Transfer Learning in Traffic Prediction with Graph Neural Networks

Yunjie Huang, Xiaozhuang Song, Shiyao Zhang, *Member, IEEE*, and James J.Q. Yu, *Senior Member, IEEE*

***Abstract*— Statistics on urban traffic speed flows are essential for thoughtful city planning. Recently, data-driven traffic prediction methods have become the state-of-the-art for a wide range of traffic forecasting tasks. However, many small cities have a limited amount of traffic data available for building data-driven models due to lack of data collection methods. With the acceleration of urbanization, the need for traffic construction of small and medium-sized cities is imminent. To tackle the above problems, we propose a TransfEr lEarning approach with graPh nEural nEtworks (TEEPEE) for traffic prediction that can forecast the traffic speed in data-scarce areas with massive value data from developed cities. In particular, TEEPEE uses graph clustering to divide the traffic network map into multiple sub-graphs. Graph clustering captures more spatial information in the transfer process. To evaluate the effectiveness of TEEPEE, we conduct experiments on two real-world datasets and compare them with other baseline models. The results demonstrate that TEEPEE is among the best efforts of baseline models. We provide a comprehensive analysis of the experimental results in this work.**

I. Introduction

Rapid urbanization development has modernized people's lives. However, it also brings significant problems to modern cities, such as traffic congestion, environmental pollution, and exhausting land usage [1]. The proliferation of big data and the rapid development of computing power offer the possibility of using data science and computation technology to solve these problems. Urban computing aims to build smart cities by using massive data created in cities. Although data-driven urban computing has emerged due to the proliferation of data, there are still many cities that lack data. This problem of data scarcity can be mainly attributed to the high cost for constructing a comprehensive system of city-wide traffic sensors and the requirement of considerable time to collect data [2]. One available approach is to use inter-city transfer learning to help cities that lack data develop their smart systems.

The concept of inter-city transfer learning is proposed in [3]. The data usage in inter-city transfer learning can be divided into three strategies [4]. The first is "Cross-City" which learns knowledge to city with sufficient data, and transfer this knowledge to city with insufficient data. This way has developed good results in tasks such as air quality prediction [3], ridesharing detection [5], and chain shop site recommendation [6], etc. The second is "Cross-Modality" which uses the different types of data in the city such as weather, daytime, and social check-in information. For instance, the popularity of social network check-ins may be an indicator of the density of physical crowd flow in [7]. And the last is "Combining Cross-Modality and Cross-City Transfer" which includes both above strategies to transfer knowledge.

The task of traffic speed prediction is a challenge for the construction of smart city systems. Autoregressive Integrated Moving Average (ARIMA) and Historical Average (HA) are the traditional methods for speed prediction but they are limited on non-stationary sequence. Although data-driven models can make better use of big data than ARIMA and HA, this task also faces the same problem of data scarcity in cities. In previous work, despite there has been researched work on the spatio-temporal forecasting tasks with the transfer method, there is a gap in traffic speed prediction. The first related research [2] is only compared with the traditional method Autoregressive Integrated Moving Average (ARIMA) and Historical Average (HA). To further demonstrate the effectiveness of transfer learning in few-data traffic forecasting, we propose TEEPEE. The main contributions of this work are as follows.

- We propose a TransfEr lEarning approach with graPh nEural nEtworks (TEEPEE) for traffic prediction. TEEPEE is able to provide traffic forecasts for cities with sparse data by learning from the sufficient data of other cities.
- We have adopted a graph clustering approach in TEEPEE to capture spatial information in the traffic road networks. TEEPEE divides the collected data into sub-graphs based on spatial attributes and combine them with a Graph Convolutional Network (GCN). Then, the model trained with source data collected in data-rich cities is applied to cities with fewer data.
- We provide a comprehensive experiments on two real-world datasets and compare the performance of TEEPEE with other baseline models. The experimental results demonstrate that TEEPEE's effectiveness in improving traffic prediction accuracy in small cities with insufficient data.

The rest of this paper is organized as follows. Section II presents the related work on traffic speed prediction. The definition of the problem is presented in Section III. Section IV presents the main techniques used in TEEPEE. Besides,

This work is supported by the Stable Support Plan Program of Shenzhen Natural Science Fund No. 20200925155105002, by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032, and by the Guangdong Provincial Key Laboratory (Grant No. 2020B121201001). Yunjie Huang, Xiaozhuang Song, and James J.Q. Yu are with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China. Shiyao Zhang is with the Academy for Advanced Interdisciplinary Studies, Southern University of Science and Technology, Shenzhen, China. James J.Q. Yu is the corresponding author.

we also introduce the main structure of TEEPEE in this section. In Section V, we conduct several experiments to demonstrate the effectiveness of TEEPEE. Finally, Section VI concludes this work.

