Generic and Dynamic Graph Representation Learning for Crowd Flow Modeling

Liangzhe Han[1], Ruixing Zhang[1], Leilei Sun[1]*, Bowen Du[1], Yanjie Fu[2], Tongyu Zhu[1]

[1]State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
[2]Department of Computer Science, University of Central Florida, Florida, USA
{liangzhehan, 19373115, leileisun, dubowen, zhutongyu}@buaa.edu.cn, yanjie.fu@ucf.edu

Abstract

Many deep spatio-temporal learning methods have been proposed for crowd flow modeling in recent years. However, most of them focus on designing a spatial and temporal convolution mechanism to aggregate information from nearby nodes and historical observations for a pre-defined prediction task. Different from the existing research, this paper aims to provide a generic and dynamic representation learning method for crowd flow modeling. The main idea of our method is to maintain a continuous-time representation for each node, and update the representations of all nodes continuously according to the streaming observed data. Along this line, a particular encoder-decoder architecture is proposed, where the encoder converts the newly happened transactions into a timestamped message, and then the representations of related nodes are updated according to the generated message. The role of the decoder is to guide the representation learning process by reconstructing the observed transactions based on the most recent node representations. Moreover, a number of virtual nodes are added to discover macro-level spatial patterns and also share the representations among spatially-interacted stations. Experiments have been conducted on two real-world datasets for four popular prediction tasks in crowd flow modeling. The result demonstrates that our method could achieve better prediction performance for all the tasks than baseline methods.

Introduction

Crowd flow modeling aims to describe how citizens flow between traffic zones or stations. This subject is of great importance to urban planning and traffic control.

To achieve crowd flow modeling, there are multiple types of tasks to solve, including inflow prediction, outflow prediction, origin-destination (OD) flow prediction, and travel time estimation. In essence, the traveling behaviors of citizens have momentous spatial dependence and temporal evolving patterns. Recently, a large of spatial-temporal methods has been intuitively proposed on this subject. For spatial relations between traffic nodes, some studies split the area as grids and leverage convolutional neural networks (CNN) to solve the prediction problem (Yao et al. 2018; Liu et al. 2019a; Wang et al. 2020). Some other studies take a further step to consider stations or arbitrary-shaped regions by the graph neural networks (GNN) (Geng et al. 2019; Shi et al. 2020). For temporal relations along with time, previous work basically aggregates historical data as snapshots, and recurrent neural networks (RNN) (Wang et al. 2019), fusion modules (Zhang, Zheng, and Qi 2017), and attention-based modules (Wang et al. 2020) are designed on them to predict crowd flow in the future. For instance, STDN (Yao et al. 2019) utilizes CNN for spatial relations and LSTM for temporal relations to predict inflow and outflow for each gird; GEML (Wang et al. 2019) combines GNN and LSTM to predict OD flow between each pair of nodes. It can be seen that most previous researches focus on designing temporal and spatial convolutional modules. Additionally, most researches provide a particular prediction for a subset of tasks.

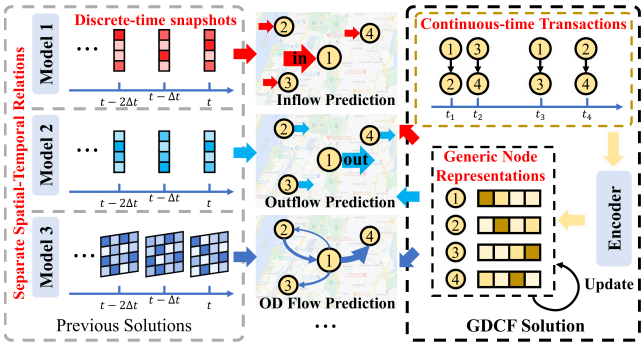

Figure 1: Generic representations for crowd flow modeling.

Different from previous studies, this paper aims to conduct generic and dynamic representation learning for crowd flow modeling, as shown in Figure 1. On the one hand, this idea provides a novel insight into crowd flow modeling. Instead of focusing on spatial convolutions between traffic nodes and temporal modules for multiple snapshots, it encodes spatial and temporal information with unique node representations. On the other hand, generic representations learned in this way can be leveraged to solve multiple types of downstream tasks, including but not limited to the prevalent station-level flow prediction and OD flow prediction. This characteristic makes it able to adapt to complex conditions and beneficial for a further range.

However, it is a non-trivial task to bond crowd flow mod-

*Corresponding Author.

eling and representation learning due to the following reasons: (1) To achieve effective crowd flow modeling, spatial-temporal patterns in raw data are supposed to be well handled. How to maintain representations for each node to store spatial-temporal information is important but challenging. (2) The crowd status is evolving all the time, which can be reflected in streaming transactions. How to update representations by newly observed transactions with continuous time features is a challenging task. (3) The generic representations aim to support multiple scenarios instead of fit on certain tasks. It is challenging to design a mechanism such that the representations store as much information as possible.

