Transferable Graph Structure Learning for Graph-based Traffic Forecasting Across Cities

Yilun Jin
Hong Kong University of Science and Technology
Hong Kong SAR, China
yilun.jin@connect.ust.hk

Kai Chen
Hong Kong University of Science and Technology
Hong Kong SAR, China
kaichen@cse.ust.hk

Qiang Yang
Hong Kong University of Science and Technology
Hong Kong SAR, China
WeBank
Shenzhen, China
qyang@cse.ust.hk

ABSTRACT

Graph-based deep learning models are powerful in modeling spatio-temporal graphs for traffic forecasting. In practice, accurate forecasting models rely on sufficient traffic data, which may not be accessible in real-world applications. To address this problem, transfer learning methods are designed to transfer knowledge from the source graph with abundant data to the target graph with limited data. However, existing methods adopt pre-defined graph structures for knowledge extraction and transfer, which may be noisy or biased and negatively impact the performance of knowledge transfer. To address the problem, we propose TransGTR, a transferable structure learning framework for traffic forecasting that jointly learns and transfers the graph structures and forecasting models across cities. TransGTR consists of a node feature network, a structure generator, and a forecasting model. We train the node feature network with knowledge distillation to extract city-agnostic node features, such that the structure generator, taking the node features as inputs, can be transferred across both cities. Furthermore, we train the structure generator via a temporal decoupled regularization, such that the spatial features learned with the generated graphs share similar distributions across cities and thus facilitate knowledge transfer for the forecasting model. We evaluate TransGTR on real-world traffic speed datasets, where under a fair comparison, TransGTR outperforms state-of-the-art baselines by up to 5.4%.

CCS CONCEPTS

• **Computing methodologies** → **Transfer learning**; • **Information systems** → **Spatial-temporal systems**.

KEYWORDS

Traffic Forecasting, Transfer Learning, Graph Structure Learning

ACM Reference Format:
Yilun Jin, Kai Chen, and Qiang Yang. 2023. Transferable Graph Structure Learning for Graph-based Traffic Forecasting Across Cities. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599529

KDD '23, August 6–10, 2023, Long Beach, CA, USA

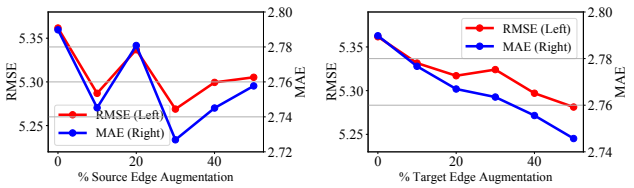
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599529

1 INTRODUCTION

Traffic forecasting is a fundamental problem for a variety of smart city applications. Accurately forecasting future traffic conditions serves as the foundation of numerous smart city services, such as trip planning [21, 23, 34], resource management [5, 45, 48], accident prediction [13, 46], etc. Traffic data can be generally modeled as spatio-temporal graphs, where sensors correspond to nodes and the dependencies between nodes correspond to edges. Therefore, with the success of learning on graphs [11, 20, 36], many deep learning models are proposed to solve the problem of traffic forecasting and have achieved state-of-the-art performances, such as spatio-temporal graph neural networks (STGNN) [1, 22, 40, 44]. However, the success of these models relies on large-scale traffic data, which may be inaccessible in real-world applications. For example, it takes a long time for newly-deployed sensors to collect large-scale data, during which the quality of smart city services may be unsatisfactory. Therefore, enhancing the performance of graph-based traffic forecasting under insufficient data is of pressing importance.

To address the data scarcity problem, researchers propose transfer learning [29] methods for traffic forecasting, aiming to transfer knowledge from a city with abundant data (i.e. the *source* city) to one with insufficient data (i.e. the *target* city). Among them, RegionTrans, MetaST, and CrossTReS [19, 37, 42] focus on grid-based data, where a city is divided into grids with fixed sizes and spatial relations. However, grid-based data fail to describe spatio-temporal graphs with irregular and flexible node-wise connections, and thus, these methods are not compatible with graph-based traffic forecasting. To complement the drawback, ST-GFSL [27] and DASTNet [33] are proposed to transfer knowledge for graph-based traffic forecasting. They propose meta-learning and domain adaptation methods to bridge the common knowledge between cities and use it to enhance the forecasting performance in the target city. However, a common drawback of ST-GFSL and DASTNet is that both methods directly adopt pre-defined graph structures for knowledge extraction and transfer. In practice, the pre-defined graph structures are handcrafted with rules, and may thus be noisy, missing, or biased, which may negatively impact knowledge transfer between cities.

We perform experiments on real-world traffic datasets, METR-LA (source) and PEMSD7M (target) [17] to illustrate the drawback of using pre-defined graphs for knowledge transfer. We augment the graph structures of both cities via the triadic closure rule [26, 49].



(a) Augmenting the Source Graph (b) Augmenting the Target Graph

Figure 1: Forecasting RMSE and MAE on PEMSD7M (target) with different ratios of edge augmentation on METR-LA (source) and PEMSD7M (target) graph structures. 0 denotes the original graph structures. For source augmentation, the target graph structure is kept unchanged, and vice versa. DASTNet [33] is applied for knowledge transfer.

We connect top-k% node pairs with the most common neighbors in both graphs, and apply DASTNet [33] to transfer knowledge between the augmented graphs. We vary the ratio k on both source and target graphs and plot the performances (average RMSE/MAE over 12 horizons) on PEMSD7M in Fig. 1. As shown, augmenting either the source (Fig. 1(a)) or the target graph (Fig. 1(b)) leads to lower errors (up to 2.4%) compared to directly using pre-defined graphs. The results suggest that the pre-defined graph structures, either the source or the target, may not be optimal for knowledge transfer for graph-based traffic forecasting, and that it is crucial to study *transferable graph structure learning*, i.e. training a *structure generator* to learn graph structures, and transferring both the generator and the *forecasting model* across cities. By doing so, we observe two opportunities. On one hand, the structure generator transferred from the source city is enriched with source knowledge, and thus, it can better identify helpful node-wise dependencies and learn a more effective target graph. On the other hand, by jointly learning graph structures for both cities, we can narrow the discrepancy between source and target data distributions and facilitate knowledge transfer for the forecasting model.

The problem of learning graph structures jointly with traffic forecasting models has been widely studied. Specifically, as long-term temporal patterns reveal node-wise similarity, researchers extract *node features* from long-term traffic data (e.g. weeks or months) and feed them to the structure generator to learn the graph [4, 30, 31]. However, existing works jointly learn the structure generator and forecasting model in one city with rich traffic data, while we aim to transfer both the generator and the forecasting model across cities. Moreover, no long-term traffic data is available in the target city. Therefore, two challenges arise.

- **Learning city-agnostic node features for transferable structure generators.** As inputs of the structure generator, city-agnostic node features must be learned before the source structure generator can be transferred to the target city. However, while long-term data is available in the source city, only short-term data exists in the target city. Therefore, long-term features such as periodicity and trends cannot be adequately learned in the target city, leading to a discrepancy between node features in both cities.
- **Learning graph structures for transferable forecasting models across cities.** Existing works learn graph structures to optimize forecasting performance in a single city. However, as the data distributions across cities are different [19], the graph structure thus learned may lead to a forecasting model with city-specific knowledge that is not transferable to another city. Furthermore, as graphs are discrete and irregular data, it is hard to evaluate their distributions or to minimize the distances between them.

In this paper, we propose TransGTR (**Trans**ferable **G**raphs for **Tr**affics), a transferable graph structure learning framework for graph-based traffic forecasting across cities. TransGTR consists of three main components, a node feature network $f_{\theta_{nf}}$, a structure generator f_ϕ, and a forecasting model f_θ. To learn a transferable structure generator, we train the node feature network $f_{\theta_{nf}}$ by distilling long-term knowledge from source data to short-term target data. Thus, target node features are enriched with source knowledge and generalizable with source node features, and the structure generator learned upon them can be transferred between cities. To further learn a transferable forecasting model, we jointly train the structure generator f_ϕ and the forecasting model f_θ with a temporal decoupled regularization. Specifically, by viewing graph structures as spatial feature extractors [38], we regularize f_ϕ, f_θ such that the spatial features extracted with them follow similar distributions in both cities and are thus transferable. We further separate temporal dynamics from spatial features to reduce the variance of the regularization. We conduct extensive experiments on real-world traffic datasets, where under a fair comparison, TransGTR achieves an improvement of up to 5.4% compared to state-of-the-art baselines.

To summarize, we make the following contributions.

- To the best of our knowledge, this is the first attempt to study the problem of transferable graph structure learning for graph-based traffic forecasting.
- We propose TransGTR, a transferable graph structure learning framework for traffic forecasting. With the city-agnostic node features and the temporal decoupled regularization, TransGTR jointly learns transferable structure generators and forecasting models across cities to improve the forecasting performance in the target city with limited data.
- We perform experiments on real-world graph-based traffic speed datasets where TransGTR outperforms state-of-the-art baselines by as much as 5.4%.