II. Related Work

In this section, we review the previous work in traffic prediction. We also introduce representative efforts on transfer learning in cross-city prediction tasks.

A. Traffic speed prediction

Previous work on this topic can be broadly divided into two main categories [8]: classical statistical methods and machine learning models (especially deep learning). At the early stage, statistical methods are used to study traffic systems. However, the ability of these models to process highly complex non-linear time-series data is quite limited. With the increasing range and diversity of traffic data, data-driven traffic prediction methods are shown to be promising, which outperforms traditional simulation-based methods.

Traditionally, spatio-temporal forecasting uses basic time-series models such as Autoregressive Integrated Moving Average (ARIMA) [9], Kalman filters and its variants, regression models with spatio-temporal regularisation, and Support Vector Regression (SVR), etc. The machine learning methods mentioned above have achieved good results in classification and regression tasks using big data. However, the methods need feature processing artificially. As the number of data increases, some data-hungry deep learning models are gradually starting to be used in traffic prediction tasks.

Deep learning methods can extract high-dimensional features from massive data effectively. Data-driven deep learning traffic prediction models, such as Deep Belief Networks (DBN) [10] and Stacked AutoEncoders (SAE) [11], have shown their superior performance on traffic flow prediction (traffic forcasting tasks). Recently, Recurrent Neural Networks (RNN) have been widely adopted in time sequence forecasting tasks for their performance on temporal dependency modeling. With such models, [12] formulates the temporal dependency for traffic prediction tasks. On the other hand, a traffic network is a structure with high spatial correlation. Hence, to model the complex spatial dependency of traffic networks is of critical importance for traffic prediction models as well. Convolutional Neural Networks (CNN) that capture spatial relationship is therefore applied in traffic problems; see [13], [14] for some examples. However, traffic data, unlike images, are non-Euclidean. Consequently, CNN may not be a good predictor of traffic data. More recently, researchers have started to extend the convolution operator to the more general graph structures [15]. Compared to the grid structure, graph structure better resembles the topology of traffic network. Moreover, representing traffic networks as graphs allow models to capture fine-grained spatial dependency in traffic networks.

B. Transfer learning

Although deep learning is developed due to the cities' data proliferation, there are still cities with insufficient data. Therefore, transfer models are proposed to solve data scarcity problems. The work [3] propose the model "FLORAL" – an inter-city transfer model to alleviate the data deficit in small cities. FLORAL is defined as a flexible multimodal transfer learning method that can transfer knowledge from cities with enough model data and labels to cities where labels are scarce. [16] proposes RegionTrans which explores the possibility of spatio-temporal transfer across cities. Xu [17] and Lin [2] use SVM and dynamic time wrapping in cross-city speed prediction, respectively. [18] proposed TL-DCRNN – a speed prediction method with transfer learning. [19] proposes a potential information transfer mechanism for online urban traffic speed estimation with little historical data.

In this work, we propose TransfEr lEarning approach with graPh nEural nEtworks (TEEPEE), a model for traffic prediction that can forecast the traffic speed in data-scarce cities with massive value data from data-rich cities.

III. Problem Definition

The problem of using transferable knowledge to predict traffic speed is defined as follows. Let S represent a Source city with abundant data, and T represents a Target city with little or even no data. $G^S = (V^S, E^S, A^S)$ is a weighted graph of the source city's traffic road network, where V^S is the set of N^S nodes in the city's road network. E^S is the set of directed edges connecting these nodes, and $A^S = \{A^S_{ij}\} \in \mathbb{B}^{N^S \times N^S}$ is an adjacency matrix representing the connectivity between nodes in the traffic road network. If there exists a connection from nodes N_i to N_j, but not from N_j to N_i, then $A^S_{ij} = 1$, $A^S_{ji} = 0$.

The traffic state of the source city at time step t can be represented as a graph state $X^S_t \in \mathbb{R}^{N^S \times F}$, where F is the number of traffic features we interested in. In this paper, we only focus on the speed, i.e., F equals 1. Given H historical traffic states $X^S = (X^S_{t_1}, X^S_{t_2}, ..., X^S_{t_H}) \in \mathbb{R}^{H \times N^S \times F}$ to train a model for predicting the next Q time steps' traffic state on the source graph.