To address these issues, this paper proposes a **G**eneric and **D**ynamic graph representation learning framework for **C**rowd **F**low modeling (GDCF). Specifically, we propose an encoder-decoder architecture and adopt a 2-phase training mechanism to generate generic node representations. When compressing the transaction information to node representations, instead of processing the input as snapshots and discretizing the time information, the encoder views the happening time as a continuous feature and computes directly on the raw timestamped transactions. It maintains memories for each traffic node and updates them when new transactions happen. Meanwhile, an adaptive global relation learning module is designed to aggregate traffic nodes as virtual nodes and share useful information among nodes at macro levels. Then, a continuous-time dynamic graph decoder is proposed to reconstruct the input based on the node representations to ensure they contain crucial information about crowd flow. After training the autoencoder in phase 1, a simple model in phase 2 can obtain satisfactory performance based on the well-learned node representations. Contributions of this work can be summarized as:

- **A novel crowd flow modeling framework**, which enables multiple downstream prediction tasks based on a generic and dynamic representation of each node. This framework is quite different from previous research mainly designed for specific tasks.
- **An encoder-decoder architecture**, where the encoder compresses raw transactions to evolving node representations and the decoder recovers the input. It ensures that the node representations can contain crucial information from the input as much as possible.
- **A spatial relation learning module**, which can discover latent spatial relations between nodes and update node representations at macro levels.

Preliminaries

Transaction Dynamic Graph

Unlike previous work generally processing historical transactions as snapshots, GDCF aims to learn directly from raw data to keep more information into node representations. In this paper, the data is organized as a continuous-time dynamic graph. The dynamic graph is denoted as $\mathcal{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V} = \{v_1, v_2, \cdots, v_N\}$ is a finite set of N traffic nodes; $\mathbb{E} = \{e_1, e_2, \cdots, e_M\}$ is the set of M timestamped transactions. Each traffic node represents a fixed metro station or a taxi zone and an edge $e_m = (v_m^o, v_m^d, t_m)$ represents that a citizen travels from v_m^o to v_m^d at time t_m. Each node has a feature vector, and features of all nodes are denoted as $\mathbf{F} \in \mathbb{R}^{N \times d^F}$, where d^F is the dimension of feature vectors. Moreover, the transaction dynamic graph at a certain time t is denoted as $\mathcal{G}^t = (\mathbb{V}, \{e_k | t_k < t\})$, which contains all transactions before time t.

Crowd Flow

Crowd flow is a kind of data that describes how people travel between traffic nodes. In this paper, three types of crowd flow are considered: inflow, outflow, and OD flow. The inflow refers to how many citizens get in a traffic node from other nodes. It is denoted as $\mathbf{X}^{I,t:t+\tau} \in \mathbb{R}^N$ and inflow of traffic node i is defined as $X_i^{I,t:t+\tau} = |\{e_m | e_m \in \mathbb{E} \wedge (t \leq t_m < t + \tau) \wedge v_m^d = v_i\}|$. The outflow means how many citizens leave out a traffic node. Similarly, it is defined as $X_i^{O,t:t+\tau} = |\{e_m | e_m \in \mathbb{E} \wedge (t \leq t_m < t + \tau) \wedge v_m^o = v_i\}|$. Moreover, OD flow describes both the amount and the destination of the crowd flow, which is defined as a matrix $\mathbf{X}^{OD,t:t+\tau} \in \mathbb{R}^{N \times N}$ and its (i, j)-entry is how many citizens travel from v_i to v_j: $X_{i,j}^{OD,t:t+\tau} = |\{e_m | e_m \in \mathbb{E} \wedge (t \leq t_m < t + \tau) \wedge v_m^o = v_i \wedge v_m^d = v_i\}|$.

Problem Definition

Given the historical transaction records until time t, $\mathcal{G}^t = (\mathbb{V}, \{e_k | t_k < t\})$, our goal is to learn an encoder f and decoder g such that $f(\mathcal{G}^t) = \mathbf{Z}^t$ and $g(\mathbf{Z}^t) = \mathcal{G}^t$, where $\mathbf{Z}^t \in \mathbb{R}^{N \times d}$ represents high-level representations $\mathbf{z}_i^t \in \mathbb{R}^d$ for each node i at time t. The objective is defined as:

$$f, g = \arg\min_{f,g} \mathcal{L}(\mathcal{G}^t, g(f(\mathcal{G}^t))). \tag{1}$$

These representations will then be retrieved and used for downstream tasks including forecasting inflow, outflow and OD flow: $X_i^{I,t:t+\tau} = f^I(\mathbf{z}_i^t)$, $X_i^{O,t:t+\tau} = f^O(\mathbf{z}_i^t)$ and $X_{i,j}^{OD,t:t+\tau} = f^{OD}(\mathbf{z}_i^t, \mathbf{z}_j^t)$.

Methodology

The overall pipeline is shown in Figure 2, which follows a two-phase training strategy. In the first phase, it follows an encoder-decoder structure. The core idea of this structure is to generate dynamic node representations to maintain historical transaction information as much as possible. It is achieved by a continuous-time dynamic graph encoder and an edge-level continuous-time dynamic graph decoder. The continuous-time dynamic graph encoder maintains a memory vector for each traffic node and updates them when new transactions happen. Based on these node representations, the decoder aims to reconstruct the input transactions to ensure that these representations can encode crucial spatial-temporal patterns. Moreover, an adaptive global relation learning module is proposed to discover and handle spatial relations between traffic nodes. In the second phase, the representations can empower multiple downstream tasks including inflow prediction, outflow prediction, OD flow prediction, and more potential tasks.

