Dynamic Graph Learning Based on Hierarchical Memory for Origin-Destination Demand Prediction

Ruixing Zhang, **Liangzhe Han**, **Boyi Liu**, **Jiayuan Zeng** and **Leilei Sun**[*]
SKLSDE Lab, Beihang University, Beijing 100191, China
{yyxzhj, liangzhehan, 1906boy, yuaner, leileisun}@buaa.edu.cn

Abstract

Recent years have witnessed a rapid growth of applying deep spatiotemporal methods in traffic forecasting. However, the prediction of origin-destination (OD) demands is still a challenging problem since the number of OD pairs is usually quadratic to the number of stations. In this case, most of the existing spatiotemporal methods fail to handle spatial relations on such a large scale. To address this problem, this paper provides a dynamic graph representation learning framework for OD demands prediction. In particular, a hierarchical memory updater is first proposed to maintain a time-aware representation for each node, and the representations are updated according to the most recently observed OD trips in continuous-time and multiple discrete-time ways. Second, a spatiotemporal propagation mechanism is provided to aggregate representations of neighbor nodes along a random spatiotemporal route which treats origin and destination as two different semantic entities. Last, an objective function is designed to derive the future OD demands according to the most recent node representations, and also to tackle the data sparsity problem in OD prediction. Extensive experiments have been conducted on two real-world datasets, and the experimental results demonstrate the superiority of the proposed method. The code and data are available at https://github.com/Rising0321/HMOD.

1 Introduction

In the past decade, with the rapid growth of traffic infrastructure, an extensive amount of high-quality traffic data is collected for the industry and the research community. This opportunity promotes multiple promising applications of Intelligent Transportation System (ITS) including traffic time estimation [Han *et al.*, 2021], traffic trajectory prediction [Ma *et al.*, 2019], and traffic demand prediction [Yao *et al.*, 2018].

Almost all the traffic applications require a meaningful and successful representation for traffic nodes (e.g., road segments, metro stations), which involves spatial dependency among nodes and temporal dependency across time. Among these applications, traffic demand prediction, which has a significant influence on transportation planning, is a vital and attractive domain in ITS. Especially, the node-to-node OD demand prediction attracts more and more researchers in recent years since it can not only provide the number of passengers, but it can also show which target places passengers tend to choose, which implies a potential to better model human mobility and further benefit the whole system.

To solve the problems, researchers have proposed a variety of models based on cutting-edge deep learning methods, such as CNNs [Noursalehi *et al.*, 2021], RNNs [Zhang *et al.*, 2021] and Transformers [Sankar *et al.*, 2020]. GNNs, as their potential to learn high-quality representation on graph, are utilized in OD demand prediction to mine the complex spatial dependencies, which has achieved considerable success [Wang *et al.*, 2019]; [Shi *et al.*, 2020]; [Wang *et al.*, 2021].

Despite all the efforts, two important issues are rarely discussed: First, previous methods tend to divide the dynamic trips into multiple regular-spaced time slices, using OD matrix at each time slice to represent the demand during the corresponding time(e.g. [Zhang *et al.*, 2021]). This process makes it easy to fit existing recurrent deep learning methods, but this data compression replaces raw time information with discrete-time slices and misses useful information. Second, previous methods consider node relations directly and simply. For example, some researchers view two nodes are closely related if there are a great number of passengers from one to another in the last period like [Shi *et al.*, 2020]. However, they fail to distinguish between origins and destinations, which neglects relations between origin nodes that have similar demands.

This paper aims to address the above two issues and proposes a dynamic node representation framework for OD demand prediction. However, it is a nontrivial endeavor to design such a framework due to the following challenges: First, the time granularity about how to organize the historical OD demand matrix is hard to choose. Choosing too coarse time granularity will result in an inability to sense useful information such as the trend while choosing too fine time granularity will lead to substantial noise. How to handle the complex temporal dynamics of trips is fundamental to node representations. Second, relations between nodes are time-evolving.

[*]Contact Author

Demand from two nodes to other nodes could be similar on weekdays and completely different on weekends. How to dynamically capture these relations is crucial to node representations. Third, the model is required to predict demand values on every pair of nodes. Due to demand imbalance, many OD pairs have no demand at a certain time. How to properly handle these pairs is also challenging to stable training.