$G^T = (V^T, E^T, A^T)$ is the graph with N^T nodes that represents the target city's road network with limited data. The meaning of V^T, E^T, A^T is the same as the V^S, E^S, A^S in the source city but now for the target city. Given H current traffic states $X^T = (X^T_{t_1}, X^T_{t_2}, ..., X^T_{t_H}) \in \mathbb{R}^{H \times N^T \times F}$ on the graph G^T, these states are used as inputs to trained model mentioned earlier. The outputs are the next Q time steps' states $Y = (X^T_{t_{H+1}}, X^T_{t_{H+1}}, ..., X^T_{t_{H+Q}}) \in \mathbb{R}^{Q \times N^T \times F}$. Note that, the number of nodes in the source and target city's graph are not the same, i.e., $N^S \neq N^T$.

IV. Proposed Methodology

In this section, we describe the methods we use in TEEPEE: graph partitioning and spatial dependency modeling. We introduce the framework of TEEPEE at the end of this section.

A. Graph Partitioning

There are several graph partitioning methods that can be used to partition the graph during the training period. Usually, these methods cannot divide the graph into exactly equal-size sub-graphs, but they can give partitions of similar sizes. In this paper, we take the k-way graph partitioning method of Metis [20] and partition the traffic graph of our target city into M^T similarly sized sub-graphs. In the case that the number of nodes can not be divided by M^T, the 0-completion method in [18] can be used. In addition, an example of how to represent a city's road graph is shown in Fig. 1. There are two traffic roads, a two-way road, and a one-way road. The segments of each road are labeled from 1 to 5, and they are defined as the nodes of the graph. For node 1 (road segment 1), there is no node connecting to it directly. For node 2 (road segment 2), there are two nodes, i.e., 3 and 4, connecting to it. Therefore, there are edges (3, 2), (4, 2), respectively, and $A_{32} = A_{42} = A_{15} = 1$.

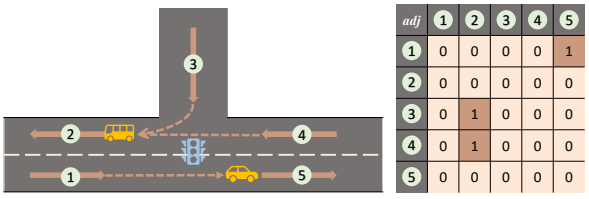

Fig. 1. Graph representation of road network

However, since the Metis method requires symmetric adjacency matrices, we use a simple symmetrization method on the adjacency matrices when performing graph partitioning as follows:

$$A_{ij} = A_{ji} = \begin{cases} 1 & \text{if } A_{ij} = 1 \text{ or } A_{ji} = 1 \\ 0 & \text{otherwise.} \end{cases} . \quad (1)$$

B. Spatial Dependency Model

GCN is effective to build the spatial dependency in traffic networks. It overcomes the disadvantage of CNN that is only deal with euclidean spatial data (i.e., images and grids [21]). As graph structure better reflects the topology of traffic network, GCN can capture fine-grained spatial features for traffic network.

The essential goal of GCN is to extract the spatial features of traffic networks. A common solution is to adopt the spectral graph convolution, which is based on the theory of spectral graphs and uses spectral clustering to construct filters in the Fourier domain to investigate the properties of the graph by means of the eigenvalues and eigenvectors of the graph's Laplace matrix [22]. The Laplace matrix is symmetric. And the matrix has non-zero elements only at the vertices themselves and at their 1-hop neighbors. These advantages make the convolution process simpler. Therefore, GCN is popular in many areas.

We first compute the graph Laplacian matrix with the adjacency matrix.

$$L = D^{-\frac{1}{2}} \widetilde{A} D^{-\frac{1}{2}}, \quad (2)$$

where $\widetilde{A} = A + I_N$ is the adjacency matrix with added self-connections, I_N is the identity matrix, $D = diag(\sum_j A_{ij})$ is the diagonal degree matrix.