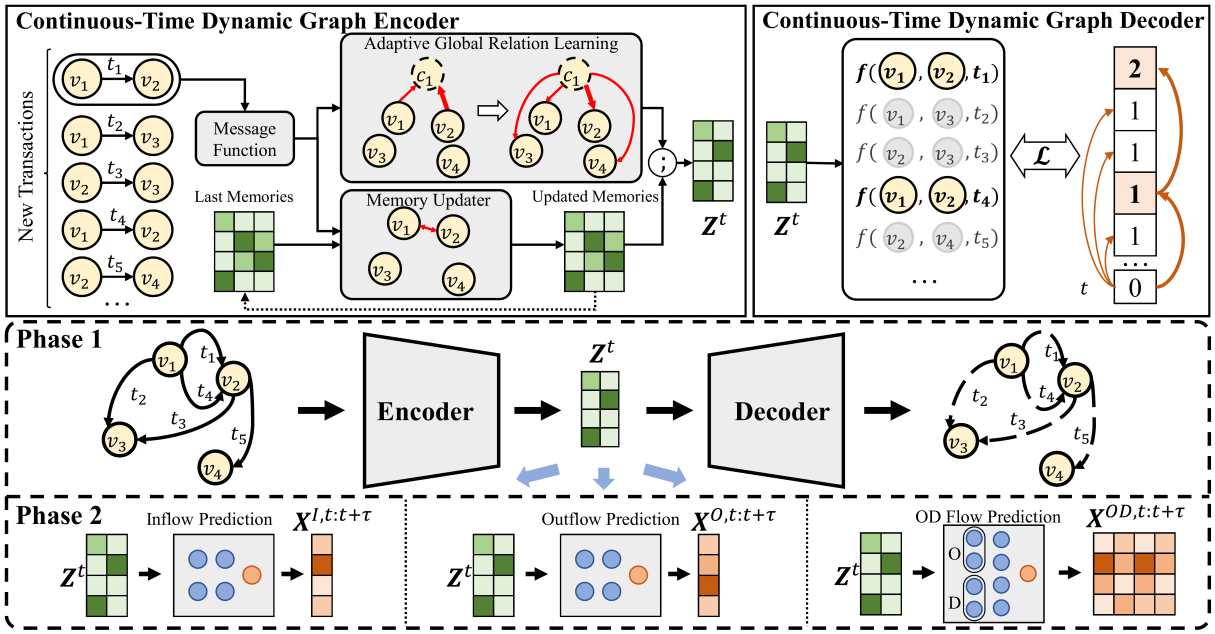

Figure 2: The overall architecture of GDCF. In phase 1, an encoder and a decoder are trained to reconstruct the input crowd flow dynamic graph. The encoder will maintain memories for nodes and compress transaction records into evolving node representations. The decoder recovers the input with the node representations to ensure they contain crucial information. In phase 2, the well-trained node representations will be retrieved and used to solve multiple tasks.

Continuous-Time Dynamic Graph Encoder

Unlike snapshots-based methods, which discretize time features and will cause information loss, this paper adopts the thoughts of continuous-time dynamic representation learning and proposes a continuous-time dynamic encoder. The encoder can compress transaction records with continuous time features into evolving node representations.

To model crowd flow between traffic nodes, the amount and the happening time are two crucial factors. And there are two basic assumptions: first, the more recently a transaction happens, the more important it will be; second, the more frequently node i interacts with node j, the more effect node j should have on node i. Along this line, the representation of node i can be expected to be a weighted combination of representations from historically interacted nodes:

$$\mathbf{z}_i^t = \frac{\sum_{(v_i,v_j,t')\in E^t} exp(-\lambda(t-t'))\mathbf{z}_j^{t'}}{\sum_{(v_i,v_j,t')\in E^t} exp(-\lambda(t-t'))}, \tag{2}$$

where $\mathbf{z}_i^t$ is representation of node i at time t. However, directly calculating by Equation 2 needs to recompute the weights of all previous interactions. Here we can choose to maintain two accumulators as memories, $\mathbf{a}$ for the numerator and b for the denominator, to update the representations in an online manner:

$$\mathbf{z}_i^t = \frac{\mathbf{a}_i^t}{b_i^t} = \frac{exp(-\lambda(t-t^-))\mathbf{a}_i^{t^-} + \mathbf{z}_j^{t^-}}{exp(-\lambda(t-t^-))b_i^{t^-} + 1}. \tag{3}$$

The updating process is the addition of decayed last memories and information from newly happened interactions.

Furthermore, when learning node representations, more node features and complex patterns need to be taken into consideration. Formally, we first compute messages as:

$$\begin{aligned} \mathbf{p}_i^t &= \sum_{(v_i,v_j,t_k)\in E^t - E^{t^-}} exp(-\lambda(t-t_k))[\mathbf{z}_j^{t^-};\mathbf{F}_j], \\ q_i^t &= \sum_{(v_i,v_j,t_k)\in E^t - E^{t^-}} exp(-\lambda(t-t_k)), \end{aligned} \tag{4}$$

where t^- is the last update time of node representations and $[\cdot;\cdot]$ is a concatenation of two vectors. And node representations are generated based on the updated memories:

$$\mathbf{z}_i^{t'} = \frac{\mathbf{a}_i^t}{b_i^t} = \frac{exp(-\lambda(t-t^-))\mathbf{a}_i^{t^-} + MLP(\mathbf{p}_i^t)}{exp - \lambda((t-t^-))b_i^{t^-} + q_i^t}. \tag{5}$$

This procedure introduces node features $\mathbf{F}$ in the messages and more expressive power by $MLP()$ in the update procedure to make it adapt to more situations.

Adaptive Global Relation Learning Module

In the above procedure, a transaction only affects representations of two involved nodes. However, it may contain useful information for a wider range. For example, the rise of crowd flow on one node may indicate the rise on other nodes with similar functions. However, updating all node representations whenever new transactions happen is time-consuming, and the relations between traffic nodes are also complex and implicit.