In this paper, we propose a Hierarchical Memory dynamic graph representation learning framework for OD Prediction (HMOD). First, a hierarchical memory updater is designed to integrate the continuous-time information and multi-level discrete-time information. This updater avoids limits of time information loss in previous discrete-time methods. Meanwhile, it constantly updates memory for dynamic node representation by message passing scheme and a message fusion mechanism is provided to integrate hierarchical memories. Second, a spatiotemporal propagation module is designed to aggregate neighbor node representations. This module is based on real-time demand and treats origins and destinations as two different semantic entities. It can effectively exploit similar relations between origins instead of a mix of origins and destinations. Third, an output layer and a specially designed objective function are deployed to predict the OD matrix. The objective function relaxes the error of no-demand pairs to alleviate the influence of data imbalance.

The contributions of this work are summarized as follows:

- We design hierarchical memories to integrate discrete-time information and continuous-time information of OD demand. To the best of our knowledge, it is the first time that traffic node representation learning is extended to the continuous-time dynamic graph view.
- We propose an origin-destination embedding module to aggregate neighbor information along conditioned random walks. The module views origins and destinations as different semantic entities and can thus distinguish them to avoid mixing two types of information up.
- Extensive experiments are conducted on two real-world traffic datasets to evaluate the performance of the proposed framework and key components. The results demonstrate that our framework significantly outperforms other state-of-the-art methods.

2 Related Work

2.1 OD Demand Prediction

With extensive trip data are recorded by traffic cards and APPs , researchers are attracted to solve a more fine-grained traffic demand prediction, OD demand prediction. A simple solution is directly taking OD matrix as a two-dimensional image and applying convolution-based or spectral-based methods on it, such as [Noursalehi *et al.*, 2021]. This kind of method fails to capture the spatiotemporal dependencies because they are limited to consider further relationship and topology, which is complex but common in traffic. Along with the development of graph neural networks, some researchers view stations or zones as nodes in the graph and design a variety of frameworks to address the problem of learning spatiotemporal dependencies. GEML [Wang *et al.*, 2019] uses GNN and LSTM to perform grid representation and capture temporal patterns. DNEAT [Zhang *et al.*, 2021] design a spatiotemporal attention network and exploit different time granularity to mine complex temporal patterns. Although amazing results have been achieved by these methods, they still face the problem of missing fine-grained patterns since they are all based on discrete-time snapshots to update the representation.

2.2 Dynamic Graph Representation Learning

Dynamic graph representation learning is a technique to maintain evolving node states with time going on, while traditional graph representation learning frameworks generate static representations ignoring the dynamic changing content. For example, GraphSAGE [Hamilton *et al.*, 2017] and GAT [Veličković *et al.*, 2017] train an unchanged embedding without considering inserted or removed edges.

Recently, Dynamic Graph Representation Learning methods are proposed to address the above issues. Generally, the methods can be divided into two categories. The first category is called Discrete-Time Dynamic Graph (DTDG). The DTDG methods firstly define a length τ and then will update the embeddings every τ unit time. DynamicTraid [Zhou *et al.*, 2018] and tNodeEmbed [Singer *et al.*, 2019] are the representative ones. Although previously mentioned OD prediction methods are all able to model evolving patterns of node representations, it faces severe problems to sense fine-grained information. On the contrary, the second category, named Continuous-Time Dynamic Graph (CTDG), updates node representations when an event happens. It is a more natural way to update the embeddings since raw events come in sequence instead of coming as snapshots. The typical ones are TGN [Rossi *et al.*, 2020], DyRep [Trivedi *et al.*, 2019], JODIE [Kumar *et al.*, 2019]. However, these methods could only capture temporal dependencies of limited time steps.

3 Preliminaries

The trip dynamic graph is formulated as $\mathcal{G} = (V,\ E)$, where $V = \{v_1, v_2, \cdots, v_N\}$ is the set of N nodes and $E = \{e_1, e_2, \ldots, e_M\}$ is the edge set of M edges. These nodes represent stations or regions in the real world. Each edge $e_m = (u_m, v_m, t_m, \mathbf{f_m})$ represents a passenger departed at t_m from u_m to v_m with associated feature vector $\mathbf{f_m} \in \mathbb{R}^{d_F}$. The feature vector contains travel distance, weather, payment, and tips. At a specific time point t, the dynamic graph can be represent as $\mathcal{G}_t = (V, \{e_m | t_m < t\})$, which contains trips happened before t. The OD demand matrix during time interval from t to $t+\tau$ is denoted is $\mathbf{Y}_{t:t+\tau} \in \mathbb{R}^{N*N}$ whose (i,j)-entry represents the volume from v_i to v_j during this time interval. The purpose of OD demand prediction is to predict future OD matrix $\mathbf{Y}_{t,t+\tau}$ given historical trip data as a dynamic graph $\mathcal{G}_t$.

