics of a station when the sequence length is short. Additionally, although there are often spatial and logical relationships between stations, this study treated stations as independent nodes for prediction and judgment. This simplification may have affected the prediction accuracy.

Given the limitations of the experiments, future research should focus on two main aspects: first, for station feature extraction, future studies should consider using second-order neighbors to represent stations. Since the second-order neighbors of a station are still stations, this approach can better capture the logical similarities between stations. Second, the integration of spatial location information (including the geographical location of stations and the public transportation routes passing through them) should be considered to represent station features. By doing so, the model can better learn the implicit relationships between stations during training, further improving the prediction accuracy.

The model proposed in this study not only addresses the current issue of traffic management departments relying solely on station flow rate and volume for predictions but also enhances data utilization efficiency by incorporating individual passenger characteristics to more accurately predict future station passenger flow. This method can identify differences among various passenger groups, which can help improve a station's early warning and risk response capabilities in emergencies or special incidents.

7. Conclusions

This paper proposed a passenger behavior prediction method based on dynamic graph representation learning. Our approach considers both the long-term and short-term patterns of passenger behavior, defining the passenger behavior prediction problem as a connection prediction problem between heterogeneous nodes in a continuous-time dynamic graph. We modeled the passenger's historical interaction patterns using fixed-length sequences, ensuring robust performance in the face of continuously growing interaction records in the real world. To capture the different behaviors over long and short periods, we employed a sequential representation method to model historical interaction records and used an MLP model to learn passenger behavior characteristics. Finally, to validate the

effectiveness of our method, we conducted statistical analyses on datasets and experiments on the Beijing subway dataset. The experiments not only demonstrated the excellent performance of DyGPP but also highlighted its effective behavior prediction capability within dynamic graph models.

Author Contributions: Conceptualization, M.X.; Methodology, M.X.; Validation, M.X.; Formal analysis, M.X.; Investigation, M.X.; Resources, M.X.; Data curation, M.X.; Writing—original draft, M.X.; Writing—review & editing, M.X., T.Z., J.Y. and B.D.; Supervision, B.D. and R.H.; Funding acquisition, B.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 51991395, 51991391, U1811463 and the S&T Program of Hebei grant number 225A0802D.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to authorization requirements from the data source.

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

References

1. Jia, R.; Li, Z.; Xia, Y.; Zhu, J.; Ma, N.; Chai, H.; Liu, Z. Urban road traffic condition forecasting based on sparse ride-hailing service data. *IET Intell. Transp. Syst.* **2020**, *14*, 668–674. [CrossRef]
2. Tsai, T.H.; Lee, C.K.; Wei, C.H. Neural network based temporal feature models for short-term railway passenger demand forecasting. *Expert Syst. Appl.* **2009**, *36*, 3728–3736. [CrossRef]
3. Liu, L.; Chen, R.C.; Zhu, S. Impacts of weather on short-term metro passenger flow forecasting using a deep LSTM neural network. *Appl. Sci.* **2020**, *10*, 2962. [CrossRef]
4. Lijuan, Y.; Zhang, S.; Guocai, L. Neural Network-Based Passenger Flow Prediction: Take a Campus for Example. In Proceedings of the 2020 13th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 12–13 December 2020; pp. 384–387.
5. Tang, L.; Zhao, Y.; Cabrera, J.; Ma, J.; Tsui, K.L. Forecasting Short-Term Passenger Flow: An Empirical Study on Shenzhen Metro. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3613–3622. [CrossRef]
6. Gu, J.; Jiang, Z.; Fan, W.; Chen, J. Short-term trajectory prediction for individual metro passengers integrating diverse mobility patterns with adaptive location-awareness. *Inform. Sci.* **2022**, *599*, 25–43. [CrossRef]
7. McFadden, D. The measurement of urban travel demand. *J. Public Econ.* **1974**, *3*, 303–328. [CrossRef]
8. Wei, Y.; Chen, M.C. Forecasting the short-term metro passenger flow with empirical mode decomposition and neural networks. *Transp. Res. Part Emerg. Technol.* **2012**, *21*, 148–162. [CrossRef]
9. Menon, A.K.; Lee, Y. Predicting short-term public transport demand via inhomogeneous Poisson processes. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 2207–2210.
10. Zhai, H.; Tian, R.; Cui, L.; Xu, X.; Zhang, W. A Novel Hierarchical Hybrid Model for Short-Term Bus Passenger Flow Forecasting. *J. Adv. Transp.* **2020**, *2020*, 7917353. [CrossRef]
11. Hao, S.; Lee, D.H.; Zhao, D. Sequence to sequence learning with attention mechanism for short-term passenger flow prediction in large-scale metro system. *Transp. Res. Part Emerg. Technol.* **2019**, *107*, 287–300. [CrossRef]
12. Barros, C.D.; Mendonça, M.R.; Vieira, A.B.; Ziviani, A. A survey on embedding dynamic graphs. *ACM Comput. Surv. (CSUR)* **2021**, *55*, 1–37. [CrossRef]
13. Kazemi, S.M.; Goel, R.; Jain, K.; Kobyzev, I.; Sethi, A.; Forsyth, P.; Poupart, P. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.* **2020**, *21*, 1–73.
14. You, J.; Du, T.; Leskovec, J. ROLAND: Graph learning framework for dynamic graphs. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 2358–2366.
15. Zhang, K.; Cao, Q.; Fang, G.; Xu, B.; Zou, H.; Shen, H.; Cheng, X. Dyted: Disentangled representation learning for discrete-time dynamic graph. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 3309–3320.
16. Nguyen, G.H.; Lee, J.B.; Rossi, R.A.; Ahmed, N.K.; Koh, E.; Kim, S. Dynamic network embeddings: From random walks to temporal random walks. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 1085–1092.
17. Yu, W.; Cheng, W.; Aggarwal, C.C.; Zhang, K.; Chen, H.; Wang, W. Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2018; pp. 2672–2681.
18. Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; Bronstein, M. Temporal graph networks for deep learning on dynamic graphs. *arXiv* **2020**, arXiv:2006.10637.