To address these issues, an adaptive global relation learning module is proposed here to discover spatial relations between nodes and propagate information to more nodes. It

is implemented by creating n virtual nodes to collect and distribute information. Specifically, representations of virtual nodes are kept as memories and a parameter matrix $\mathbf{A} \in \mathbb{R}^{N \times n}$ is assigned to denote belonging relations between the traffic nodes and virtual nodes. After getting messages from Equation 4, they are collected for virtual nodes:

$$\mathbf{p}_i^{c,t} = \frac{\sum_{j=1}^{N} exp(A_{j,i})(\mathbf{W}\frac{\mathbf{p}_j^t}{q_j^t})}{\sum_{j=1}^{N} exp(A_{j,i})}, i = 1, 2, \cdots, n \quad (6)$$

where $\mathbf{p}_i^{c,t}$ denotes messages for virtual node i at time t. Then representations of virtual node i are updated as:

$$\mathbf{z}_i^{c,t} = exp(-\lambda(t - t^-))\mathbf{z}_i^{c,t^-} + MLP(\mathbf{p}_i^{c,t}). \quad (7)$$

When generating representations for traffic nodes, the information contained in virtual nodes is distributed globally:

$$\mathbf{z}_i^{c,t'} = \frac{\sum_{j=1}^{n} exp(A_{i,j})\mathbf{z}_j^{c,t}}{\sum_{j=1}^{n} exp(A_{i,j})}, i = 1, 2, \cdots, N \quad (8)$$

$$\mathbf{Z}^t = [\mathbf{Z}^{t'}; \mathbf{Z}^{c,t'}]. \quad (9)$$

Continuous-time Dynamic Graph Decoder

In the above section, a continuous-time dynamic graph encoder is proposed to compress historical transaction records into evolving node representations. Another essential problem of phase 1 is how to ensure that these node representations can contain useful information as much as possible.

Here, we design an encoder-decoder structure, which reconstructs the input as its output. A key problem is how to design a decoder to extract a dynamic graph given node representations. However, it is a non-trivial task to cope with the continuous-time dynamic graph. On the one hand, instead of merely predicting whether there exists a transaction between two traffic nodes, another non-negligible factor is the happening time of transactions. On the other hand, as stated before, the amount and the density of transactions between two traffic nodes are crucial in crowd flow modeling.

To this end, this paper proposes a novel continuous-time decoder to recover input crowd flow dynamic graph. The objective of the decoder is to predict $\mathbf{Y}^t$, the amount of transactions from v_k^o to v_k^d between t_k and t for each transaction in input $(v_k^o, v_k^d, t_k) \in \mathbb{E}^t - \mathbb{E}^{t^-}$. Formally, for input $e_k = (v_k^o, v_k^d, t_k)$, the k-th entry Y_k^t is calculated by:

$$\begin{aligned} Y_k^t = &|\{e_m | e_m \in \mathbb{E}^t \wedge v_m^o = v_k^o \\ &\wedge v_m^d = v_k^d \wedge (t_k \leq t_m < t)\}|. \end{aligned} \quad (10)$$

And the updated node representations can be directly leveraged to predict Y_k^t:

$$\hat{Y}_k^t = g(v_k^o, v_k^d, t_k) = MLP([\mathbf{z}_{v_k^o}^t; \mathbf{z}_{v_k^d}^t; (t - t_k)]). \quad (11)$$

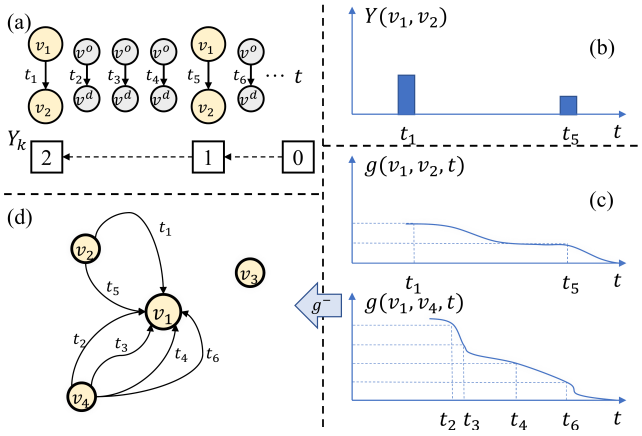

Figure 3: Illustration of the continuous-time decoder.

The loss function of phase 1 is defined as:

$$\begin{aligned} \mathcal{L} = &\frac{1}{|\mathbb{E}^t - \mathbb{E}^{t^-}|} \sum_{e_k \in \mathbb{E}^t - \mathbb{E}^{t^-}} \frac{(Y_k^t - \hat{Y}_k^t)^2}{C_{v_k^o, v_k^d}^{t^-:t}} \\ &+ \frac{1}{|\mathbb{E}^0|} \sum_{(v_k^o, v_k^d) \in \mathbb{E}^0} g(v_k^o, v_k^d, t^-)^2, \end{aligned} \quad (12)$$

where $\mathbb{E}^t - \mathbb{E}^{t^-}$ is the batch of newly happened transactions since last update time t^-, $C_{v^o, v^d}^{t^-:t} = |\{e_i | e_i \in (\mathbb{E}^t - \mathbb{E}^{t^-}) \wedge v_i^o = v^o \wedge v_i^d = v^d\}|$ is a normalizer between different node pairs and $\mathbb{E}^0 = \{(v^o, v^d) | (v^o, v^d, \cdot) \notin (\mathbb{E}^t - \mathbb{E}^{t^-})\}$ is the complementary set consisting of node pairs that never appear in the input. The first term models the appearing pattern of transactions and the second term punishes node pairs that have no crowd flow in a period of time.