Problem 1. *Given the dynamic graph $\mathcal{G}_t$ at timestamp t, it is expected to learn a function Ψ to predict the OD Matrix $\hat{\mathbf{Y}}_{t,t+\tau} = \Psi(\mathcal{G}_t)$, i.e.,*

$$\Psi^* = \arg\min_{\Psi} \mathrm{ODLoss}(\mathbf{Y}_{t,t+\tau}, \Psi(\mathcal{G}_t)), \tag{1}$$

where ODLoss will be given in the next section.

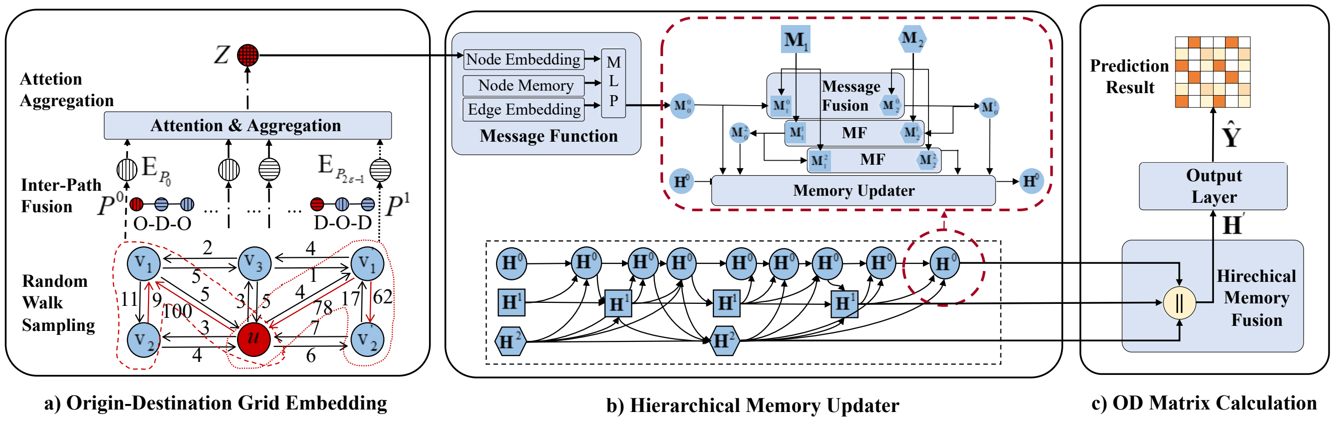

Figure 1: The Overall Framework of HMOD.

4 Methodology

In this section, we will elaborate details of **HMOD**. The overall architecture is shown in Figure 1. First, we will introduce the hierarchical memory updater which maintains the node representations and dynamically update them based on message passing, which can preserve hierarchical time semantic information for nodes. Next, to generate high-quality messages for memory update, an Origin-Destination Node Embedding method is designed to aggregate neighbor information by spatiotemporal random walks. In the last part, the output layer and specially designed objective function are introduced for the final prediction and training.

4.1 Hierarchical Memory Updater

Historical trips, which are original edges with continuous timestamps, reflect OD demand and are fundamental to node representation. Previous methods like GEML generally count edges in a fixed time window and make its time discrete, which discards practical time information. The more coarse time granularity is to generate time windows, the more information is lost. Unlike previous methods, we propose to maintain hierarchical-time memories for nodes by a hierarchical memory updater to address this problem. On the one hand, a part of it views time as continuous features and is updated when an edge comes. On the other hand, another part maintains multiple discrete-time memories to reveal a macro node state. Further, a memory fusion mechanism is designed to mix each level's memory while keeping mainly receiving the corresponding level message. Formally, $D+1$ memories $\mathbf{H} \in \mathbb{R}^{(D+1)\times d_H} = \{\mathbf{H}^0, \mathbf{H}^1, \mathbf{H}^2, \cdots, \mathbf{H}^{\mathrm{D}}\}$ are maintained on each node, where $\mathbf{H}^0$ represents continuous-time memory and $\mathbf{H}^{\mathrm{d}}(d \geq 1)$ represents macro discrete-time memory covering $\Delta\mathrm{T_d}$ time unit. Their updating methods are as follows.