2 RELATED WORK

2.1 Graph-based Traffic Forecasting

Spatio-temporal graphs are natural representations of traffic data, where nodes represent sensors with their time series, and edges between nodes depict dependencies between sensors (e.g. spatial closeness). To solve the graph-based traffic forecasting problem, many deep forecasting models have been proposed, such as spatio-temporal graph neural network (STGNN) models. STGNNs commonly apply sequential models (e.g. causal convolutions or recurrent neural networks (RNN)) to capture temporal features from individual time series, and apply graph neural networks (e.g. graph convolutional network (GCN) [20], graph attention network (GAT) [36],

etc.) to aggregate features from related time series [9, 18, 22, 40, 44]. However, despite being powerful with rich traffic data, STGNN models fail to make accurate forecasting without sufficient training data, which is the problem we aim to address in this paper.

2.2 Structure Learning for Traffic Forecasting

A graph structure that faithfully depicts dependencies between nodes is crucial to accurate traffic forecasting. However, pre-defined graph structures are often handcrafted with heuristics (e.g. spatial distance) and thus may contain bias and noise and compromise the performance of forecasting. To address the issue, structure learning methods are proposed for traffic forecasting aiming to learn more reliable dependencies between nodes. Existing works on structure learning [4, 30, 39] generally extract features from very long-term traffic data to generate the graph and optimize it jointly with the forecasting model. Empirically, structure learning has been shown effective for a wide range of forecasting tasks [51]. Different from existing works that learn the graph structure in a single city, we aim to learn graph structures that encode transferable knowledge across cities, with the target city having only short-term data.

While the above works focus on learning a *time-invariant* graph structure for all time steps, there are also works that learn *time-dependent* graph structures for different time steps [12, 31, 43]. However, these methods commonly require much more parameters to learn evolving graph structures, which are prone to overfitting in a target city with limited data. Therefore, we focus on learning transferable *time-invariant* graph structures in this paper.

2.3 Transfer Learning for Traffic Forecasting

Data insufficiency is a common problem in real-world traffic forecasting tasks. For example, for newly deployed traffic speed sensors, it takes a long time to accumulate large-scale traffic data, during which many smart city services may be compromised. To address the problem, transfer learning methods have been proposed for traffic forecasting. Among them, RegionTrans, MetaST, and CrossTReS [19, 37, 42] aim to transfer knowledge for grid-based data, and are thus not optimal for graph-based traffic data. On the contrary, ST-GFSL [27] and DASTNet [33] are designed for graph-based traffic data. They apply hypernetworks [10], meta-learning [7, 27], and domain adaptation [8] to extract common knowledge to improve forecasting in the target city. However, both ST-GFSL and DASTNet apply the pre-defined graph structures to extract and transfer knowledge, which, as shown in Fig. 1, is generally not optimal.

3 PRELIMINARIES & PROBLEM DEFINITION

In this section, we first introduce the necessary backgrounds and notations, and then formally define our problem of transferable structure learning for traffic forecasting.

3.1 Background and Notations

Definition 1 (Graph-based Traffic Data). *We define a graph-based traffic dataset as $\mathcal{D} = (\mathcal{V}, \mathbf{X}, \mathbf{A})$, where $\mathcal{V}$ is the set of nodes (e.g. sensors), $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times T}$ is the matrix for time series data (e.g. traffic speed, car flow, etc.), T is the total number of time steps, and $\mathbf{A} \in [0,1]^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix depicting the dependencies between nodes. We denote the time series data of node v as $\mathbf{x}_v \in \mathbb{R}^T$.*

Definition 2 (Transfer Learning for Traffic Forecasting). *Given a source city with graph-based traffic data $\mathcal{D}_\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathbf{X}_\mathcal{S}, \mathbf{A}_\mathcal{S})$ and a target city with data $\mathcal{D}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathbf{X}_\mathcal{T}, \mathbf{A}_\mathcal{T})$, $T_\mathcal{T} \ll T_\mathcal{S}$, we aim to learn a forecasting model f_θ with target data and rich source data*

$$\left[\hat{\mathbf{X}}_\mathcal{T}^{t+1}, \ldots, \hat{\mathbf{X}}_\mathcal{T}^{t+k_o}\right] = f_\theta\left(\left[\mathbf{X}_\mathcal{T}^{t-k_i+1}, \ldots, \mathbf{X}_\mathcal{T}^{t}\right], \mathbf{A}_\mathcal{T}; \mathbf{X}_\mathcal{S}, \mathbf{A}_\mathcal{S}\right), \quad (1)$$

to minimize the following error

$$\mathcal{L}_\mathcal{T} = \sum_{v\in\mathcal{V}_\mathcal{T}} \sum_{t=k_i}^{T_\mathcal{T}-k_o} \sum_{j=1}^{k_o} \mathcal{L}_{err}\left(\hat{\mathbf{x}}_v^{t+j}, \mathbf{x}_v^{t+j}\right), \quad (2)$$

where k_o, k_i are the forecasting and the input horizons, respectively, and $\mathcal{L}_{err}$ is an error function such as squared error or absolute error.

3.2 Motivation and Problem Definition

Existing works such as DASTNet [33] and ST-GFSL [27] focus on the problem of Def. 2. They extract knowledge from source data $\mathbf{X}_\mathcal{S}, \mathbf{A}_\mathcal{S}$, such as well-trained forecasting model parameters, and transfer them to the target data $\mathbf{X}_\mathcal{T}, \mathbf{A}_\mathcal{T}$ to improve forecasting performance. However, both methods use the pre-defined graph structures $\mathbf{A}_\mathcal{S}, \mathbf{A}_\mathcal{T}$ to extract and transfer knowledge. In practice, the pre-defined graphs may be noisy, missing, or contain city-specific heuristics. As a result, the knowledge, such as the forecasting model learned from the pre-defined graph structures, may also inherit the noise or city-specific heuristics, which negatively impacts knowledge transfer across cities, as shown in Fig. 1.

Based on the above observation, we define the problem of transferable structure learning for traffic forecasting as follows.

Definition 3 (Transferable Structure Learning for Traffic Forecasting). *Given a source city and a target city with graph-based traffic data $\mathcal{D}_\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathbf{X}_\mathcal{S}, \mathbf{A}_\mathcal{S})$ and $\mathcal{D}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathbf{X}_\mathcal{T}, \mathbf{X}_\mathcal{T})$, $T_\mathcal{T} \ll T_\mathcal{S}$, we aim to jointly learn a structure generator f_ϕ and a forecasting model f_θ,*

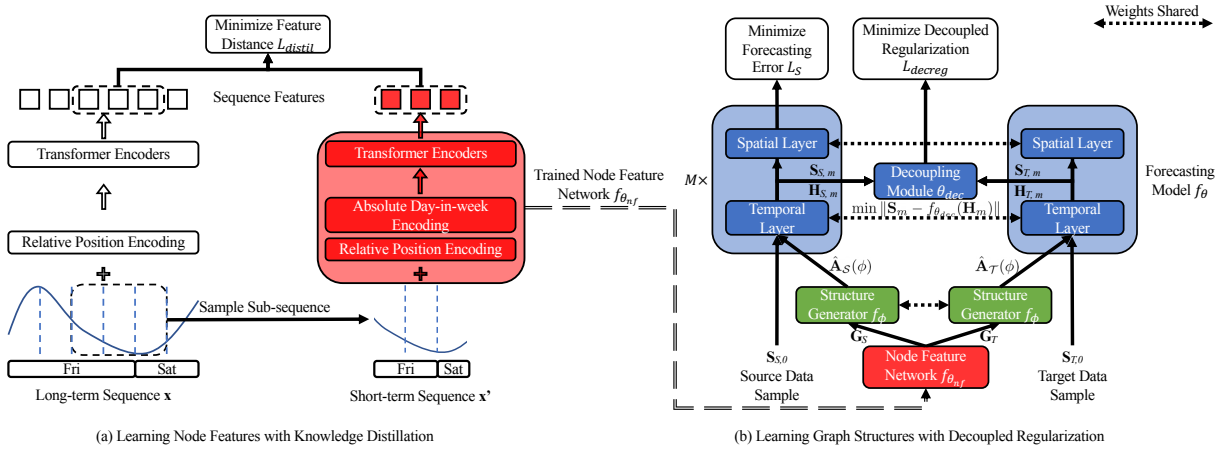
$$\begin{aligned} &\hat{\mathbf{A}}_\mathcal{S}(\phi), \hat{\mathbf{A}}_\mathcal{T}(\phi) = f_\phi(\mathbf{X}_\mathcal{S}, \mathbf{X}_\mathcal{T}) \\ &\left[\hat{\mathbf{X}}_\mathcal{T}^{t+1}, \ldots, \hat{\mathbf{X}}_\mathcal{T}^{t+k_o}\right] = f_\theta\left(\left[\mathbf{X}_\mathcal{T}^{t-k_i+1}, \ldots, \mathbf{X}_\mathcal{T}^{t}\right], \hat{\mathbf{A}}_\mathcal{T}(\phi), \mathbf{X}_\mathcal{S}, \hat{\mathbf{A}}_\mathcal{S}(\phi)\right), \end{aligned} \quad (3)$$

such that error on the target city $\mathcal{L}_\mathcal{T}$ (Eqn. 2) is minimized.

Different from Def. 2, the problem of Def. 3 learns and transfers both the forecasting model f_θ and the structure generator f_ϕ, which leads to two potential advantages. On one hand, the structure generator f_ϕ is trained with both target data and rich source data, thus providing additional knowledge to identify helpful edges and learn a better target graph. On the other hand, by learning both graphs $\hat{\mathbf{A}}_\mathcal{S}(\phi)$ and $\hat{\mathbf{A}}_\mathcal{T}(\phi)$, we can minimize the discrepancy between source and target data distributions $(\mathbf{X}_\mathcal{S}, \hat{\mathbf{A}}_\mathcal{S}(\phi))$ and $(\mathbf{X}_\mathcal{T}, \hat{\mathbf{A}}_\mathcal{T}(\phi))$ and thus learn the forecasting model with more transferable knowledge and better target forecasting performance.