The rationale of the decoder can be explained by an example shown in Figure 3. The input of this example is transactions in Figure 3(a). For each input transaction e_k, Y_k^t is calculated according to Equation 10. Specifically, for $e_k = (v_1, v_2, t_1)$, the corresponding Y_k^t is calculated as 2, the amount of transactions from v_1 to v_2 between t_1 and t; for $e_k = (v_1, v_2, t_5)$, the corresponding Y_k^t is calculated as 1. The objective $\mathbf{Y}^t$ between v_1 and v_2 across time is illustrated in Figure 3(b). These two transactions act as two slices from a cumulative amount function of transactions between two traffic nodes. Given the $\mathbf{Y}^t$, we can train the decoder $g()$, which takes time t_k and node representations of v_k^o and v_k^d as input to predict Y_k^t. If the decoder is well trained and the input time is densely sampled, it is expected to output a curve, as shown in the upper half of Figure 3(c). Meanwhile, other node pairs may have different cumulative patterns, as shown in the bottom half of Figure 3(c). These curves between each pair of traffic nodes can be easily transferred to the form of the continuous-time dynamic graph shown in Figure 3(d) calculating time by the inverse function $g^-()$. This example indicates that if the task is well-solved, which aims to predict the cumulative amount of transactions, the node representations can contain information to recover the input continuous-time dynamic graph.

In summary, this decoder has the following advantages to reconstruct the continuous-time crowd flow dynamic graph: First, it can distinguish crowd flow between traffic nodes by the amount. As the decoder aims to predict the cumulative amount, it is natural to compress the abundant information into the node representations. Second, it can reconstruct the time of the input transaction records by the reverse function as the above example shows. Third, note that the task is associated with each input transaction, it can be easily extended for more attributes of transactions if needed.

Downstream Task and Training Procedure

In phase 1, the autoencoder is trained with loss function defined in Equation 12. Specifically, the model is trained on the training set and the best parameters are chosen by the performance on the validation set. Then, the encoder is used to generate evolving node representations for the whole dataset. In phase 2, the saved node representations in the training set are used to train the downstream model and the best parameters on the validation set are used to evaluate performance on the test set. For inflow and outflow prediction, representations of one node serve as input; for OD flow prediction, representations of the origin node and destination node are combined to be the input:

$$\begin{aligned} X_i^{I,t:t+\tau} = MLP(\mathbf{z}_i^t), X_i^{O,t:t+\tau} = MLP(\mathbf{z}_i^t), \\ X_{i,j}^{OD,t:t+\tau} = MLP([\mathbf{z}_i^t; \mathbf{z}_j^t]). \end{aligned} \tag{13}$$

Experiments

Datasets

The experiments are conducted on two real-world datasets: **BJSubway**, which contains IC card transaction records in Beijing Subway from 2017.6 to 2017.7, and **NYTaxi**, which contains taxi orders in Manhattan from 2019.1 to 2019.6 [1]. More details about these datasets are listed in Table 1. The source code is available at https://github.com/liangzhehan/GDCF.

Dataset	BJSubway	NYTaxi
# Traffic Nodes	268	63
# Transactions	279,227,618	38,498,427
# Train Days	42	139
# Validation Days	7	21
# Test Days	7	21

Table 1: Statistic information of datasets.

Baselines

The performance of GDCF is compared with baselines including simple statistic methods HA (Historical Average), traditional machine learning methods LR (Linear Regression), XGBoost (Chen et al. 2015), crowd flow prediction methods based on snapshots GEML (Wang et al. 2019), DNEAT (Zhang et al. 2021), and continuous-time dynamic graph representation learning methods TGN (Rossi et al. 2020), DyRep (Trivedi et al. 2019).

[1]Data is available at https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Experiment Setups

For both datasets, the number of virtual nodes n in the adaptive global relation learning module is $\sqrt{N}$. One-hot encoding is used as node features. The proposed method is implemented with the Pytorch toolbox on a machine with 4 Tesla T4 GPUs. The memory dimension and message dimension are 256 on BJSubway and 128 on NYTaxi. Adam optimizer with an initial learning rate of 0.0001 and an early stopping strategy with patience 20 are utilized to train the proposed model in phase 1. In phase 2, the initial learning rate is 0.001 and the patience is set as 50. The learning rate of all deep learning methods is chosen from [0.01, 0.001, 0.0001, 0.00001] according to the best performance on the validation set. The best parameters on the validation set are chosen to evaluate the performance of the test set. All deep learning methods are repeated with different seeds 5 times and the average value is reported. Mean Average Error (MAE), Root Mean Square Error (RMSE), Pearson Correlation Coefficient (PCC), and Symmetric Mean Absolute Percent Error (SMAPE) are selected as metrics to compare.