For each convolutional layer,

$$H^{(l+1)} = \sigma(LH^{(l)}\theta^{(l)}), \quad (3)$$

where l is the layer number. $H^{(l)}$ stands for the output of the l-th layers. $\theta^{(l)}$ represents the l-th layer's parameters. And $\sigma(\cdot)$ is the sigmoid function. Combined with Equation 3, the output is calculated as follows,

$$f(X, A) = \sigma(L\,\mathrm{ReLU}(LXW_0)W_1), \quad (4)$$

where $f(X, A)$ is the output. X is the feature matrix. W_0 stands for the weight matrix mapping input to the hidden unit, and W_1 represents the weight of the next layer. $\mathrm{ReLU}(\cdot)$ represents the Rectified Linear Unit [23]. In summary, we use the GCN model to learn spatial information in traffic netwroks.

C. Framework of TEEPEE

Although we have emphasized the importance of obtaining spatial relationship for traffic data in the above, traffic speed prediction is essentially a time series forecasting task. So, the model which works on time series data is necessary. In TEEPEE, we use Gate Recurrent Unit (GRU) to capture temporal information. GRU is a variant of RNN. It alleviates the disadvantages (i.e., gradient explosion and gradient vanishing) of RNN. And compared with another variant of RNN (i.e., Long-Short Term Memory, LSTM), GRU has fewer parameters and a simpler structure. We combine GCN and GRU in the ST-block, which was mentioned later. TEEPEE's framework is shown in Fig. 2. There are several parts in TEEPEE.

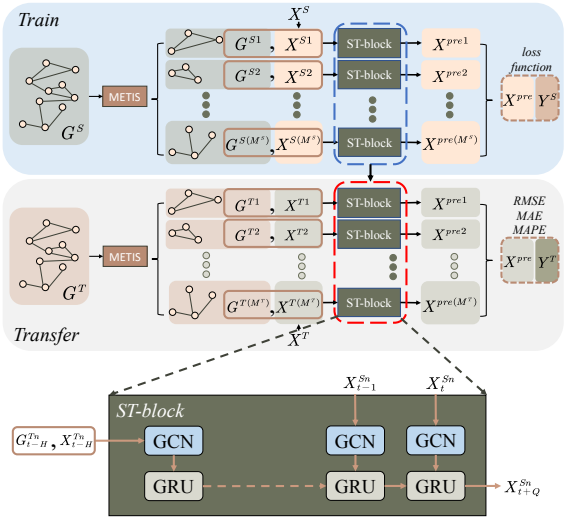

Fig. 2. The framework of TEEPEE

1) Partition: First, we partition the target city's traffic road graph (G^T) in M^T sub-graphs ($G^{T1}, G^{T2}, ...G^{T(M^T)}$). Then, we partition the source city's traffic road graph (G^S) into sub-graphs ($G^{S1}, G^{S2}, ...G^{S(M^S)}$) with similar sized as target city's sub-graphs.

2) Train process: In the training process, we combine each sub-graph with the H historical traffic states ($X^{S1}/X^{S2}/.../X^{S(M^S)}$) on the sub-graph as one pair of inputs for ST-block (mentioned later). The outputs of ST-blocks are the next Q time steps' traffic states for each sub-graph and concatenate them as prediction result of the whole source city's graph (G^S). Then, we make a loss with the ground-truth value.

3) Transfer process: When performing inter-city transfer, we also combined each sub-graph of the target city's traffic road graph with the H current traffic states ($X^{T1}/X^{T2}/.../X^{T(M^T)}$) as a pair of inputs. But in this part, we use the ST-blocks trained by the source city's data directly. The next steps are the same as the training process. Only at the end, we compute three metrics (mentioned in Section V-C) with ground-truth value.

4) ST-block: ST-block is a spatio-temporal network that combines GCNs and GRUs. Its inputs are H pairs of sub-graph and traffic states. GCNs capture the spatial information of each pair and feed them into GRUs. Then, GRUs capture the temporal information of these pairs. The output is the next Q time steps' traffic states on one sub-graph.

V. Experiment

In this section, we evaluate the performance of the proposed model on two real-world datasets. In particular, we employed two variants of the proposed TEEPEE to verify the effectiveness of TEEPEE's components.

a) TEEPEE: We use this original model to verify the performance of the transfer learning so that the data of the target city will not be used in the transfer directly.

b) TEEPEE_retrain: TEEPEE_retrain is a variant of TEEPEE. The difference between TEEPEE and TEEPEE_retrain is that the input data in TEEPEE_retrain additionally includes a subset from the target city's traffic dataset besides the data from the source city. The subset contains limited data (e.g., one day). In this case, we expect TEEPEE_retrain's performance approach to the prediction results of having a large amount of data from the target city for training.