4 PROPOSED METHOD

In this section, we introduce the proposed framework, TransGTR. We first present an overview of TransGTR, before introducing detailed components and the training process.



(a) Learning Node Features with Knowledge Distillation

(b) Learning Graph Structures with Decoupled Regularization

Figure 2: Overview of TransGTR with a node feature network $f_{\theta_{nf}}$ (red), a structure generator f_ϕ (green), and a forecasting model f_θ (blue). The node feature network learns city-agnostic node features with knowledge distillation, such that knowledge encoded in the structure generator can be transferred across cities. To further learn forecasting models with transferable knowledge, the structure generator is trained by minimizing the distance between spatial features from both cities.

4.1 Overview of TransGTR

The key challenges to transferable graph structure learning are two-fold. First, we should learn city-agnostic node features from long-term source data and short-term target data to learn a transferable structure generator. Second, we should learn graph structures for both cities to minimize the discrepancy between their data distributions, thus learning a transferable forecasting model. To address both challenges, TransGTR consists of three main components.

- **Node Feature Network** $f_{\theta_{nf}}$. As inputs to the structure generator, city-agnostic node features must be learned such that the structure generator f_ϕ can be transferred across cities. However, a discrepancy may exist between node features learned from long-term source data and short-term target data. To mitigate the issue, $f_{\theta_{nf}}$ is trained with long-term temporal knowledge distilled from the source city.
- **Structure generator** f_ϕ. Given city-agnostic node features from $f_{\theta_{nf}}$, f_ϕ generates graph structures $\hat{\mathbf{A}}_\mathcal{S}(\phi), \hat{\mathbf{A}}_\mathcal{T}(\phi)$ for both cities. To facilitate learning transferable forecasting models, we view graph structures as spatial feature extractors and learn f_ϕ with a temporal decoupled regularization, such that spatial features extracted with $\hat{\mathbf{A}}_\mathcal{S}(\phi), \hat{\mathbf{A}}_\mathcal{T}(\phi)$ share similar distributions and are thus transferable. In addition, we separate temporal dynamics from the spatial features such that the variance of the spatial features is reduced, and the regularization is less noisy.
- **Forecasting Model** f_θ. It takes the learned graph structures $\hat{\mathbf{A}}_\mathcal{S}(\phi), \hat{\mathbf{A}}_\mathcal{T}(\phi)$ and outputs forecasting results.

We illustrate the framework of the proposed TransGTR in Fig. 2.

4.2 TransGTR Components

4.2.1 Node Feature Network $f_{\theta_{nf}}$. The purpose of $f_{\theta_{nf}}$ is to capture city-agnostic node features across source and target cities upon which f_ϕ can be learned with knowledge from both cities. As long-term temporal features have been widely used to learn graph structures for traffic forecasting [4, 30], we use the TSFormer model [31], designed to capture long-term features from time series, as $f_{\theta_{nf}}$. Given an input sequence $\mathbf{x} \in \mathbb{R}^{L \cdot P}$, TSFormer first splits $\mathbf{x}$ into patches of length P and projects them into patch embeddings $\mathbf{x}_{emb} \in \mathbb{R}^{L \times n_{emb}}$. Then, $\mathbf{x}_{emb}$ is added with a relative and trainable positional encoding $\mathbf{pe}_{rel} \in \mathbb{R}^{L \times n_{emb}}$ and fed into a series of Transformer encoder blocks [35]. We denote the outputs of the encoder as $\mathbf{x}_{enc} \in \mathbb{R}^{L \times n_{emb}} = f_{\theta_{nf}}(\mathbf{x})$. We refer readers to [31] for details regarding TSFormer. We also note that TransGTR is compatible with other Transformer-based models for time series.

Enhancing weekly periodicity with day-in-week encodings. Weekly periodicity is a common long-term property in traffic data. For example, slow traffic speeds can often be observed during rush hours on weekdays, but not on weekends. However, while the source city contains rich data from which periodicity can be learned, only short-term (e.g. several days) data is accessible in the target city. Therefore, it is difficult to learn node features for the target city that reflect temporal periodicity, and thus, the node features in the target city may not be generalizable with those in the source city.

To mitigate the issue, we propose to add another positional encoding module, an absolute day-in-week encoding $\mathbf{e}_{diw} \in \mathbb{R}^{7 \times n_{emb}}$ to the node feature network $f_{\theta_{nf}}$. Specifically, for all patches in the input sequence $\mathbf{x}$, we obtain their day-in-week information and aggregate them into $\mathbf{t}_{diw} \in [1, \dots, 7]^{L}$[1]. The day-in-week encodings for the input $\mathbf{x}$ are then obtained via an embedding lookup

$$\mathbf{pe}_{diw}(\mathbf{x}) = \mathbf{e}_{diw}[\mathbf{t}_{diw}], \tag{4}$$

which are then added to the patch embeddings $\mathbf{x}_{emb}$ and the relative positional encodings $\mathbf{pe}_{rel}$, and fed into the Transformer encoders.

[1] If a patch spans across two days, its day-in-week is determined via majority voting.

4.2.2 Structure Generator f_ϕ. The structure generator f_ϕ takes the node features learned by $f_{\theta_{nf}}$, i.e.

$$\mathbf{G}_\mathcal{S} = f_{\theta_{nf}}(\mathbf{X}_\mathcal{S}), \mathbf{G}_\mathcal{T} = f_{\theta_{nf}}(\mathbf{X}_\mathcal{T}), \tag{5}$$

and transforms them into graph structures $\hat{\mathbf{A}}_\mathcal{S}(\phi), \hat{\mathbf{A}}_\mathcal{T}(\phi)$ for both cities. By doing so, f_ϕ is learned with both target data and abundant source data, and can thus better identify helpful edges from noisy ones. TransGTR does not assume specific architectures for f_ϕ. We refer readers to Appendix Section C for our implementation.

4.2.3 Forecasting Model f_θ. Given input data $[\mathbf{X}^{t-k_i+1}, \dots \mathbf{X}^t]$ and the graph structure $\hat{\mathbf{A}}(\phi)$ given by f_ϕ, the forecasting model f_θ transforms them into predictions $[\hat{\mathbf{X}}^{t+1}, \dots \hat{\mathbf{X}}^{t+k_o}]$. Similarly, TransGTR does not require specific forecasting models. We generally assume that f_θ consists of M stacked spatial and temporal layers, i.e.

$$\begin{aligned} \mathbf{H}_m &= \text{TemporalLayer}_m(\mathbf{S}_{m-1}), \\ \mathbf{S}_m &= \text{GNNLayer}_m(\mathbf{H}_m, \hat{\mathbf{A}}(\phi)), m = 1, \dots M, \end{aligned} \tag{6}$$

where $\mathbf{H}_m, \mathbf{S}_m$ stand for the temporal and spatial features at layer m, respectively, and $\mathbf{S}_0 = [\mathbf{X}^{t-k_i+1}, \dots, \mathbf{X}^t]$ is the input time series. Finally, a regressor transforms $\mathbf{S}_M$ to $[\hat{\mathbf{X}}^{t+1}, \dots, \hat{\mathbf{X}}^{t+k_o}]$ as the prediction. The formulation in Eqn. 6 subsumes a wide variety of STGNN models, such as STGCN, DCRNN, GraphWaveNet [22, 40, 44], etc.

4.3 Transferable Structure Learning with TransGTR

In this section, we introduce how TransGTR addresses the two challenges in Section 4.1, i.e. learning city-agnostic node features, and learning graph structures for transferable forecasting models.

4.3.1 Learning City-agnostic Node Features via Knowledge Distillation. City-agnostic node features are crucial to transfer the knowledge encoded in the structure generator across cities. However, it is challenging to learn city-agnostic node features from *long-term* source data and *short-term* target data. To address the challenge, we adapt knowledge distillation [15], widely used to transfer knowledge from a large teacher network to a small student network, to transfer long-term knowledge to short-term target data. The intuition is that, features learned from long-term data contain additional knowledge than those learned from short-term data, and that we can learn such additional knowledge by fitting those features.