Comparison Results

From the comparison in Table 2, it can be observed: (1) GDCF outperforms other methods in most cases, especially on the BJSubway dataset, which suggests the effectiveness of the proposed method to learn meaningful node representations for crowd flow modeling. (2) Though continuous-time dynamic graph representation learning methods (TGN and DyRep) can directly operate raw timestamped transactions, they don't perform well in these tasks and DyRep even encounters the out-of-memory problem. One potential reason may be that some designs of them are improper for crowd flow modeling. For example, TGN and DyRep both aggregate information from only dozens of sampled neighbors, which may be enough for some tasks but there are an extremely large number of events in the context of crowd flow modeling. In this task, the sampling ratio is so small and will bring more randomness and noise. (3) Deep learning-based methods can perform better than simple methods in some cases of OD flow prediction problems. However, they are generally worse on inflow and outflow prediction, especially on NYTaxi. The reason may be that patterns in these tasks only involve single traffic nodes and are simpler. Meanwhile, the traffic nodes on NYTaxi are less than those on BJSubway, which is more beneficial to simple models. Moreover, the proposed GDCF performs well on both simple and complex tasks demonstrating its robustness.

Ablation Study

An ablation study is conducted on the BJSubway dataset with two variants: **GDCF (e2e)** solves the crowd flow prediction tasks in the end-to-end manner. This variant directly passes node representations from the encoder to the prediction module and trains these two modules together. **GDCF w/o GR** removes the adaptive global relation learning module and only updates representations of interacted nodes when new transactions happen. The result is shown in Figure 4; it can be observed that: (1) GDCF performs better

Task	Method	BJSubway				NYTaxi			
		MAE	RMSE	PCC	SMAPE	MAE	RMSE	PCC	SMAPE
Inflow	HA	350.1292	696.9214	0.5212	0.5388	33.7985	52.1861	0.6704	0.5713
	LR	135.4600	244.4182	0.9540	0.2890	11.1817	17.3974	0.9683	0.3060
	XGBoost	103.9266	192.3048	0.9718	0.2035	10.7097	**16.5697**	**0.9713**	0.2721
	GEML	122.8809	273.8728	0.9428	0.2477	12.4134	20.5709	0.9571	0.3008
	DNEAT	152.1489	308.2064	0.9499	0.4007	16.1151	23.9384	0.9700	0.4173
	TGN	303.2444	553.1976	0.7392	0.4889	18.0415	30.2193	0.9027	0.4038
	DyRep	-	-	-	-	12.0174	19.2415	0.9622	0.3245
	GDCF	**78.9725**	**142.3619**	**0.9862**	**0.1757**	**10.5956**	*17.1821*	*0.9693*	**0.2536**
Outflow	HA	362.5770	677.1209	0.5145	0.5849	34.5884	54.0211	0.6936	0.5906
	LR	146.0107	277.7490	0.9359	0.3126	11.4772	18.3448	0.9690	0.3559
	XGBoost	121.0290	236.4546	0.9539	0.2419	11.0106	17.9179	0.9705	0.2951
	GEML	124.3017	246.6373	0.9505	0.2738	12.9655	22.2864	0.9558	0.3514
	DNEAT	156.0036	325.6022	0.9485	0.3147	17.5183	28.3914	0.9680	0.4413
	TGN	229.5789	486.9007	0.7831	0.3680	13.3570	21.8626	0.9563	0.3540
	DyRep	-	-	-	-	9.7327	15.9267	0.9770	0.2904
	GDCF	**71.7893**	**122.1684**	**0.9886**	**0.1700**	**9.4054**	**15.7269**	**0.9774**	**0.2478**
OD Flow	HA	2.1873	7.0008	0.5084	0.6588	0.8957	1.9452	0.6864	0.4458
	LR	1.9387	5.3557	0.7520	0.7040	0.6944	1.3645	0.8578	0.4112
	XGBoost	1.8158	5.8157	0.6984	0.6481	0.6965	1.3674	0.8572	0.4203
	GEML	1.8133	4.6409	0.8216	0.6472	0.6568	1.3072	0.8721	0.3829
	DNEAT	1.4772	5.7170	0.7238	**0.4954**	0.6510	1.5502	0.8184	0.3501
	TGN	2.1031	5.8927	0.6755	0.6921	0.6382	1.2884	0.8768	0.3737
	DyRep	-	-	-	-	0.6049	1.2166	0.8893	0.3510
	GDCF	**1.3926**	**3.6135**	**0.8959**	*0.5382*	**0.5807**	**1.1750**	**0.8968**	**0.3321**

Table 2: Results of comparison with baselines.

than GDCF (e2e) in all cases; it demonstrates that the proposed encoder-decoder structure and the continuous-time dynamic graph decoder can learn useful node representations for crowd flow modeling. The reason may be that the decoder aims to reconstruct the input transactions, which can provide more detailed supervisory signals to help the learning procedure. (2) The outperformance over GDCF w/o GR demonstrates that by discovering relations between traffic nodes and propagating transaction information to a wider range, the adaptive global relation module benefits the ability of crowd flow modeling.

Method	MAE	RMSE	PCC	SMAPE
R-TTE	183.9617	256.8924	0.8026	0.2694
S-TTE	184.3105	257.0542	0.8020	0.2702
OD-TTE	156.0614	222.2881	0.8576	0.2337
GDCF-TTE	**141.7542**	**207.7649**	**0.8808**	**0.2120**

Table 3: Results on Travel Time Estimation.