Message Function

The above hierarchical memory aims to always maintain updated node states. When a memory updating process is triggered, that is to say for continuous-time memories there is a new edge coming and for discrete-time memories, $\Delta\mathrm{T_d}$ time unit passed by, we need to calculate a message to update node memories according to information triggered this update. For continuous-time memories, the message contains the last updated node states, encoding of the time gap from the last update, and aggregated neighbor information which will be discussed in 4.2. For discrete-time memories, the message contains the last updated node states, OD matrix during $\Delta\mathrm{T_d}$, and aggregated neighbor information. For the specefic node i, the above message generation process can be formulated as:

$$\mathbf{M}' = \begin{cases} [\mathbf{H}^d || \Theta(\mathbf{H}^{\mathrm{d}}, \mathcal{G}_{\mathrm{t}}) || \Omega_{\mathrm{d}}(\mathbf{H}^{\mathrm{d}}, \mathcal{G}_{\mathrm{t}})], d = 0 \\ [\mathbf{H}^d || \mathbf{Y}^{\mathrm{i}}_{\mathrm{t}-2^{\mathrm{d}-1}\Delta\mathrm{T,t}} || \Omega_{\mathrm{d}}(\mathbf{H}^{\mathrm{d}}, \mathcal{G}_{\mathrm{t}})], d \neq 0 \end{cases}, \quad (2)$$

$$\mathbf{M}^0_d = \mathbf{W}^{m_1}\sigma(\mathbf{W}^{m_2}\mathbf{M}' + \mathbf{b}^{m_2}) + \mathbf{b}^{m_1}, \quad (3)$$

$$\Theta(\mathbf{H}^d, \mathcal{G}_t) = (1 + \mathbf{W}^e(t - \mathbf{t}^-_i) + \mathbf{b}^e) \odot \mathbf{H}^d, \quad (4)$$

where $\mathbf{M}^0_d \in \mathbb{R}^{d_M}$ represents the message described above, $\odot$ represents element-wise product, $\mathbf{t}^-_i$ represents last update time of node i, $\mathbf{Y}^{\mathrm{i}}_{\mathrm{t}-2^{\mathrm{d}-1}\Delta\mathrm{T,t}}$ means the i-th row of the OD matrix. $\mathbf{W}^*$ and $\mathbf{b}^*$ are learnable parameters. $\Theta(\mathbf{H}^d, \mathbf{G})$ represents the time encoding, which is utilized to sensing different time span's different effect. $\Omega_d(\mathbf{H}^d, \mathbf{G})$ represents the node embedding method. σ represents the ReLU function.

Message Fusion

The message directly computed by the previously mentioned message function only contains information on a specific track, i.e. continuous information or discrete information of a specific granularity. However, the continuous message may lack certain global spatiotemporal information while the fine-grained spatiotemporal information may be absent in discrete messages. However, how to integrate other time granularity information while preserving own message is a challenging problem. To this end, a message fusion mechanism is proposed to integrate the specific track message with others to integrate multiple types of spatiotemporal information.

Formally, the latest message $\mathbf{M} \in \mathbb{R}^{(L+1)\times d_M} = \{\mathbf{M}^0, \mathbf{M}^1, \cdots, \mathbf{M}^L\}$ is stored with corresponding memory, where $\mathbf{M}^0$ is the output of message function and $\mathbf{M}^l (l \geq 0)$ is computed as follows. When computing $\mathbf{M}^l$, other messages from other time level is first aggregated then concatenated with $\mathbf{M}^{l-1}$. It is intended to mix information from other time granularity while maintain the corresponding time granularity information. The concatenation is then passed to a fully connected layer to calculate the result. Intuitively, if treating the updating process of a node's memories as a graph where each memory at a certain timestamp is a node, $\mathbf{M}^0$ represents node's feature and $\mathbf{M}^l$ aggregate $\mathbf{M}^{l-1}$ with its l-hop neighbor. This mechanism can be formulated as:

$$\mathbf{M}_d^l = \mathbf{W}_{mess}[\mathbf{M}_d^{l-1}||\mathrm{AGG}(\{\mathbf{M}_k^{l-1}, \forall k \neq d\})], \quad (5)$$

$$\mathrm{AGG}(\{\mathbf{M}_t\}) = \max(\{\sigma(\mathbf{W}^{pool}\mathbf{M}_t + \mathbf{b}^{pool})\}), \quad (6)$$

where $\mathbf{M}_d^l$ refers to message of d^{th} memory at l^{th} layer, σ is ReLU function, $\mathbf{W}^{pool}$, $\mathbf{W}^{mess}$, $\mathbf{b}^{pool}$ and $\mathbf{b}^{mess}$ are learnable parameters. Besides, a visual explanation of the implication graph is present in Figure1 b).