Specifically, we first follow STEP [31] to pre-train a node feature network $f_{\theta_{nf,\mathcal{S}}}$ with rich source data to capture long-term temporal knowledge. The source feature network is pre-trained with masked autoencoding [14] to reconstruct masked patches using unmasked ones in a long input sequence $\mathbf{x} \in \mathbb{R}^{L \cdot P}$. We then aim to distill the rich knowledge encoded in $f_{\theta_{nf,\mathcal{S}}}$ to $f_{\theta_{nf}}$ which only takes short-term sequences as inputs. Given a long-term sequence $\mathbf{x} \in \mathbb{R}^{L \cdot P}$, we obtain its corresponding short-term sequence $\mathbf{x}' \in \mathbb{R}^{L_{short} \cdot P}$ as

$$\mathbf{x}' = \mathbf{x}[p \cdot P : (p + L_{short}) \cdot P], \tag{7}$$

where $\mathbf{x}[a : b]$ denotes the operation of selecting elements with indices between a and b, and p is a random position. We then feed $\mathbf{x}$ and $\mathbf{x}'$ to $f_{\theta_{nf,\mathcal{S}}}$ and $f_{\theta_{nf}}$ respectively to obtain the corresponding long-term and short-term features $\mathbf{x}_{enc}, \mathbf{x}'_{enc}$,

$$\begin{aligned} \mathbf{x}_{enc} &= f_{\theta_{nf,\mathcal{S}}}(\mathbf{x}) \in \mathbb{R}^{L \times n_{emb}}, \\ \mathbf{x}'_{enc} &= f_{\theta_{nf}}(\mathbf{x}') \in \mathbb{R}^{L_{short} \times n_{emb}}. \end{aligned} \tag{8}$$

As $\mathbf{x}_{enc}$ is enriched with long-term observations (i.e. $\mathbf{x} \backslash \mathbf{x}'$) while $\mathbf{x}'_{enc}$ is not, $\mathbf{x}_{enc}$ should contain richer long-term temporal knowledge than $\mathbf{x}'_{enc}$. Therefore, we minimize the distance between $\mathbf{x}'_{enc}$ and the corresponding patch features in $\mathbf{x}_{enc}$, such that $f_{\theta_{nf}}$ learns to seek long-term features from short-term data,

$$\mathcal{L}_{distil}(\mathbf{x}) = \left\| \mathbf{x}'_{enc} - \mathbf{x}_{enc}[p : p + L_{short}] \right\|^2, \tag{9}$$

where $\| \cdot \|$ denotes the L_2 distance. Finally, to learn both city-agnostic and helpful node features, we train $f_{\theta_{nf}}$ by minimizing the following objective,

$$\min_{\theta_{nf}} \mathcal{L}_{MaskedAE}(\mathbf{x}_{v_t}) + \lambda_d \mathcal{L}_{distil}(\mathbf{x}_{v_s}), \tag{10}$$

where $\mathbf{x}_{v_s}, \mathbf{x}_{v_t}$ are data samples from source and target cities, $\mathcal{L}_{MaskedAE}$ denotes the loss of masked autoencoding [14], and λ_d is a hyperparameter. In this way, $f_{\theta_{nf}}$ learns to generate node features for both source (via $\mathcal{L}_{distill}(\mathbf{x}_{v_s})$) and target cities (via $\mathcal{L}_{MaskedAE}(\mathbf{x}_{v_t})$). Thus, f_ϕ, which takes the city-agnostic node features as inputs, can be transferred to generate graph structures across cities.

4.3.2 Learning Graph Structures via Temporal Decoupled Regularization. In this section, we introduce how to learn graph structures to facilitate learning transferable forecasting models across cities. Specifically, we first make two observations from existing works.

- **Graph structure itself is a feature extractor**. Each GNN layer consists of two operations, feature passing where node features are passed along the edges, and feature transformation where nodes aggregate and transform the received features. As studied in [28, 40, 50], the feature passing step, involving only the graph structure, can be seen as a low-pass filter on its graph signals (i.e. node features) that extract smooth node features. Therefore, in addition to a part of the input data, the graph structure works as an implicit spatial feature extractor for graph-based traffic data.
- **Feature extractors should extract similar features across domains**. Many existing works on domain adaptation [8, 24, 25] have shown that for a feature extractor to be transferable across domains, it should extract features from different domains with similar distributions.

Inspired by these observations, we propose a spatial feature regularization term to minimize the distance between spatial features extracted from both source and target cities, i.e.

$$\mathcal{L}_{reg} = \sum_{m=1}^{M} d\left(\mathbf{S}_{\mathcal{S},m}, \mathbf{S}_{\mathcal{T},m}\right), \tag{11}$$

where $\mathbf{S}_{\mathcal{S},m}$ denotes the spatial features extracted from source input data $[\mathbf{X}_\mathcal{S}^{t-k_i+1}, \dots, \mathbf{X}_\mathcal{S}^t]$ at layer m, and similarly for $\mathbf{S}_{\mathcal{T},m}$. We adopt CORAL [32] as the distance metric $d(\cdot, \cdot)$.

Robust Regularization via Temporal Decoupling. A major assumption in domain adaptation is that data samples are independently and identically distributed [2, 3, 47]. However, the assumption does

not hold for the task of traffic forecasting. Specifically, data samples from different time steps do not necessarily follow the same distribution. For example, traffic during rush hours on weekdays commonly suffers from congestion, and thus, speed data sampled from rush hours differ significantly from those sampled from weekends. As a result, the spatial features $\mathbf{S}_m$ may have wildly different distributions even for the same graph structures, which leads to noise and instability in the evaluation and optimization of $\mathcal{L}_{reg}$.

We propose a temporal decoupled regularization to address the issue. The intuitions of the temporal decoupling are two-fold.

- The differences in the distributions of training samples, and consequently $\mathbf{S}_m$ are caused by the dynamic nature of traffic data, and thus, we can reduce the distribution differences by decoupling the temporal dynamics from $\mathbf{S}_m$.
- The temporal dynamics of $\mathbf{S}_m$ comes solely from its preceding temporal features $\mathbf{H}_m$.

We design a series of decoupling modules $\theta_{dec,m}$ to implement the intuition, where m denotes the layer index. Inspired by regression analysis, we aim to train the decoupling modules to reconstruct $\mathbf{S}_m$ with its preceding temporal features $\mathbf{H}_m$, i.e.

$$\min_{\{\theta_{dec,m}\}_{m=1,\dots M}} \mathcal{L}_{recons} = \sum_{m=1}^{M} \left\| \mathbf{S}_m - f_{\theta_{dec,m}}(\mathbf{H}_m) \right\|^2. \tag{12}$$

In this way, the spatial features $\mathbf{S}_m$ are implicitly decomposed into two parts, $f_{\theta_{dec,m}}(\mathbf{H}_m)$ which is explained by the temporal dynamics in $\mathbf{H}_m$, and the residual spatial features $\tilde{\mathbf{S}}_m = \mathbf{S}_m - f_{\theta_{dec,m}}(\mathbf{H}_m)$ which are not dynamic. We then replace $\mathbf{S}_{\mathcal{S},m}, \mathbf{S}_{\mathcal{T},m}$ in $\mathcal{L}_{reg}$ with the residual spatial features $\tilde{\mathbf{S}}_{\mathcal{S},m}, \tilde{\mathbf{S}}_{\mathcal{T},m}$ to obtain the decoupled regularization $\mathcal{L}_{decreg}$ as a more stable regularization than $\mathcal{L}_{reg}$.

Finally, we jointly train the structure generator f_ϕ and the forecasting model f_θ with the following objective to learn graph structures that help extract transferable knowledge across cities

$$\min_{\phi,\theta} \mathcal{L}_{\mathcal{S}} + \lambda_r \mathcal{L}_{decreg}, \tag{13}$$

where $\mathcal{L}_{\mathcal{S}}$ is the forecasting error evaluated on abundant source data, similar to Eqn. 2, and λ_r is a hyperparameter. In addition, we alternately train the decoupling modules $\{\theta_{dec,m}\}$ with Eqn. 12. We note that $\{\theta_{dec,m}\}$ and θ, ϕ are not jointly optimized to avoid learning naive graph structures, i.e. $\hat{\mathbf{A}} = \mathbf{I}$.

4.3.3 Overall Training Algorithm. The overall training process of TransGTR is shown in Algorithm 1 in the Appendix. We first train the node feature network $f_{\theta_{nf}}$ for city-agnostic node features. Then, we fix the node feature network $f_{\theta_{nf}}$ to reduce computation and memory overhead, and train θ, ϕ to learn transferable forecasting models. Finally, f_θ, f_ϕ are further fine-tuned on target data.

5 EXPERIMENTS

In this section, we present our experimental evaluations on TransGTR. Our evaluations aim to answer the following questions:

- How does TransGTR compare against state-of-the-art transfer learning methods for graph-based traffic forecasting?
- How do individual TransGTR design components and hyperparameters affect the overall performance of TransGTR?
- How can the advantages of TransGTR be intuitively shown and understood?

5.1 Experimental Setup

Datasets. We use four real-world traffic speed datasets, METR-LA, PEMS-BAY, PEMSD7M, and HKTD for evaluation. We show statistics of the datasets in Table 4 in the Appendix. We take METR-LA and PEMS-BAY, datasets with more samples as source cities, and the rest as target cities. For target cities, to simulate the lack of data, we use 7-day and 3-day data for training. We cover more details about data processing in Appendix Section A.

Base Models. We take GTS [30] as the base structure generator f_ϕ, and Graph-WaveNet (GWN) [40] as the base forecasting model f_θ due to their empirically good performance and efficiency [17]. We also note that TransGTR applies to other structure generators and graph-based forecasting models (with the formulation in Eqn. 6). Details of base models are discussed in Appendix Section C.

Metrics. Following existing works [22, 27, 31, 40], we evaluate the forecasting performance by root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). We evaluate the performance with two forecasting horizons, 30 minutes (6 steps ahead) and 60 minutes (12 steps ahead).

Comparison Methods. We compare TransGTR with the following transfer learning methods for traffic forecasting.