Travel Time Estimation

In the above experiments, it is demonstrated that GDCF performs well on three basic crowd flow tasks. Moreover, another advantage of GDCF is that the node representations generated in phase 1 rely on reconstruction, which is generic to more tasks. In this section, an experiment for travel time estimation on NYTaxi is designed to demonstrate the generality of the learned node representations. This experiment aims to predict the travel time for each trip with the representation of the origin and the destination. The experiments are based on four models, which have the same structure but take different node representations as input: **R-TTE** generates random vectors to represent nodes; **S-TTE** represents nodes by static one-hot encoding; **OD-TTE** represents nodes as the corresponding row of the OD matrix, which simply incorporates crowd flow information; **GDCF-TTE** takes node representations from phase 1 to estimate the travel time. The result is shown in Table 3; it can be observed that: (1) OD-TTE performs better than R-TTE and S-TTE. This demonstrates that leveraging the crowd flow information is generally beneficial to potential and meaningful tasks. (2) GDCF-TTE outperforms OD-TTE, which indicates that instead of merely introducing crowd flow information, node representations from GDCF can contain more useful patterns through the encoder-decoder architecture. The great performance on this task also demonstrates the generality of the learned node representations.

Influence of Training Data Ratio

As the model has been already trained to compress the crowd flow information into meaningful node representation in phase 1. In phase 2, the model can be well-trained by only a small ratio of data. To demonstrate this, an experiment is conducted on BJSubway by changing the number of training days in phases 2 to 1, 2, 3, 4, 5, 6, 7, 14, 21, 28, and 35. The results are shown in Figure 5, it can be observed that: (1) Based on node representations in phase 1, GDCF using

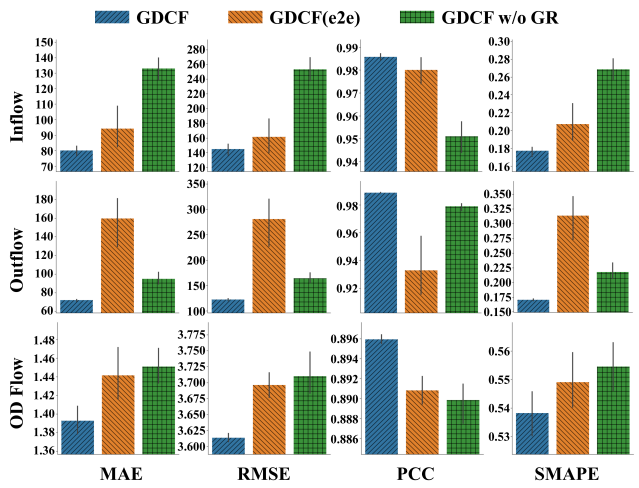

Figure 4: Ablation Study.

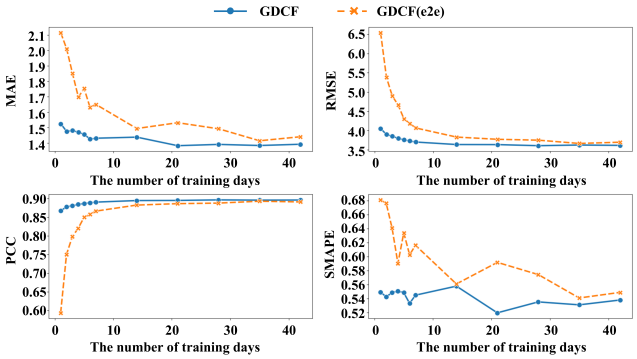

Figure 5: Influence of Training Ratio.

only 7-day training data in phase 2 can achieve comparable performance with 42-day training data. This indicates that the model can compress useful patterns into node representations and meanwhile effectively eliminates noises in little training data. (2) Compared with GDCF (e2e), GDCF using the same ratio of training data in phase 2 can perform better by a big gap. This characteristic is extremely useful in some scenarios. For instance, detailed transaction data is always unavailable due to privacy. By leveraging the proposed encoder-decoder architecture, the department with the private data can learn evolving node representations and release them for public use without sharing the raw data.

Related Work

Crowd Flow Modeling

With the development of neural networks, deep learning-based crowd flow modeling has been widely studied in recent years. Most of the existing work focuses on designing the spatial and temporal convolution mechanism to aggregate information from nearby nodes and historical observations for specific prediction tasks. ST-ResNet split the studied area into grids and introduced CNN-based residual networks to learn the spatial relations between grids to predict their inflow and outflow (Zhang, Zheng, and Qi 2017). Some other studies, including CoST-Net (Ye et al. 2019), DeepSTN+ (Lin et al. 2019), DMVST-Net (Yao et al. 2018) and STDN (Yao et al. 2019), also followed a similar formalization and designed CNN-based spatial modules. Noticing the limitation of gridding, some researchers proposed GNNs to predict inflow and outflow (Geng et al. 2019; Liu et al. 2021; Ye et al. 2021). And there were also some methods to solve OD flow prediction with CNNs (Wang et al. 2020; Liu et al. 2019a) or GNNs (Wang et al. 2019; Zhang et al. 2021; Shi et al. 2020; Liu et al. 2022). For temporal relations, previous studies always process historical data as snapshots and leverage RNNs, fusion components, or attention-based modules to cope with it. However, they are generally trained in an end-to-end manner for only a certain subset of tasks.

Graph Representation Learning

Graph representation learning aims to represent the nodes or edges in a given graph as vectors. The research starts by representing the node in static graphs, where the graphs remain the same over time (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Velickovic et al. 2018; Xu et al. 2019). However, these methods cannot handle the situation in which nodes or edges would change. In recent years, some researchers shift to the field of dynamic graph representation learning. A branch of them represented a dynamic graph as multiple graph snapshots with discrete time features, which is still inflexible to fit general cases (Goyal et al. 2018; Singer, Guy, and Radinsky 2019; Sankar et al. 2020; Pareja et al. 2020). Another branch viewed time as a continuous feature, which treats the dynamic graph as streaming timestamped events, to generate node representations (Nguyen et al. 2018; Trivedi et al. 2019; Kumar, Zhang, and Leskovec 2019; Liu et al. 2019b; Xu et al. 2020; Rossi et al. 2020). Although these methods behaved well in many tasks, how to introduce this idea into crowd flow modeling, where the events are much denser and global spatial relations between nodes are influential, has not been explored.