Memory Updater

When the model has computed the message for each memory, another problem arises: how to effectively update memories while preventing the model from forgetting past information. Thus, due to the capability to memorize information with reduced parameters, GRU [Cho *et al.*, 2014] is employed to update memories here. On account of a deeper message having more information, $\mathbf{M}_d^L$ is directly used as GRU's input, which can be formulated as:

$$\mathbf{H}^d = \mathrm{GRU}(\mathbf{H}^d, \mathbf{M}_d^L). \quad (7)$$

Note that there exist extremely dense edges in OD prediction, for example, millions of edges may occur in a day. It would take too much time for the model to train with one edge each iteration. Thus, a message aggregator is designed to aggregate continuous-time messages of an edge batch and reduce the training time. To be specific, instead of simply average messages of multiple edges, we extend the last update memory with another item which both considers node states and influence of time in a batch as exponential decay. Then we replace the first $\mathbf{H}$ in $\mathbf{M}'$ with the average of the concatenation memories in a batch. Formally, when $d = 0$, for the specific node i, equation (2) can be rewritten as:

$$\mathbf{H}'' = \mathbf{MEAN}_{b \in B}([\mathbf{H}^0||\mathbf{H}^0 * \mathbf{exp}(t - \mathbf{t}_b^-)]), \quad (8)$$

$$\mathbf{M}' = \begin{cases} [\mathbf{H}''||\Theta(\mathbf{H}^\mathrm{d}, \mathbf{G})||\Omega_\mathrm{d}(\mathbf{H}^\mathrm{d}, \mathbf{G})], d = 0, \\ [\mathbf{H}^d||\mathbf{Y}^\mathrm{i}_{\mathrm{t}-2^{\mathrm{d}-1}\Delta\mathrm{T},\mathrm{t}}||\Omega_\mathrm{d}(\mathbf{H}^\mathrm{d}, \mathbf{G})], d \neq 0, \end{cases} \quad (9)$$

where b represents edges departed at i in a batch. Note that the choice of the batch can be variant. The message of continuous memories can be aggregated every ρ edges while being aggregated every τ time unit is also acceptable. The latter is chosen in our implementation for convenience.

4.2 Origin-Destination Node Embedding

When embedding traffic nodes to feature space, it is crucial to consider spatial topology relations between nodes. Most previous works for OD demand prediction capture nodes' spatial relations in a direct and simple way; they mostly leverage simple GNNs on either predefined geographic graph or last transition graph. In summary, they treat origin nodes and destination nodes as the same semantic entities. However, it may not fit this problem to neglect different semantic meanings between the origin nodes and destination nodes. Besides, since the OD graph is a fully connected graph, simply aggregating all neighbors of a node may result in high computation complexity. Therefore, a random walk based Origin-Destination Node Embedding is proposed for an effective and efficient node embedding.

Random Walk Sampling

Random walk-based methods have achieved amazing success in graph embedding due to their ability to sense graph structure, e.g. DeepWalk [Perozzi *et al.*, 2014], node2vec [Grover and Leskovec, 2016]. However, directly utilizing previous random walk methods will mix up semantic relations of origin nodes and destination nodes, which should be carefully considered in OD demand prediction. For the example in Figure 1 a), there exists a dense OD demand from v_3 to v_4 and from v_3 to u at morning peak, where u and v_4 are two working area and v_3 is a living area. It appears that though there is little demand between u and v_4, they are more likely to share a similar demand pattern to other nodes. Motivated by this, the following random walk procedure is designed to distinguish the semantic entities while aggregating neighbor information.

This procedure can be explained as a) keeping sampling forward edges and reverse edges alternately and recursively, b) keeping sampling reverse edges and forward edges alternately and recursively. For example, when calculating embedding for node u in Figure 1 a), a random sampling process first starts from OD forward edges with origin node u and get destination node v_1. Then, OD reverse edges with destination v_1 are random sampled to get origin node v_2. Finally, a node-set with ω nodes is sampled, which is denoted as P^0. For another random walk, a random sampling process starts from OD reverse edges with destination node u and gets origin node v_3. Then, edges with origin node v_3 are random sampled to get node v_4. Finally, a node-set with ω nodes is sampled, which is denoted as P^1. Repeatedly using the above two random walk strategies, 2ε walks are obtained. For each step, the random sample probability is formulated as:

$$P = \begin{cases} \phi(\exp(\hat{t} - \mathbf{t}^-)), d = 0, \\ \phi(\mathrm{Y}_{\mathrm{t}-2^{\mathrm{d}-1}\Delta\mathrm{T},\mathrm{t}} + \epsilon), \mathrm{d} \neq 0, \end{cases} \quad (10)$$

where ϕ represents row-wise normalization to make the sum of probability equal to 1, $d = 0$ represents generating random walks according to continuous-time edges and $d \neq 0$ represents generating random walks according to d^{th} discrete-time memories. When calculating messages for continuous-time memories, $\hat{t}$ is firstly initialized as the calculating time t. When sampled an edge (u, v, t'), $\hat{t}$ is changed to t'.