19. Kumar, S.; Zhang, X.; Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1269–1278.
20. Tolstikhin, I.O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 24261–24272.
21. Wang, L.; Chang, X.; Li, S.; Chu, Y.; Li, H.; Zhang, W.; He, X.; Song, L.; Zhou, J.; Yang, H. Tcl: Transformer-based dynamic graph modelling via contrastive learning. *arXiv* **2021**, arXiv:2105.07944.
22. Yu, L.; Sun, L.; Du, B.; Lv, W. Towards better dynamic graph learning: New architecture and unified library. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 67686–67700.
23. Xu, D.; Ruan, C.; Korpeoglu, E.; Kumar, S.; Achan, K. Inductive representation learning on temporal graphs. *arXiv* **2020**, arXiv:2002.07962.
24. Poursafaei, F.; Huang, S.; Pelrine, K.; Rabbany, R. Towards better evaluation for dynamic link prediction. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 32928–32941.
25. Ruder, S. An overview of gradient descent optimization algorithms. *arXiv* **2016**, arXiv:1609.04747.
26. Darken, C.; Chang, J.; Moody, J. Learning rate schedules for faster stochastic gradient search. In Proceedings of the Neural Networks for Signal Processing, Helsingoer, Denmark, 31 August–2 September 1992; Volume 2, pp. 3–12.
27. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
28. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
29. Lampert, M.; Blöcker, C.; Scholtes, I. From Link Prediction to Forecasting: Information Loss in Batch-based Temporal Graph Learning. *arXiv* **2024**, arXiv:2406.04897.
30. Cheng, Z.; Trépanier, M.; Sun, L. Incorporating travel behavior regularity into passenger flow forecasting. *Transp. Res. Part Emerg. Technol.* **2021**, *128*, 103200. [CrossRef]
31. Cantelmo, G.; Qurashi, M.; Prakash, A.A.; Antoniou, C.; Viti, F. Incorporating trip chaining within online demand estimation. *Transp. Res. Procedia* **2019**, *38*, 462–481. [CrossRef]
32. Cong, W.; Zhang, S.; Kang, J.; Yuan, B.; Wu, H.; Zhou, X.; Tong, H.; Mahdavi, M. Do we really need complicated model architectures for temporal networks? *arXiv* **2023**, arXiv:2302.11636.
33. Karypis, G. Evaluation of Item-Based Top-N Recommendation Algorithms. In Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, GO, USA, 5–10 November 2001; pp. 247–254.
34. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
35. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
36. Meyes, R.; Lu, M.; de Puiseau, C.W.; Meisen, T. Ablation studies in artificial neural networks. *arXiv* **2019**, arXiv:1901.08644.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.