Drawing Informative Gradients from Sources: A One-stage Transfer Learning Framework for Cross-city Spatiotemporal Forecasting

Yudong Zhang[1], **Xu Wang**[1,2*], **Xuan Yu**[1], **Zhaoyang Sun**[1], **Kai Wang**[1], **Yang Wang**[1,2,3*]
[1]University of Science and Technology of China
[2]Suzhou Institute for Advanced Research, USTC
[3]Key Laboratory of Precision and Intelligent Chemistry, USTC
{zyd2020@mail., wx309@, yx2024@mail., sunzhaoyang@mail., zaizwk@mail., angyan@}ustc.edu.cn

Abstract

Spatiotemporal forecasting (STF) is pivotal in urban computing, yet data scarcity in developing cities hampers robust model training. Addressing this, recent studies leverage transfer learning to migrate knowledge from data-rich (source) to data-poor (target) cities. This strategy, while effective, faces challenges as pre-trained models risk absorbing noise and harmful information due to data distribution disparities, potentially undermining the accuracy of forecasts for target cities. To address this issue, we propose a one-stage STF framework named Target-Skewed Joint Training (TSJT). Central to TSJT is a novel Target-Skewed Backward training strategy that selectively refines gradients from source city data, preserving only the elements that positively impact the target city. To further enhance the quality of these gradients, we have designed a Node Prompting Module (NPM). TSJT is crafted for seamless integration with existing STF models, endowing them with the capability to efficiently tackle challenges stemming from data scarcity. Experimental results on several real-world datasets from multiple cities substantiate the efficacy of TSJT in the realm of cross-city transfer learning.

Introduction

Spatiotemporal forecasting (STF) models facilitate various smart-city applications across various domains, e.g., traffic (Zhang et al. 2024; Wang et al. 2023), and climate (Wu et al. 2023; Chen et al. 2023). Leveraging the foundation of extensive spatiotemporal data, numerous deep learning models (Li et al. 2018; Wu et al. 2019; Rao et al. 2022; Yuan et al. 2024a) have been proposed, achieving remarkable success in the realm of predictive accuracy. However, the uneven development levels and disparate data collection policies across cities often result in a scarcity of spatiotemporal data in many cities. This scarcity presents a significant barrier to the effective application of STF models in these cities.

The analogous patterns of human activity across various cities have inspired researchers to develop cross-city transfer learning strategies (Wei, Zheng, and Yang 2016). These strategies capitalize on the vast datasets from data-rich cities, known as *source cities*, to extract generalizable and transferable knowledge. The aim is to apply this learned knowledge effectively to enhance spatiotemporal forecasting in cities with data limitations, referred to as *target cities*. Recent cross-city transfer learning methods (Ouyang et al. 2024; Chen, Liu, and Li 2023; Tang et al. 2022) are mostly composed of two procedures, learning transferable knowledge from source cities and fine-tuning on target cities. The strategies of existing works on learning transferable knowledge from source cities can be divided into two categories: (1) Designing crafty model architectures and pre-training the models on source cities with ingenious strategies. DASTNet (Tang et al. 2022) leverages the adversarial domain adaptation techniques to learn the domain-invariant node embeddings, which are further incorporated to model the temporal traffic data. TPB (Liu, Zheng, and Yu 2023) pre-trains a masked autoencoder, uses the autoencoder to generate a spatiotemporal pattern bank by clustering embeddings of spatiotemporal data, and fine-tunes the pattern bank along with an STF model for target city. (2) Learning to generate initial parameters of forecasting models and fine-tuning the models on the target city. ST-GFSL (Lu et al. 2022) learns a hypernetwork trained on source cities, which generates parameters of the STF model for the target city based on node-level meta knowledge. GPD (Yuan et al. 2024b) recasts spatiotemporal few-shot learning as pre-training a generative diffusion model, which generates tailored neural networks guided by prompts, allowing for adaptability to diverse data distributions and city-specific characteristics.

Challenge. Though the two categories of methods employ diverse learning strategies, they are united by a common principle that, by constructing learning tasks on data from source cities, models can acquire knowledge that is transferable not only across source cities but also anticipated to be applicable to the target city. However, *data distributions between source and target cities often exhibit considerable divergence, which results in the acquisition of knowledge that includes not only potential noise but also information that may be counterproductive* (Jin, Chen, and Yang 2022; Yuan et al. 2024b). This discrepancy presents significant hurdles to the successful transfer of knowledge to a target city. Consequently, models trained on the extensive source data may assimilate noise or even detrimental source-specific knowledge, adversely affecting the efficacy of transfer learning when applied to the target city. **How to learn the beneficial information for the target city from the data from source cities remains an open and**

*Xu Wang and Yang Wang are the corresponding authors

challenging problem.