- **FT-GWN** and **FT-GTS**. We first train GWN and GTS models with the source data, and fine-tune them with target data.
- **RegionTrans** [37]. With the GWN trained with source data, RegionTrans computes the similarity between source and target nodes and uses it to regularize target fine-tuning.
- **DASTNet** [33]. In addition to the GWN trained with source data, DASTNet also learns domain adaptive node embeddings to bridge structurally similar nodes between cities.
- **ST-GFSL** [27]. ST-GFSL leverages hypernetworks [10] to generate node-specific parameters. The node-specific parameters are generated via node-level meta-knowledge, extracted from the graph-based traffic data, such that similar nodes share similar parameters.

We also compare with baselines that only use target data.

- **ARIMA**, i.e. auto-regressive intergrated moving average. It is a statistical regression model for time series.
- **GWN** and **GTS** models trained target data only.

For a fair comparison, except for ARIMA, all baselines adopt GWN as the base forecasting model. We present details of baselines in Appendix Section B. We also discuss implementation details and hyperparameter settings of TransGTR in Appendix Section C. We provide the link to the code and data at

`https://github.com/KL4805/TransGTR/`

5.2 Performance Comparison

In this section, we present quantitative evaluations of TransGTR. We evaluate TransGTR and baselines on two datasets, PEMSD7M and HKTD. For each task, we transfer from both METR-LA and PEMS-BAY. We report the means and standard deviations of 5 independent runs in Table 1. Due to space limitations, we separately

Table 1: Comparative evaluation results with PEMSD7M and HKTD as target cities. LA and BAY stand for METR-LA and PEMS-BAY as source cities, respectively. In each column, the best result is presented in bold and the second best is underlined.

Target City	Baselines	Target Data	7-day								3-day							
		Horizon	30 mins				60 mins				30 mins				60 mins			
		Metrics	RMSE		MAE		RMSE		MAE		RMSE		MAE		RMSE		MAE	
PEMSD7M	Target Only	ARIMA	6.525		3.682		8.942		5.426		6.526		3.698		8.946		5.453	
		GWN	5.748		2.999		7.279		3.824		6.053		3.126		7.994		4.162	
		GTS	5.639		2.988		7.071		3.746		5.831		3.111		7.508		4.014	
	Transfer	**Source**	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY
		FT-GWN	5.645	5.771	2.913	2.970	7.038	7.128	3.636	3.690	5.873	5.935	3.045	3.077	7.349	7.596	3.845	3.978
		FT-GTS	5.685	5.651	2.908	2.901	6.952	6.899	3.526	3.543	5.946	5.986	3.024	3.078	7.203	7.205	3.736	3.749
		RegionTrans	5.654	5.702	2.909	2.935	6.986	7.077	3.597	3.659	5.868	5.948	3.046	3.073	7.376	7.545	3.862	3.963
		DASTNet	5.659	5.633	2.901	2.905	6.976	6.954	3.553	3.599	5.839	5.908	3.031	3.078	7.245	7.294	3.774	3.811
		ST-GFSL	5.647	5.642	2.941	2.927	6.937	6.931	3.535	3.541	5.840	5.912	3.012	3.071	7.219	7.218	3.738	3.744
		TransGTR	**5.461**	**5.454**	**2.800**	**2.802**	**6.565**	**6.601**	**3.340**	**3.373**	**5.627**	**5.679**	**2.960**	**2.958**	**6.922**	**6.931**	**3.604**	**3.599**
		Std. Dev.	0.024	0.015	0.019	0.007	0.041	0.022	0.028	0.008	0.029	0.040	0.017	0.016	0.040	0.053	0.026	0.020
HKTD	Target Only	ARIMA	6.648		3.816		8.249		4.843		6.650		3.822		8.253		5.863	
		GWN	6.062		3.386		7.206		4.000		6.333		3.477		7.727		4.333	
		GTS	6.052		3.380		6.954		3.903		6.252		3.453		7.249		4.011	
	Transfer	**Source**	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY
		FT-GWN	5.755	5.792	3.237	3.264	6.552	6.551	3.682	3.728	5.939	5.949	3.351	3.373	6.984	6.927	3.922	3.980
		FT-GTS	5.792	5.796	3.242	3.253	6.420	6.496	3.633	3.682	5.999	5.982	3.369	3.351	6.784	6.849	3.854	3.862
		RegionTrans	5.696	5.728	3.216	3.228	6.424	6.456	3.654	3.683	5.935	5.943	3.342	3.345	6.870	6.894	3.880	3.933
		DASTNet	5.690	5.704	3.200	3.221	6.411	6.442	3.619	3.655	**5.905**	5.921	3.379	3.361	6.786	6.798	3.881	3.862
		ST-GFSL	5.704	5.739	3.225	3.231	6.477	6.435	3.624	3.638	5.960	5.993	3.392	3.388	6.847	6.821	3.869	3.878
		TransGTR	**5.666**	**5.661**	**3.141**	**3.140**	**6.205**	**6.232**	**3.441**	**3.455**	5.928	**5.877**	**3.305**	**3.290**	**6.622**	**6.589**	**3.693**	**3.697**
		Std. Dev.	0.018	0.016	0.007	0.018	0.026	0.022	0.013	0.017	0.019	0.018	0.010	0.011	0.031	0.025	0.011	0.010

Table 2: Results of Model Analysis. The target city is chosen as PEMSD7M with 7-day data.

Analyzed Component	Source	METR-LA				PEMS-BAY			
	Horizon	30 mins		60 mins		30 mins		60 mins	
	Metric	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Node Feature Learning	TransGTR-NoDistil	5.573	2.869	6.742	3.439	5.545	2.846	6.761	3.474
	TransGTR-NoWP	5.519	2.837	6.671	3.382	5.559	2.833	6.715	3.423
Graph Structure Learning	TransGTR-NoSL	5.645	2.913	7.038	3.636	5.771	2.970	7.128	3.690
	TransGTR-NoReg	5.591	2.860	6.764	3.434	5.591	2.873	6.778	3.460
	TransGTR-NoDec	5.552	2.843	6.693	3.415	5.529	2.840	6.729	3.428
	TransGTR	**5.461**	**2.800**	**6.565**	**3.340**	**5.454**	**2.802**	**6.601**	**3.373**

show the results of MAPE in Appendix Section D. We make the following observations from Table 1.

- TransGTR consistently achieves improvements against state-of-the-art baselines. For PEMSD7M as the target city, TransGTR performs the best in all evaluated metrics. For HKTD as the target city, TransGTR performs the best in 15 out of 16 evaluated metrics. Compared with the second best results, TransGTR achieves an improvement of up to 5.4%.
- TransGTR is more competitive with longer forecasting horizons. For example, for PEMSD7M as the target city, TransGTR achieves an improvement of up to 3.6% with a 30-minute forecasting horizon, compared to 5.4% with a 60-minute forecasting horizon. The observation corresponds with the design of TransGTR that transfers long-term temporal knowledge to the target city, such that the learned graph structure in the target city better reflect node-wise dependencies.

5.3 Model Analysis

In this section, we analyze the effects of individual components in TransGTR. We analyze both node feature learning for transferable structure generators, and graph structure learning for transferable forecasting models. The other part is not changed when we analyze one part. We perform analyses with METR-LA and PEMS-BAY as source cities, PEMSD7M as the target city with 7-day training data.

5.3.1 City-agnostic Node Feature Learning. We analyze how learning city-agnostic node features affects the performance of TransGTR. We compare TransGTR with the following variants.

- **TransGTR-NoDistil**. We remove knowledge distillation and only train $f_{\theta_{nf}}$ with target data. In this way, TransGTR-NoDistil generates target node features $\mathbf{G}_{\mathcal{T}}$ that are not generalizable with $\mathbf{G}_{\mathcal{S}}$, and thus, the structure generator trained on source data may not transfer well to target data.

- **TransGTR-NoWP**, in which we remove the day-in-week encoding from $f_{\theta_{nf}}$. In this way, TransGTR-NoWP may not adequately model the property of weekly periodicity, which is crucial in long-term traffic data.

We use RMSE and MAE on PEMSD7M as metrics, similar to Section 5.2. We show the results in Table 2 with the following observations.

- Compared with TransGTR-NoDistil, TransGTR-NoWP and TransGTR consistently achieve better performances. The improvement indicates that with knowledge distillation, $f_{\theta_{nf}}$ learns city-agnostic node features in both cities, upon which the shared structure generator is enriched with both source and target knowledge to learn more helpful graphs.
- TransGTR consistently outperforms TransGTR-NoWP, which echoes the fact that weekly periodicity is an important property in long-term traffic data, but cannot be well modeled with only short-term target data. Thus, with the day-in-week encodings, $f_{\theta_{nf}}$ learns node features for the target city that better resemble those in the source city.

5.3.2 Graph Structure Learning with Decoupled Regularization. We compare TransGTR with the following variants to analyze the effects of the temporal decoupled regularization.

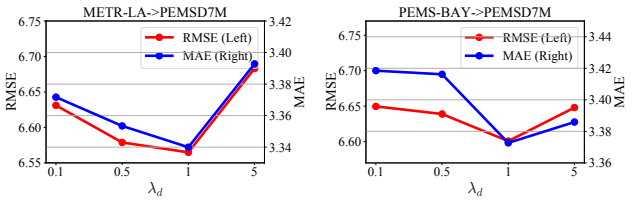
- **TransGTR-NoSL**, in which we disable structure learning and use the pre-defined graph structures. Without structure learning, TransGTR-NoSL is equivalent to FT-GWN.
- **TransGTR-NoReg**. We set $\lambda_r = 0$ in Eqn. 13. In this way, the structure generator learns a graph structure that only improves forecasting performance in the source city.
- **TransGTR-NoDec**. We remove the decoupling modules $\theta_{dec,m}$ and replace $\mathcal{L}_{decreg}$ with $\mathcal{L}_{reg}$ in Eqn. 13. In this way, the spatial feature regularization suffers from variances caused by temporal dynamics, causing instability that may harm the overall performance.