Furthermore, we verify the performance of TEEPEE and TEEPEE_retrain by comparing them with other baseline models on the traffic speed forecasting task.

A. Experimental Settings

a) Experiment Environment: All experiments are conducted on a Linux server with Intel E5-2620v4 CPU and GeForce RTX 2080Ti CPU. All baselines and the proposed TEEPEE are built with Pytorch 1.7.0 and Python 3.8.3.

b) Hyperparameter Settings: In accordance with the previous works, we use H = 12 traffic conditions with a short history (60 min) to predict future traffic conditions and Q = 3 / 6 / 9 / 12 (15 min / 30 min / 45 min / 60 min). We train our model with the Adam optimizer [24]. The initial learning rate is set to 0.001. The batch size is 64. The training epoch for all models is 500. And we set the number of the hidden unit as 32. The number of sub-graphs into which the target city is divided is 4.

B. Dataset Description

In this work, two real-world network-wide traffic speed datasets are utilized.

a) Nav-BJ: This dataset consists of the average speed of Beijing from March 1st to March 31st of 2019. There are 1159 nodes (road segments) of Beijing as the investigated data. The experimental data contains two main matrices. One is the 1159×1159 adjacency matrix, which describes the spatial relationship between the nodes. Each value in the matrix represents whether the node represented by the row is connected to the node represented by the column. The other is a feature matrix, where each row represents the speed of all nodes in a traffic graph at a certain time. Each column represents the speed at a node from the start time to the end. The timestamp interval is set to five minutes.

b) Nav-SH: This dataset consists of the average speed of Shanghai at the same time as Nav-BJ. There are 400 nodes (road segments) of Shanghai for prediction. It also has two matrices as Nav-BJ. The first is a 400×400 adjacency matrix. The next is the feature matrix as in Nav-BJ.

In this paper, through the cross-validation process, we consider the dataset from Beijing as a large dataset (i.e., the source dataset in transfer learning). At the same time, Shanghai is identified as the target city for knowledge transfer. Its data is not used for training but only serves as real-time input of TEEPEE. In TEEPEE_retrain, we use one day's subset of Nav-SH to retrain the parameters in TEEPEE. We follow the previous deep learning research and map the data with the range $(0, 1)$ using a sigmoid activity function after the Z-score normalization.

C. Evaluation Metrics

We choose three metrics to evaluate the performance of models, including:

1) Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}\left|(Y_i - \hat{Y}_i)\right| \tag{5}$$

2) Root Mean Squared Error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(Y_i - \hat{Y}_i)^2} \tag{6}$$

3) Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\widehat{Y}_i - Y_i}{Y_i}\right| \tag{7}$$

Where Y_i and $\widehat{Y}_i$ are ground truth and predicted speed, respectively.

D. Baseline Models

We compare TEEPEE with two categories of methods: non-transfer learning and transfer learning methods. For non-transfer learning methods, we select the TGCN model [25], which is among the state-of-the-art for traffic prediction.

1) Non-transfer learning:

TABLE I

PERFORMANCE COMPARISON OF DIFFERENT APPROACHES FOR TRAFFIC PREDICTION ON SHANGHAI DATASETS.

method		TGCN_L	TGCN_S	TEEPEE_direct	TEEPEE	TEEPEE_retrain
15min	RMSE	7.96	9.75	18.98	9.83	**8.88**
	MAE	5.79	7.22	13.38	7.31	**6.57**
	MAPE	7.13	8.95	15.96	8.88	**8.07**
30min	RMSE	8.12	9.47	18.42	10.18	**8.99**
	MAE	5.91	7.04	13.26	7.65	**6.64**
	MAPE	7.27	8.64	16.19	9.33	**8.13**
45min	RMSE	8.21	9.75	18.33	10.29	**9.31**
	MAE	5.98	7.26	13.06	7.71	**6.89**
	MAPE	7.35	8.90	15.62	9.31	**8.50**
60min	RMSE	8.29	9.95	19.23	10.31	**9.33**
	MAE	6.06	7.43	13.81	7.73	**6.92**
	MAPE	7.38	8.96	17.76	9.47	**8.37**