Intra-Walk Aggregation

Based on sampled walks from the dynamic graph, we propose an intra-path aggregation module to obtain walk representations. As described above, the walks contain nodes with different semantic meanings. Therefore, a special transformation layer is designed to project nodes' into the same embedding space, which passes the node sampled as the origin to a fully connected layer W^O and passes the node sampled

	Beijing Metro						New York Taxi					
	≥ 0		≥ 3		≥ 5		≥ 0		≥ 3		≥ 5	
Method	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC	RMSE	PCC
HA	4.8672	0.8035	10.5651	0.8012	13.2245	0.7956	1.4501	0.8481	3.8801	0.7460	4.8517	0.7140
LR	5.3556	0.7521	11.7136	0.7325	15.1065	0.7229	1.3631	0.8586	3.3003	0.8042	4.1724	0.7887
XGBoost	5.7716	0.7039	12.9623	0.6453	16.8519	0.6142	1.3575	0.8599	3.3001	0.8064	4.1736	0.7915
GEML	4.6143	0.8268	10.1698	0.8075	13.1405	0.7995	1.3188	0.8718	3.1293	0.8161	3.9172	0.7980
DNEAT	5.6378	0.7334	12.8147	0.6622	16.7077	0.6180	1.5249	0.8242	3.9687	0.7161	5.1619	3.6520
TGN	6.1313	0.6453	13.5933	0.6091	17.6037	0.6003	1.2947	0.8747	3.0971	0.8188	3.9104	0.8037
HMOD	**3.7686**	**0.8861**	**8.2641**	**0.8730**	**10.6426**	**0.8675**	**1.1926**	**0.8936**	**2.8590**	**0.8507**	**3.5760**	**0.8392**

Table 1: The Comparison Results on Beijing Metro and New York Taxi.

as the destination to a fully connected layer W^D. After node content transformation, mean aggregator is applied to compute representation of a walk, which can be formulated as:

$$\mathbf{E}_{P_i} = \begin{cases} (\sum_{j=2k} \mathbf{W}^O \mathbf{H}^d_{P^{2t'}_j} + \sum_{j=2k+1} \mathbf{W}^D \mathbf{H}^d_{P^{2t'}_j})/2\omega \\ (\sum_{j=2k} \mathbf{W}^D \mathbf{H}^d_{P^{2t'+1}_j} + \sum_{j=2k+1} \mathbf{W}^O \mathbf{H}^d_{P^{2t'+1}_j})/2\omega \end{cases} \tag{11}$$

where $\mathbf{H}^d_{P^i_j}$ represents the d^{th} memory of node P^i_j, which may be different from our previous definition. $j = 2k$ means even position of a walk since even position and odd position has different semantic meaning, $P^{2t'}$ means even number of the walk since even number of walks start sampling from forwarding edges while an odd number of walks start sampling from reverse edges.

Inter-Walk Aggregation

After intra-walk aggregation for each walk embedding, an attention aggregation is designed to aggregate information from different walks, which can be formulated as:

$$\mathbf{E}'_{P_i} = \mathbf{W}^a \mathbf{E}_{P_i}, \tag{12}$$

$$\alpha_i = \mathbf{a}\mathbf{E}'_{P_i}, \tag{13}$$

$$\beta_i = \frac{\exp(\alpha_i)}{\sum_{i=0}^{2\varepsilon-1} \exp(\alpha_i)}, \tag{14}$$

$$\mathbf{Z} = \beta_\mathbf{i} \mathbf{E}_{\mathbf{P_i}}, \tag{15}$$

where $\mathbf{W}^a$, $\mathbf{a}$ are learnable parameters.

4.3 OD Matrix Prediction

Fusion and Prediction

If requiring predicting OD Matrix at timestamp t, the hierarchical memories at t are fused to make the prediction. Since each memory contain specific granularity information, the memories are concatenated to keep the hierarchical information. Then a MLP is adopted to make the prediction. The i^{th} row of prediction result can be formulated as:

$$\mathbf{H}' = \|_{d=0}^{D} \mathbf{H}^d \tag{16}$$

$$\hat{\mathbf{Y}}^i = \mathbf{W}^{o_1} \sigma(\mathbf{W}^{o_2} \mathbf{H}' + \mathbf{b}^{o_2}) + \mathbf{b}^{o_1} \tag{17}$$

where $\mathbf{W}^{o_1}$, $\mathbf{W}^{o_2}$, $\mathbf{b}^{o_1}$, $\mathbf{b}^{o_2}$ are learnable parameters.