We also show results in Table 2 with the following observations.

- TransGTR-NoSL yields significantly worse performances compared with all other variants, which, in addition to Fig. 1, indicates that learning graph structures is crucial for transferring knowledge for traffic forecasting across cities.
- TransGTR-NoReg performs worse compared to variants with regularization. As TransGTR-NoReg learns a graph structure only for the source city, the observation suggests that such a graph structure may only capture city-specific knowledge and thus harm knowledge transfer.
- TransGTR-NoDec improves marginally over TransGTR-NoReg and is less competitive than TransGTR, which underscores the importance of temporal decoupling in the regularization term, without which the performance is compromised.

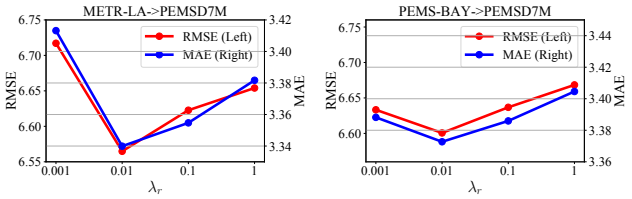
5.4 Parameter Analysis

In this section, we analyze the effects of hyperparameters λ_d (Eqn. 10) and λ_r (Eqn. 13) in TransGTR. We follow the setting as Section 5.3. We vary $\lambda_d \in \{0.1, 0.5, 1, 5\}$, $\lambda_r \in \{0.001, 0.01, 0.1, 1\}$, and plot the corresponding performances in Fig. 3 and 4. We observe that as we increase both λ_d and λ_r, the forecasting errors first decrease and then increase after specific values ($\lambda_d = 1, \lambda_r = 0.01$). As shown in Eqn. 10, λ_d aims to strike a balance between source knowledge and target knowledge, and thus, setting a large λ_d results in node features that are biased to the source city, upon which the structure generator is also biased. Similarly, as shown in Eqn. 13, an unduly large λ_r may result in learning naive graph structures (i.e. $\hat{\mathbf{A}} = \mathbf{I}$) to minimize $\mathcal{L}_{decreg}$, which also harms the overall performance. Therefore, the observations correspond with TransGTR designs.



(a) METR-LA to PEMSD7M (b) PEMS-BAY to PEMSD7M

Figure 3: Results of parameter analysis on λ_d, from both METR-LA and PEMS-BAY to PEMSD7M. The reported metrics are evaluated with a forecasting horizon of 60 minutes.



(a) METR-LA to PEMSD7M (b) PEMS-BAY to PEMSD7M

Figure 4: Results of parameter analysis on λ_r, from both METR-LA and PEMS-BAY to PEMSD7M. The reported metrics are evaluated with a forecasting horizon of 60 minutes.

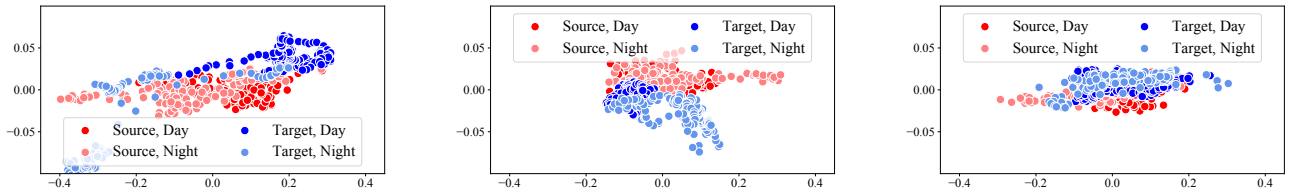
5.5 Case Studies

We present illustrative case studies to help understand the advantages of TransGTR. All studies are performed with METR-LA as source and PEMSD7M as target with 7-day training data.

5.5.1 Graph Structure Analysis. We first analyze how the graph structures learned by TransGTR help the task of traffic forecasting. We compare the following graph structures.

- **Pre-defined**, i.e. the pre-defined graph structure.
- **Random**: We generate an Erdős-Rényi random graph [6] with the same density as the pre-defined graph.
- **Target-only**: We train TransGTR with only target data and obtain the learned graph in the target city.
- **TransGTR**: We train TransGTR with source and target data and obtain the learned graph in the target city.

Intuitively, a graph structure is more suitable for traffic forecasting if it connects nodes with more similar traffic readings, while a graph structure is noisy if it connects nodes with both similar and dissimilar traffic readings. Therefore, for each connected node pair (u, v) in each graph, we compute the distance (RMSE and MAE)



(a) $\lambda_r = 0$, spatial features (b) $\lambda_r = 0$, residual spatial features (c) $\lambda_r = 0.01$, residual spatial features

Figure 5: Visualization of spatial features S_M and residual spatial features $\tilde{S}_M$ obtained from TransGTR with $\lambda_r = 0$ and $\lambda_r = 1$. Red dots represent source features, while blue dots represent target features. In addition, dark dots represent day-time features, while light dots represent night-time features. The axes in all sub-figures are of the same range.

Table 3: Mean RMSE and MAE (± std. dev. *within the graph*) between connected node pairs in different graph structures.

Graph Structures	RMSE	MAE
Random	14.90±4.63	10.00±3.49
Pre-defined	13.86±5.21	9.27±3.79
Target-only	13.80±4.19	9.21±3.12
TransGTR	12.85±4.17	8.43±3.22

between $\mathbf{x}_u$ and $\mathbf{x}_v$ and report the mean values and standard deviations *within each graph*. The results are shown in Table 3, from which we make the following observations.

- The distances between connected node pairs in the pre-defined graph show high variance (even larger than the random graph), indicating that the pre-defined graph is noisy and not optimal for traffic forecasting. By comparison, learned graph structures (target-only and TransGTR) are less noisy, as shown by the lower variances, and are thus more appropriate for traffic forecasting.
- TransGTR achieves the lowest distances between connected node pairs among all studied methods, which, in addition to the quantitative results in Section 5.2, shows that with the knowledge transferred from the source city, TransGTR learns better graph structures for traffic forecasting.

5.5.2 Spatial Feature Analysis. We show visualizations of the learned spatial features to understand how TransGTR learns graph structures that help extract city-invariant knowledge. We train two TransGTR models with $\lambda_r = 0$ and $\lambda_r = 0.01$ and obtain the final-layer spatial features S_M and residual spatial features $\tilde{S}_M$ extracted from source and target data. We use PCA to reduce S_M and $\tilde{S}_M$ to 2 dimensions and plot them in Fig. 5. In addition, we use dark and light points to represent day-time (8am-8pm) and night-time spatial features, respectively. We make the following observations.

- From Fig. 5(a), we observe that day-time and night-time spatial features are highly separated along the x-axis, the direction with the highest variance. Specifically, day-time and night-time spatial features are mostly located on the right and left, respectively. The observation shows that temporal dynamics cause significant variances in spatial features. By comparison, the variance along the x-axis is greatly reduced in Fig. 5(b), indicating that the residual spatial features $\tilde{S}_M$ are more stable and suitable for regularization.
- From Fig. 5(b), we observe that without regularization ($\lambda_r = 0$), the residual spatial features from source and target cities are separated along the y-axis and thus represent city-specific knowledge. By comparison, in Fig. 5(c), residual spatial features from both cities are better mixed with each other, suggesting that with the decoupled regularization, TransGTR learns city-invariant knowledge by generating graph structures that extract spatial features with similar distributions.

6 CONCLUSION

In this paper, we propose TransGTR, a transferable structure learning framework for traffic forecasting that jointly learns and transfers the graph structures and the forecasting models across cities. TransGTR consists of a node feature network, a structure generator, and a forecasting model. We train the node feature network with knowledge distillation to extract city-agnostic node features, such that the structure generator, taking the node features as inputs, can be transferred across both cities. Furthermore, we train the structure generator via a temporal decoupled regularization, such that the spatial features learned with the generated graphs share similar distributions across cities and thus facilitate knowledge transfer for the forecasting model. We evaluate TransGTR on real-world traffic speed datasets, where under a fair comparison, TransGTR outperforms state-of-the-art baselines by up to 5.4%.

We discuss the limitations of TransGTR in Appendix Section E.

ACKNOWLEDGMENTS

We are grateful to Xu Geng for his helpful suggestions. This paper is supported by the Key-Area Research and Development Program of Guangdong Province (2021B0101400001), Hong Kong RGC TRS T41-603/20-R, the National Key Research and Development Program of China under Grant No.2018AAA0101100, and the Turing AI Computing Cloud [41].

REFERENCES

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in neural information processing systems* 33 (2020), 17804–17815.

[2] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79, 1 (2010), 151–175.

[3] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. 2006. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics* 22, 14 (2006), e49–e57.

[4] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in neural information processing systems* 33 (2020), 17766–17778.

[5] Suresh Chavhan and Pallapa Venkataram. 2020. Prediction based traffic management in a metropolitan area. *Journal of traffic and transportation engineering (English edition)* 7, 4 (2020), 447–466.