TABLE II

PERFORMANCE COMPARISON OF THE NUMBER OF CLUSTERS ON NAV-SH

M^T	15min			30min			45min			60min		
	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE
2	10.67	7.85	9.48	10.21	7.69	9.43	10.34	7.79	9.48	10.45	7.88	9.68
4	9.83	7.31	**8.88**	10.18	7.63	9.32	10.28	7.71	9.31	10.31	7.73	9.46
8	**9.76**	**7.28**	8.92	**9.97**	**7.45**	**9.14**	**10.06**	**7.51**	**9.09**	**10.09**	**7.56**	**9.28**
16	9.89	7.37	9.04	10.12	7.56	9.33	10.11	7.55	9.17	10.23	7.67	9.41

a) TGCN_L: The model uses a subset from the target city's traffic dataset (Nav-SH) as the training dataset. The subset contains 20 days of data, and we consider that there is a rich source of data that can be used to predict the traffic speed in the next 15–60min.

b) TGCN_S: The model uses a small subset from the target city's traffic dataset (Nav-SH) as the training dataset. Different from the TGCN_L, the subset contains one day of data which is the same as in TEEPEE_retrain. In this case, we consider the subset is small to predict the traffic speed in the next 15–60min difficultly.

2) Transfer learning:

a) TEEPEE_direct: The model uses all data from the source city (Nav-BJ) as a training set to make predictions of the target city. Unlike TEEPEE, TEEPEE_direct does not use graph partitioning; the model is intended to demonstrate the performance of partitioning.

E. Results

1) Forecasting Performance Comparison: Table I shows a comparison of different methods for 15 min / 30 min / 45 min / 60 min ahead prediction on the Nav-SH. We observe that:

- Compared to TGCN_S, TEEPEE without any data from the target city is already comparable to the performance of the TGCN_S with only a small amount of data, which indicates that inter-city data transfer is possible.
- Compared to TGCN_L, TEEPEE_retrain's performance is slightly inferior but better than TEEPEE. The reason for this is that TEEPEE_retrain uses a small subset from the target city. This shows that our transfer method, with only a small subset from target city data, can approach the results of those who employ a large amount of data for training. And it is superior to the results of a model trained with only a small amount of data (TGCN_S). This demonstrates the effectiveness of the transfer.
- Compared to TEEPEE_direct, our method is able to capture more information in the traffic network map because of the graph partitioning, which improves the metrics (MSE, MAE, MAPE) over the direct transfer method.

2) Missing Data Tolerance: Traffic data is often collected from sensors arranged on various roads. Due to the reasons of weather and environment, etc., sensors may not generate the correct readings. Additionally, there is a non-negligible probability of packet loss and other problems during data transmission. The data sent back by sensors is not always complete. So it needs to concern about how prediction methods perform in the missing data. To evaluate the performance of TEEPEE in the missing data, we set a random loss fraction η (percentage of missing data, range from 10% to 90%) of the target dataset and produced corresponding input for predicting the next hour's traffic speed. As shown in Fig. 3, TEEPEE's approach is more missing-tolerant, suggesting that TEEPEE can capture complex and incomplete spatio-temporal correlations from contaminated traffic data to adjust the dependence between observations and future time steps.

3) Sensitivity of Clusters: In TEEPEE, the traffic network in the target city is divided into several sub-graphs (i.e., clusters) with different sizes. The number of clusters may influence the performance metrics. In this experiment, we set the number of clusters as 2, 4, 8, and 16, respectively, to see how it influences the metrics. In Table II, We can observe that the best results are obtained using the setting of $M^T = 8$. When $M^T = 4$ and $M^T = 6$, the performance is undermined, though not significant. However, when M^T equals 2, the performance is much inferior. This may be due to the fact that the two cities do not have exactly

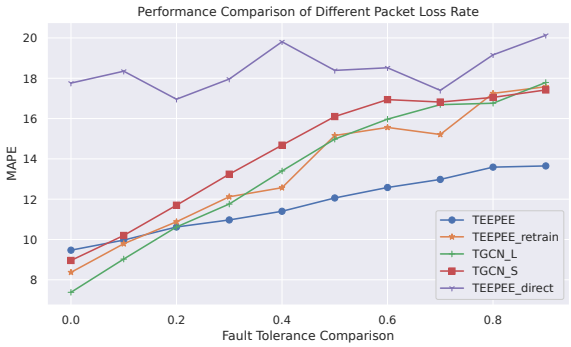

Fig. 3. Performance of the model at different loss fractions

the same traffic network and data distribution. The aim of transfer learning is to learn a similar graph and temporal structure from the source city to the target city. After graph partitioning, we consider that each sub-graph represents a traffic structure (i.e., a fork in the road, roundabout). These structures exist in each city. When the traffic graph is too large, a graph may conclude a variety of different traffic structures. The similarity between the two cities is undermined. And the performance of transfer is inferior. When we partition the graph in a suitable size that contains one traffic structure or several similar structures, the spatial and temporal information captured by the model can better transfer to the target city.