OD Loss

One last problem is there exists a large number of zeros in OD matrix while predicting them as negative numbers are also acceptable in the real-world scene. Besides, since a heavy negative number and a slice negative number may contain different meanings, directly training them to be zero could lead to poor performance. Therefore, an OD Loss is designed based on MSE, which can be formulated as:

$$\mathcal{L} = \frac{1}{|\mathbf{Y}|} \sum_{y \in \mathbf{Y}} (I(y, \hat{y})(y - \hat{y})^2), \tag{18}$$

$$I(y, \hat{y}) = \begin{cases} 0, y = 0 \& \hat{y} \leq 0 \\ 1, \text{else} \end{cases}. \tag{19}$$

5 Experiments

5.1 Dataset

To verify the performance of our framework, extensive experiments are conducted on two real-world datasets.

Beijing Metro Beijing Metro dataset contains railway trip data in Beijing from June to July in 2018. The first 6 weeks are selected as the training set, the next 1 week as the validation set, and another next 1 week as the test set. This dataset covers 268 stations of Beijing Metro and has over 200 million timestamped edges in total.

New York Taxi New York Taxi dataset contains taxi trip data in New York from January to June in 2019. The first 139 days are selected as the training set, the next 21 days as the validation set, and the next 21 days as the test set. This dataset covers 63 regions of New York and has over 38 million timestamped edges in total.

5.2 Experimental Settings

The task is to predict OD demand matrix in the next $\tau = 30$ minutes with historical trip data. Adam [Kingma and Ba, 2014] is used as the optimizer with learning rate 1e-4 for Beijing Metro and 1e-5 for New York Taxi, and the learning rate is chosen from [1e-2, 1e-3, 1e-4, 1e-5, 1e-6]. Memory dimension d_H and message dimension d_M are set to 128, which is chosen from [64, 128, 256, 512]. Each experiment contains 500 epochs and an early stop strategy with 20-epoch patience is used to avoid overfitting. The model is implemented using the PyTorch framework on a machine with Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz and 4 Tesla T4

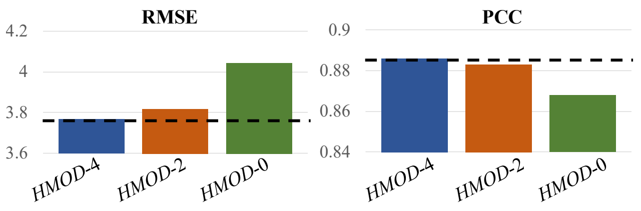

Figure 2: Ablation on Hierarchical Memories

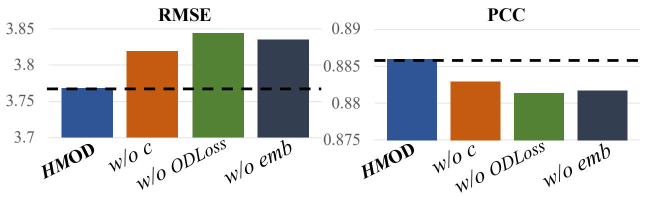

Figure 3: Ablation on Main Modules

GPUs. PCC (Pearson Correlation Coefficient) and RMSE (Root Mean Squared Error) are adopted as metrics to evaluate the performance of our model.

5.3 Baseline Methods

- HA (Historical Average) computes the historical average of OD demand matrix as the prediction.
- LR (Linear Regression) is a regression model which exploits linear correlations between input and output. The input of LR contains the historical OD demand of one OD pair from the last 4 consecutive snapshots.
- $XGBoost$ [Chen and Guestrin, 2016] adopts gradient boosting tree to learn from the previous pattern. The input of XGBoost is organized as the same as LR.
- $GEML$ [Wang *et al.*, 2019] is an OD demand prediction model based on snapshots and pre-defined neighborhoods. It adopts graph convolution along with a skip-RNN to extract spatial and temporal patterns.
- $DNEAT$ [Zhang *et al.*, 2021] is another OD demand prediction model based on snapshots and node-edge attention. It adopts graph attention along with a dilated convolution to capture spatiotemporal dependencies among discrete-time snapshots of a dynamic graph.
- TGN [Rossi *et al.*, 2020] is a continuous-time dynamic graph representation learning framework. This framework is deployed to OD prediction task by this paper.

5.4 Experimental Results

Table 1 shows the comparison results, where $\geq p$ means the accuracy in the case where truth-value $\geq p$. The evaluation with different p indicates whether the methods perform well in different situations. The results show that our model outperforms all the baselines on both RMSE and PCC, both datasets, and all the situations. In BJMetro, TGN performs poorer than other OD prediction methods. This is because BJMetro has a larger amount nodes and edges, which may make it harder to find important neighbors for TGN. However, in New York Taxi, it performs better than other baselines, which demonstrates the effectiveness of continuous-time information in this scenario. Owing to hierarchical-time node representations, the proposed method could capture fine-grained and macro information simultaneously and always perform well.