[6] Paul Erdős, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* 5, 1 (1960), 17–60.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, Vol. 70. PMLR, 1126–1135.

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research* 17, 1 (2016), 2096–2030.

[9] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[10] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

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

[12] Liangzhe Han, Bowen Du, Leilei Sun, Yanjie Fu, Yisheng Lv, and Hui Xiong. 2021. Dynamic and multi-faceted spatio-temporal deep learning for traffic speed forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 547–555.

[13] Xiaolin Han, Tobias Grubenmann, Reynold Cheng, Sze Chun Wong, Xiaodong Li, and Wenya Sun. 2020. Traffic incident detection: A trajectory-based approach. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1866–1869.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.

[15] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[16] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[17] Renhe Jiang, Du Yin, Zhaonan Wang, Yizhuo Wang, Jiewen Deng, Hangchen Liu, Zekun Cai, Jinliang Deng, Xuan Song, and Ryosuke Shibasaki. 2021. DL-Traff: Survey and Benchmark of Deep Learning Models for Urban Traffic Prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4515–4525.

[18] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* (2022), 117921.

[19] Yilun Jin, Kai Chen, and Qiang Yang. 2022. Selective Cross-City Transfer Learning for Traffic Prediction via Source City Region Re-Weighting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 731–741.

[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Yaguang Li, Kun Fu, Zheng Wang, Cyrus Shahabi, Jieping Ye, and Yan Liu. 2018. Multi-task representation learning for travel time estimation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1695–1704.

[22] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*.

[23] Hao Liu, Jindong Han, Yanjie Fu, Yanyan Li, Kai Chen, and Hui Xiong. 2022. Unified route representation learning for multi-modal transportation recommendation with spatiotemporal pre-training. *The VLDB Journal* (2022), 1–18.

[24] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*. PMLR, 97–105.

[25] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2017. Deep transfer learning with joint adaptation networks. In *ICML*. PMLR, 2208–2217.

[26] Tiancheng Lou, Jie Tang, John Hopcroft, Zhanpeng Fang, and Xiaowen Ding. 2013. Learning to predict reciprocity and triadic closure in social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7, 2 (2013), 1–25.

[27] Bin Lu, Xiaoying Gan, Weinan Zhang, Huaxiu Yao, Luoyi Fu, and Xinbing Wang. 2022. Spatio-Temporal Graph Few-Shot Learning with Cross-City Knowledge Transfer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1162–1172. https://doi.org/10.1145/3534678.3539281

[28] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).

[29] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22, 10 (2009), 1345–1359.

[30] Chao Shang, Jie Chen, and Jinbo Bi. 2021. Discrete graph structure learning for forecasting multiple time series. *International Conference on Learning Representations* (2021).

[31] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-Training Enhanced Spatial-Temporal Graph Neural Network for Multivariate Time Series Forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1567–1577. https://doi.org/10.1145/3534678.3539396

[32] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*. Springer, 443–450.

[33] Yihong Tang, Ao Qu, Andy H.F. Chow, William H.K. Lam, S.C. Wong, and Wei Ma. 2022. Domain Adversarial Spatial-Temporal Network: A Transferable Framework for Short-Term Traffic Forecasting across Cities. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 1905–1915. https://doi.org/10.1145/3511808.3557294

[34] Luan Tran, Min Y Mun, Matthew Lim, Jonah Yamato, Nathan Huh, and Cyrus Shahabi. 2020. DeepTRANS: a deep learning system for public bus travel time estimation using traffic forecasting. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2957–2960.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *International Conference on Learning Representations*.

[37] Leye Wang, Xu Geng, Xiaojuan Ma, Feng Liu, and Qiang Yang. 2019. Cross-city Transfer Learning for Deep Spatio-temporal Prediction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1893–1899.

[38] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[39] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 753–763.

[40] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1907–1913.

[41] Kaiqiang Xu, Xinchen Wan, Hao Wang, Zhenghang Ren, Xudong Liao, Decang Sun, Chaoliang Zeng, and Kai Chen. 2021. TACC: A Full-stack Cloud Computing Infrastructure for Machine Learning Tasks. *arXiv preprint arXiv:2110.01556* (2021).

[42] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. 2019. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *The World Wide Web Conference*. 2181–2191.

[43] Junchen Ye, Zihan Liu, Bowen Du, Leilei Sun, Weimiao Li, Yanjie Fu, and Hui Xiong. 2022. Learning the evolutionary and multi-scale graph structure for multivariate time series forecasting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2296–2306.

[44] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3634–3640.

[45] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2021. An effective joint prediction model for travel demands and traffic flows. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 348–359.

[46] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 984–992.

[47] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. 2019. Bridging theory and algorithm for domain adaptation. In *International Conference on Machine Learning*. PMLR, 7404–7413.

[48] Bolong Zheng, Lingfeng Ming, Qi Hu, Zhipeng Lü, Guanfeng Liu, and Xiaofang Zhou. 2022. Supply-Demand-aware Deep Reinforcement Learning for Dynamic Fleet Management. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 3 (2022), 1–19.

[49] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *Proceedings of the*

AAAI conference on artificial intelligence, Vol. 32.

[50] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*. 1215–1226.

[51] Daniel Zügner, François-Xavier Aubet, Victor Garcia Satorras, Tim Januschowski, Stephan Günnemann, and Jan Gasthaus. 2021. A study of joint graph inference and forecasting. *arXiv preprint arXiv:2109.04979* (2021).

Algorithm 1: Transferable Graph Structure Learning for Graph-based Traffic Forecasting with TransGTR

Input: Source and target traffic data $\mathbf{X}_{\mathcal{S}}, \mathbf{X}_{\mathcal{T}}$,
TSFormer $\theta_{nf,\mathcal{S}}$ pre-trained with $\mathbf{X}_{\mathcal{S}}$
Output: A node feature network $f_{\theta_{nf}}$,
a structure generator f_{ϕ},
and a forecasting model f_{θ} for the target city $\mathcal{T}$

1: `//Train the node feature network` $f_{\theta_{nf}}$
2: **while** $f_{\theta_{nf}}$ has not converged **do**
3: Sample $\mathbf{x}_{v_{\mathcal{S}}} \in \mathbf{X}_{\mathcal{S}}, \mathbf{x}_{v_{\mathcal{T}}} \in \mathbf{X}_{\mathcal{T}}$.
4: Optimize θ_{nf} via Eqn. 10 with data $\mathbf{x}_{v_{\mathcal{S}}}, \mathbf{x}_{v_{\mathcal{T}}}$.
5: **end while**
6: Compute $\mathbf{G}_{\mathcal{S}}, \mathbf{G}_{\mathcal{T}}$ via Eqn. 5.
7: `//Train the structure generator` f_{ϕ} `and forecasting model` f_{θ}
8: **while** f_{θ}, f_{ϕ} have not converged **do**
9: Sample $\mathbf{S}_{\mathcal{S},0} = [\mathbf{X}_{\mathcal{S}}^{t-k_i+1}, \dots, \mathbf{X}_{\mathcal{S}}^{t}], t < T_{\mathcal{S}} - k_o$.
10: Sample $\mathbf{S}_{\mathcal{T},0} = [\mathbf{X}_{\mathcal{T}}^{t'-k_i+1}, \dots, \mathbf{X}_{\mathcal{T}}^{t'}], t' < T_{\mathcal{T}} - k_o$.
11: Optimize θ, ϕ via Eqn. 13 with data $\mathbf{S}_{\mathcal{S},0}, \mathbf{S}_{\mathcal{T},0}$.
12: Optimize $\{\theta_{dec,m}\}$ via Eqn. 12.
13: **end while**
14: Fine-tune θ, ϕ with target data.
15: **return** Trained models $f_{\theta_{nf}}, f_{\theta}, f_{\phi}$.

Dataset	$\lvert\mathcal{V}\rvert$	Density	# Time Steps	Sample Rate
METR-LA	207	0.066	34272	5 min
PEMS-BAY	325	0.027	52116	
PEMSD7M	228	0.087	12672	
HKTD	608	0.059	30240	

Table 4: Statistics of the datasets.

A DETAILS OF DATASETS

We use four traffic speed datasets, METR-LA, PEMS-BAY, PEMSD7M, and HKTD for evaluation. Among them, METR-LA, PEMS-BAY, and PEMSD7M come from DL-Traff[2] [17], and HKTD comes from the Hong Kong Transport Department. Statistics of all datasets are shown in Table 4. For all datasets, we apply Z-score normalization, i.e. normalizing the data such that their mean values are 0 and standard deviations are 1. For METR-LA, PEMS-BAY, and PEMSD7M, we adopt their pre-defined graph structures for baselines (TransGTR does not require pre-defined graph structures). For HKTD, we define its graph structure by computing pairwise distances between sensors, and then applying thresholded Gaussian kernels.

We split the data into train, validation, and test data by 7:1:2 along the temporal axis. For source cities, we only use the training set to train models, and use the validation set for model selection. For target cities, we use the validation set for parameter tuning, the test set for evaluation, and the 7/3-day training data are sampled at the end of the training set (i.e. right before the validation set).

[2]https://github.com/deepkashiwa20/DL-Traff-Graph

B DETAILS OF BASELINES

We present details of baseline methods in this section.