Conclusion

This study proposed a generic and dynamic representation learning framework for crowd flow modeling. Different from previous studies, which focused on designing spatial and temporal convolutional blocks, the framework encodes crucial information in crowd flow into evolving node representations. Then the learned node representations can be retrieved and utilized in multiple downstream tasks. To achieve that, an encoder and a decoder were specially designed to ensure the node representations contain as much information as possible. And an adaptive global learning module was designed to exploit spatial dependency between traffic nodes. Experiments on two large-scale datasets demonstrated the effectiveness and the robustness of the proposed model. Studies on travel time estimation and varying training data demonstrated the ability to adapt to varied situations. The idea, of learning node representations and unifying multiple tasks in a general framework, shed new light on this subject. And how to leverage the idea to balance data privacy and public use could be also explored in future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62272023, 51991395, U21A20516).

References

Chen, T.; He, T.; Benesty, M.; Khotilovich, V.; Tang, Y.; Cho, H.; Chen, K.; et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4): 1–4.

Geng, X.; Li, Y.; Wang, L.; Zhang, L.; Yang, Q.; Ye, J.; and Liu, Y. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 3656–3663.

Goyal, P.; Kamra, N.; He, X.; and Liu, Y. 2018. DynGEM: Deep Embedding Method for Dynamic Graphs. *CoRR*, abs/1805.11273.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kumar, S.; Zhang, X.; and Leskovec, J. 2019. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 1269–1278. ACM.

Lin, Z.; Feng, J.; Lu, Z.; Li, Y.; and Jin, D. 2019. Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 1020–1027.

Liu, H.; Wu, Q.; Zhuang, F.; Lu, X.; Dou, D.; and Xiong, H. 2021. Community-aware multi-task transportation demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 320–327.

Liu, L.; Qiu, Z.; Li, G.; Wang, Q.; Ouyang, W.; and Lin, L. 2019a. Contextualized Spatial-Temporal Network for Taxi Origin-Destination Demand Prediction. *IEEE Trans. Intell. Transp. Syst.*, 20(10): 3875–3887.

Liu, L.; Zhu, Y.; Li, G.; Wu, Z.; Bai, L.; and Lin, L. 2022. Online Metro Origin-Destination Prediction via Heterogeneous Information Aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, X.; Hsieh, P.; Duffield, N.; Chen, R.; Xie, M.; and Wen, X. 2019b. Real-Time Streaming Graph Embedding Through Local Actions. In *Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 285–293. ACM.

Nguyen, G. H.; Lee, J. B.; Rossi, R. A.; Ahmed, N. K.; Koh, E.; and Kim, S. 2018. Continuous-Time Dynamic Network Embeddings. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon, France, April 23-27, 2018*, 969–976. ACM.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T. B.; and Leiserson, C. E. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, 5363–5370. AAAI Press.

Rossi, E.; Chamberlain, B.; Frasca, F.; Eynard, D.; Monti, F.; and Bronstein, M. 2020. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

Sankar, A.; Wu, Y.; Gou, L.; Zhang, W.; and Yang, H. 2020. DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks. In *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*, 519–527.

Shi, H.; Yao, Q.; Guo, Q.; Li, Y.; Zhang, L.; Ye, J.; Li, Y.; and Liu, Y. 2020. Predicting Origin-Destination Flow via Multi-Perspective Graph Convolutional Network. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, 1818–1821. IEEE.

Singer, U.; Guy, I.; and Radinsky, K. 2019. Node Embedding over Temporal Graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, 4605–4612. ijcai.org.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. DyRep: Learning Representations over Dynamic Graphs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Wang, S.; Miao, H.; Chen, H.; and Huang, Z. 2020. Multi-task Adversarial Spatial-Temporal Networks for Crowd Flow Prediction. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, 1555–1564. ACM.

Wang, Y.; Yin, H.; Chen, H.; Wo, T.; Xu, J.; and Zheng, K. 2019. 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*, 1227–1235.

Xu, D.; Ruan, C.; Körpeoglu, E.; Kumar, S.; and Achan, K. 2020. Inductive representation learning on temporal graphs. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Yao, H.; Tang, X.; Wei, H.; Zheng, G.; and Li, Z. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 5668–5675.

Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; and Li, Z. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the Thirty-second AAAI Conference on Artificial Ontelligence*, volume 32.

Ye, J.; Sun, L.; Du, B.; Fu, Y.; Tong, X.; and Xiong, H. 2019. Co-prediction of multiple transportation demands based on deep spatio-temporal neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 305–313.

Ye, J.; Sun, L.; Du, B.; Fu, Y.; and Xiong, H. 2021. Coupled layer-wise graph convolution for transportation demand prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 4617–4625.

Zhang, D.; Xiao, F.; Shen, M.; and Zhong, S. 2021. DNEAT: A novel dynamic node-edge attention network for origin-destination demand prediction. *Transportation Research Part C: Emerging Technologies*, 122: 102851.

Zhang, J.; Zheng, Y.; and Qi, D. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.