5.5 Ablation Study

Ablation on Hierarchical Memories

The effect of hierarchical memories on Beijing Metro is shown in figure 2. The $HMOD$-x refers to our model with x discrete-time memories, while $HMOD-4$ is the setting in our model. For the convenience in our implements, the first discrete memory leverages δT = 30 mins input, and the k-th discrete memory leverages $2^k\delta T$ mins input. The results show that only one continuous memory can achieve a significant result, which demonstrates the effectiveness of the continuous-time method. Besides, the more discrete-time information added, the better performance will HMOD achieve, which also reflects the importance of the discrete-time information.

Ablation on Main Modules

The impact of main modules on Beijing Metro is shown in figure 3. The $w/o\ c$ refers to removing the continuous-time memory part and containing four discrete-time memories to predict OD Matrix. The $w/o\ emb$ refers to removing the OD node embedding part. The $w/o\ ODLoss$ refers to replacing our designed OD Loss with MSE loss. The $w/o\ c$ outperforms other baselines demonstrating the effectiveness of our message passing framework, while it performs worse than HMOD reflects the need for continuous-time memories. Perhaps due to the fact that ODLoss makes the negative predicted demand acceptable, which makes sense in the real world, it improves the result significantly. Besides, the results on the $w/o\ emb$ demonstrate that our OD node embedding plays an important role in predicting OD demand, which could learn an meaningful node embedding.

6 Conclusion

This paper aimed to solve the pairwise OD demand prediction problem and proposed a novel dynamic node representation learning framework, HMOD. A hierarchical memory updater powered by a memory fusion mechanism was proposed to integrate both continuous-time fine-grained spatiotemporal dependencies and discrete-time general spatiotemporal dependencies. This is the first time traffic node representation learning is extended by continuous-time dynamic graph view. Meanwhile, an OD node embedding method based on temporal conditioned random walks was also proposed to aggregate semantic information from previous neighbors. Experiments on real-world datasets showed high prediction accuracy and robustness of the proposed model.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (71901011 and U1811463)

References

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM, 2016.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[Han *et al.*, 2021] Liangzhe Han, Bowen Du, Jingjing Lin, Leilei Sun, Xucheng Li, and Yizhou Peng. Multi-semantic path representation learning for travel time estimation. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kumar *et al.*, 2019] Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1269–1278, 2019.

[Ma *et al.*, 2019] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6120–6127, 2019.

[Noursalehi *et al.*, 2021] Peyman Noursalehi, Haris N Koutsopoulos, and Jinhua Zhao. Dynamic origin-destination prediction in urban rail systems: A multi-resolution spatio-temporal deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[Rossi *et al.*, 2020] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[Sankar *et al.*, 2020] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 519–527, 2020.

[Shi *et al.*, 2020] Hongzhi Shi, Quanming Yao, Qi Guo, Yaguang Li, Lingyu Zhang, Jieping Ye, Yong Li, and Yan Liu. Predicting origin-destination flow via multi-perspective graph convolutional network. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 1818–1821. IEEE, 2020.

[Singer *et al.*, 2019] Uriel Singer, Ido Guy, and Kira Radinsky. Node embedding over temporal graphs. *arXiv preprint arXiv:1903.08889*, 2019.

[Trivedi *et al.*, 2019] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. Dyrep: Learning representations over dynamic graphs. In *7th International Conference on Learning Representations, ICLR 2019,New Orleans, LA, USA, May 6-9, 2019*, 2019.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang *et al.*, 2019] Yuandong Wang, Hongzhi Yin, Hongxu Chen, Tianyu Wo, Jie Xu, and Kai Zheng. Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1227–1235, 2019.

[Wang *et al.*, 2021] Yuandong Wang, Hongzhi Yin, Tong Chen, Chunyang Liu, Ben Wang, Tianyu Wo, and Jie Xu. Gallat: A spatiotemporal graph attention network for passenger demand prediction. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2129–2134. IEEE, 2021.

[Yao *et al.*, 2018] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[Zhang *et al.*, 2021] Jinlei Zhang, Hongshu Che, Feng Chen, Wei Ma, and Zhengbing He. Short-term origin-destination demand prediction in urban rail transit systems: A channel-wise attentive split-convolutional neural network method. *Transportation Research Part C: Emerging Technologies*, 124:102928, 2021.

[Zhou *et al.*, 2018] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.