- **ARIMA**: We adopt ARIMA models with 12 AR steps, 1 integration step, and 1 MA step.
- **GWN**: We adopt the official implementation[3].
- **GTS**: We adopt the official implementation[4] and change the base model to GWN.
- **FT-GWN**, **FT-GTS**: We train the models on source data and use the source model with the lowest source validation error to initialize target fine-tuning.
- **RegionTrans**: We implement RegionTrans as we fail to find code for it. We use S-Match, i.e. we use a short period of traffic data to compute region similarity, because we do not have auxiliary data. We manually tune the parameter w and set it at $w = 0.001$.
- **DASTNet**: We adopt the official implementation[5] and change the base model to GWN.
- **ST-GFSL**: We adopt the official implementation[6].

For all baselines with official implementations, we use the default hyperparameters.

We note that there are other structure learning methods for traffic forecasting besides GTS, such as MTGNN [39] and AGCRN [1]. However, we do not take FT-MTGNN or FT-AGCRN as suitable baselines. The reason is that MTGNN and AGCRN directly optimize a node embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ to learn the graph structure $\hat{\mathbf{A}}$. As the embedding matrix $\mathbf{E}$ is transductive and cannot be reused on another dataset, FT-MTGNN and FT-AGCRN cannot transfer knowledge contained in $\mathbf{E}$ from one city to another. On the other hand, GTS learns the graph structure $\hat{\mathbf{A}}$ by extracting features from the whole time series $\mathbf{X}$, and thus, the knowledge contained in GTS can be transferred to another city via fine-tuning, making FT-GTS a more powerful baseline for cross-city transfer learning.

C IMPLEMENTATION DETAILS OF TRANSGTR

C.1 Base Models

We implement TSFormers following the official code in STEP[7]. We take GTS [30] and GWN [40] as base structure generators and forecasting models, respectively. Given the features learned by the node feature network,

$$\begin{aligned} \mathbf{G}'_{\mathcal{S}} &= f_{\theta_{nf}}(\mathbf{X}_{\mathcal{S}}) \in \mathbb{R}^{(T_{\mathcal{S}}/P) \times n_{emb}}, \\ \mathbf{G}_{\mathcal{T}} &= f_{\theta_{nf}}(\mathbf{X}_{\mathcal{T}}) \in \mathbb{R}^{(T_{\mathcal{T}}/P) \times n_{emb}}, \end{aligned} \tag{14}$$

we split $\mathbf{G}'_{\mathcal{S}}$ into batches of length $T_{\mathcal{T}}/P$ as $\mathbf{G}'_{\mathcal{S}} = [\mathbf{G}_{\mathcal{S},1}, \mathbf{G}_{\mathcal{S},2}, \dots]$, and obtain $\mathbf{G}_{\mathcal{S}} = \text{MEAN}\left(\left[\mathbf{G}_{\mathcal{S},1}, \mathbf{G}_{\mathcal{S},2}, \dots\right]\right)$. This operation is to ensure that $\mathbf{G}_{\mathcal{S}}$ and $\mathbf{G}_{\mathcal{T}}$ share the same input dimensions. Then, to learn graph structures from temporal features $\mathbf{G}$, GTS first applies a temporal feature extractor to obtain node features

$$\mathbf{Z} = \text{Conv1d}\left(\text{ReLU}\left(\text{Conv1d}\left(\mathbf{G}\right)\right)\right), \tag{15}$$

[3]https://github.com/nnzhan/Graph-WaveNet
[4]https://github.com/chaoshangcs/GTS
[5]https://github.com/YihongT/DASTNet
[6]https://github.com/RobinLu1209/ST-GFSL
[7]https://github.com/zezhishao/STEP

Baselines	Target City	PEMSD7M								HKTD							
	Data Number	7-day				3-day				7-day				3-day			
	Horizon	30 mins		60 mins		30 mins		60 mins		30 mins		60 mins		30 mins		60 mins	
Target Only	ARIMA	9.19		14.38		9.28		14.54		11.11		15.23		11.11		15.27	
	GWN	7.68		10.29		7.88		11.32		9.45		12.15		9.93		12.87	
	GTS	7.57		9.89		7.76		10.72		9.54		11.63		10.04		12.54	
Transfer	**Source**	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY	LA	BAY
	FT-GWN	7.32	7.47	9.44	9.56	7.67	7.79	10.09	10.66	8.95	8.99	10.50	10.72	9.42	9.40	11.54	11.59
	FT-GTS	7.39	7.33	9.26	9.18	7.66	7.78	9.81	9.81	9.06	8.81	10.43	10.06	9.42	9.46	11.00	11.33
	RegionTrans	7.29	7.37	9.39	9.58	7.66	7.75	10.13	10.51	8.90	8.89	10.49	10.42	9.45	9.32	11.47	11.39
	DASTNet	7.31	7.26	9.27	9.37	7.66	7.77	9.96	10.09	8.96	8.91	10.34	10.34	9.41	9.43	11.22	11.34
	ST-GFSL	7.43	7.29	9.31	9.27	7.70	7.75	10.10	9.90	9.00	9.06	10.62	10.55	9.56	9.51	11.36	11.47
	TransGTR	**7.00**	**6.99**	**8.67**	**8.78**	**7.49**	**7.45**	**9.51**	**9.42**	**8.75**	**8.68**	**9.76**	**9.80**	**9.28**	**9.18**	**10.64**	**10.55**
	Std. Dev.	0.05	0.02	0.08	0.08	0.06	0.05	0.13	0.07	0.04	0.05	0.05	0.06	0.04	0.06	0.11	0.06

Table 5: MAPE evaluation results (%). LA and BAY stand for METR-LA and PEMS-BAY as source cities, respectively. In each column, the best result is presented in bold and the second best is underlined.

where Conv1d denotes 1D convolution on the temporal dimension, $\text{ReLU}(x) = \max(0, x)$, and $\mathbf{G}$ can be $\mathbf{G}_{\mathcal{S}}$ or $\mathbf{G}_{\mathcal{T}}$. Then, GTS applies a predictor to output the edge probability between nodes i, j

$$\Theta_{ij} = \sigma\left(\text{FC}\left(\text{ReLU}\left(\text{FC}\left(\mathbf{Z}_i \| \mathbf{Z}_j\right)\right)\right)\right), \tag{16}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, FC denotes a fully connected layer, $\|$ denotes vector concatenation, and $\mathbf{Z}_i$ denotes the node features of node i from Eqn. 15. Finally, GTS samples a discrete graph structure $\hat{\mathbf{A}}$ from Θ as

$$\hat{\mathbf{A}}_{ij} = \text{GumbelSample}(\Theta_{ij}, \tau), \tag{17}$$

where GumbelSample denotes the Gumbel-Softmax reparameterization [16] used to approximate Bernoulli variables, and τ is the temperature parameter. $\hat{\mathbf{A}}_{ij}$ converges to discrete values as $\tau \to 0$.

For GTS, we follow its official implementation and set the temperature parameter as 0.5. For GWN, we also follow its official implementation and enable the adaptive adjacency matrix.

C.2 Hyperparameters

We set the source TSFormer $f_{\theta_{nf},\mathcal{S}}$ and the node feature network $f_{\theta_{nf}}$ to have 4 encoder layers, 1 decoder layer, 96 hidden features, 4 attention heads, and a mask ratio of 0.75. We take the pre-trained TSFormers on METR-LA and PEMS-BAY from STEP as the source TSFormers , trained with sequence length $L \cdot P = 2016$ (i.e. a week), and patch size $P = 12$ (i.e. an hour). For knowledge distillation, we set $L_{short} \cdot P = 864$ (i.e. 3 days). We set $\lambda_d = 1$, and optimize Eqn. 10 for 100 epochs with learning rate 0.0005.

To learn the structure generator and the forecasting model, we use 2-layer multilayer perceptrons with 64 hidden units as the decoupling modules $\theta_{dec,m}$. We set $\lambda_r = 0.01$. We alternately optimize Eqn. 13 and Eqn. 12 for 30 epochs with learning rate 0.001. For target fine-tuning, we fine-tune f_ϕ, f_θ on target data with learning rate 0.001. We adopt Adam optimizers with weight decay 1e-5.

We implement TransGTR with PyTorch 1.9.0 and run all experiments on an NVIDIA Tesla V100 GPUs with 32GB GPU memory.

D ADDITIONAL RESULTS OF MAPE

In this section, we present additional evaluation results with MAPE as the metric. We follow the same evaluation procedure as Section 5.2 and report the MAPE results in Table 5. We make similar observations that TransGTR consistently outperforms baselines with an improvement of up to 6.4%.

E LIMITATIONS AND FUTURE WORK

One major limitation of TransGTR is that, although it achieves competitive results in longer forecasting horizons (e.g. 60 minutes ahead), it is less competitive in short-term forecasting. For example, as shown in Table 1, while TransGTR achieves an improvement of up to 5% when the forecasting horizon is 60 minutes, it only achieves a 1-2% improvement when the horizon is 30 minutes. We hypothesize that the difference is caused by the fact that the optimal graph structures for short-term and long-term forecasting are different, and thus, an optimal graph structure for 60-minute forecasting may not work well for 30-minute forecasting. We consider it as an important future work.

In addition, TransGTR currently works only for one source city and one target city, while in practice, it is generally more attractive to incorporate more source knowledge by learning with multiple source cities. Therefore, we also identify transferable graph structure learning across more cities as an important future work.