VI. Conclusions

In this paper, we propose TransfEr lEarning approach with graPh nEural nEtworks (TEEPEE), a model for traffic speed prediction with transfer learning. TEEPEE can provide traffic forecasts for cities with scarce data by learning from the sufficient data of other cities. In TEEPEE, we use graph partitioning to obtain sub-graphs that contain similar traffic structures of target cities from the source city. Then, we use GCN to capture the spatial information on the sub-graphs and GRU for temporal information for predicting traffic speed. This transfer learning method fills a gap in traffic speed prediction with deep learning in cities with limited data. The experiments show that TEEPEE with no target city's data used is comparable to the model trained with limited data (TGCN_S). And when using the target city's data, the model (TEEPEE_retrain) can outperform TGCN_S and approach to TGCN_L. In summary, TEEPEE is effective on few-data traffic speed prediction with transfer learning.

References

[1] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 3, p. 38, 2014.

[2] B. Y. Lin, F. F. Xu, E. Q. Liao, and K. Q. Zhu, "Transfer learning for traffic speed prediction: A preliminary study," in *Proc. Workshops Thirty-Second AAAI Conf. Artificial Intelligence*, 2018, pp. 174–177.

[3] Y. Wei, Y. Zheng, and Q. Yang, "Transfer knowledge between cities," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 1905–1914.

[4] L. Wang, B. Guo, and Q. Yang, "Smart city development with urban transfer learning," *IEEE Computer*, vol. 51, no. 12, pp. 32–41, 2018.

[5] L. Wang, X. Geng, J. Ke, C. Peng, X. Ma, D. Zhang, and Q. Yang, "Ridesourcing car detection by transfer learning," *arXiv preprint arXiv:1705.08409*, 2017.

[6] B. Guo, J. Li, V. W. Zheng, Z. Wang, and Z. Yu, "CityTransfer: Transferring Inter- and Intra-City Knowledge for Chain Store Site Recommendation Based on Multi-Source Urban Data," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 1, no. 4, pp. 1–23, Jan. 2018.

[7] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location-based social networks," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 3, p. 30, 2016.

[8] A. Agafonov, "Traffic flow prediction using graph convolution neural networks," in *2020 10th International Conference on Information Science and Technology (ICIST)*, 2020, pp. 91–95.

[9] S. Makridakis and M. Hibon, "Arma models and the box–jenkins methodology," *Journal of Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[10] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.

[11] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[12] X. Cheng, R. Zhang, J. Zhou, and W. Xu, "Deeptransport: Learning spatial-temporal dependency for traffic condition forecasting," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[13] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction." *Sensors*, vol. 17, no. 4, p. 818, 2017.

[14] C. Chen, K. Li, S. G. Teo, G. Chen, X. Zou, X. Yang, R. C. Vijay, J. Feng, and Z. Zeng, "Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction," in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 893–898.

[15] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *IJCAI'18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3634–3640.

[16] L. Wang, X. Geng, X. Ma, F. Liu, and Q. Yang, "Cross-city transfer learning for deep spatio-temporal prediction," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, S. Kraus, Ed., 2019, pp. 1893–1899.

[17] F. F. Xu, B. Y. Lin, Q. Lu, Y. Huang, and K. Q. Zhu, "Cross-region traffic prediction for china on openstreetmap," in *Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, 2016, pp. 37–42.

[18] T. Mallick, P. Balaprakash, E. Rask, and J. MacFarlane, "Transfer learning with graph neural networks for short-term highway traffic forecasting." *arXiv preprint arXiv:2004.08038*, 2020.

[19] J. J. Yu, "Online traffic speed estimation for urban road networks with few data: A transfer learning approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 4024–4029.

[20] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[21] P. Krishnakumari, H. van Lint, T. Djukic, and O. Cats, "A data driven method for od matrix estimation," *Transportation Research Part C-emerging Technologies*, vol. 113, pp. 38–56, 2019.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR (Poster)*, 2016.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[24] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.

[25] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.