Insight. To tackle this challenge, we introduce a novel perspective on cross-city transfer learning. The data scarcity in the target city results in limited search space for the forecasting model's parameters. Considering that existing methodologies require a fine-tuning procedure, they essentially aim to identify the initial parameters of the forecasting model. When the model undergoes fine-tuning specific to the target city, the initial parameters can shift the search space to location easier to find well-performing model parameters. From this point, **cross-city transfer learning is to adjust the parameter searching space of the forecasting model on the target city based on data from source cities.**

Improvement. In light of the aforementioned analysis, this paper introduces an elegant one-stage learning framework that trains STF models directly on data from both source and target cities. To prevent the training process from being dominated by data from source cities, we propose a novel Target-Skewed Backward (TSB) training strategy, which selectively refines the gradients from source city data, retaining only the components that are beneficial to the target city. Furthermore, to enhance the quality of gradients derived from source city data, we design a Node Prompting Module (NPM). The NPM guides the model to achieve similar gradients on nodes that exhibit similar spatiotemporal patterns. The combination of TSB and NPM enables dynamic adaptation of the parameter search space of the forecasting model, continuously steering it toward configurations that are advantageous for the target city. Our proposed Target-Skewed Joint Training framework is designed for seamless integration with existing STF models, providing them with the enhanced ability to effectively address challenges arising from data scarcity. The **contributions** of this paper are summarized as follows,

- *Novel insight and framework:* We introduce a novel perspective on cross-city transfer learning, redefining it as the process of shifting the parameter search space of the forecasting model on a target city using data from source cities. A Target-Skewed Joint Training (TSJT) framework is introduced, which enables existing STF models to address data scarcity challenges effectively.
- *Advisable methodologies:* We propose a novel Target-Skewed Backward (TSB) training strategy that refines the gradients from source city data, preserving only the component advantageous for the target city. Additionally, we develop a Node Prompting Module (NPM) to guide the model to generate analogous gradients on nodes with similar spatiotemporal characteristics, thereby enriching the gradient quality.
- *Compelling empirical results:* We conduct comprehensive cross-city few-shot learning experiments across four real-world urban datasets. The results corroborate the proficiency of TSJT in addressing cross-city transfer learning.

Related Work

Spatiotemporal forecasting, vital for diverse applications, faces challenges in data-scarce developing cities due to high collection costs. Cross-city knowledge transfer promises a solution, leveraging models trained on data-rich cities to enhance learning in data-poor environments (Ouyang et al. 2024; Chen, Liu, and Li 2023; Tang et al. 2022).

Current transfer learning methods involve extracting transferable knowledge from source cities and then fine-tuning for target cities. Strategies include (1) crafting model architectures with pre-training techniques to match regions and generate graph structures that mimic source domains (Wei, Zheng, and Yang 2016; Jin, Chen, and Yang 2023; Hu et al. 2024). RegionTrans (Wei, Zheng, and Yang 2016) learns to match regions of target city to similar source city regions, and extract region-level representation for spatiotemporal forecasting. TransGTR (Jin, Chen, and Yang 2023) proposes to generate graph structures for the target domain that exhibit structural distributions akin to the source domain. (2) employing meta-learning to initialize spatial-temporal networks broadly (Jin, Chen, and Yang 2022; Yao et al. 2019). CrossTReS (Jin, Chen, and Yang 2022) trains a weighting network via source-target joint meta-learning such that source regions helpful to target fine-tuning are assigned high weights. MetaST (Yao et al. 2019) utilizes meta-learning paradigm to learn a well-generalized initialization of the spatial-temporal network, and proposes a pattern-based spatial-temporal memory to distill long-term temporal information.

Despite varied strategies, the core idea is to develop models that learn from source cities and adapt to the target city. However, divergences in data distributions between cities can lead to the acquisition of noisy or unhelpful knowledge, complicating effective knowledge transfer. The challenge of extracting beneficial information for the target city from source city data remains an open issue in the field.

Preliminary

Definition 1 (Spatiotemporal Graph). The spatiotemporal graph can be denoted as $\mathbf{G} = \{\mathbf{A}, \mathbf{X}\}$. $\mathbf{X} \in \mathbb{R}^{N \times T \times D}$ denote the D-dimension features at all T time steps of all N nodes. In the following, we also have $\mathbf{X}^t \in \mathbb{R}^{N \times D}$ denoting features of all nodes at a specific time step t. $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of spatiotemporal graph, where $A_{ij} = 1$ means there exists an edge between node x_i and x_j, $A_{ij} = 0$ means there exists no edge.

Problem 1 (Spatiotemporal Forecasting). Given adjacency matrix $\mathbf{A}$, the current time t and L-step historical spatiotemporal graph data, denoted as $\mathbf{X}^{t-L+1:t} = \{\mathbf{X}^i | i \in [t-L+1, t]\}$, the problem of spatiotemporal graph forecasting is to learn a function $f_\theta(\cdot)$ with parameter as θ which predicts the Q-step future spatiotemporal graph data. The problem can be formulated as,

$$\mathbf{X}^{t+1:t+Q} = f_\theta(\mathbf{X}^{t-L+1:t}; A). \tag{1}$$

Problem 2 (Spatiotemporal Few-shot Forecasting). Given S source cities with plenty of data, denoted as $G_{1:S}^{source} = \{G_1^{source}, G_2^{source}, \cdots, G_S^{source}\}$, and a target city G^{target} with few-shot data, spatiotemporal few-shot forecasting aims at learning a well-performed forecasting model on target city based data from both source and target cities, $\{G_{1:S}^{source}, G^{target}\}$.

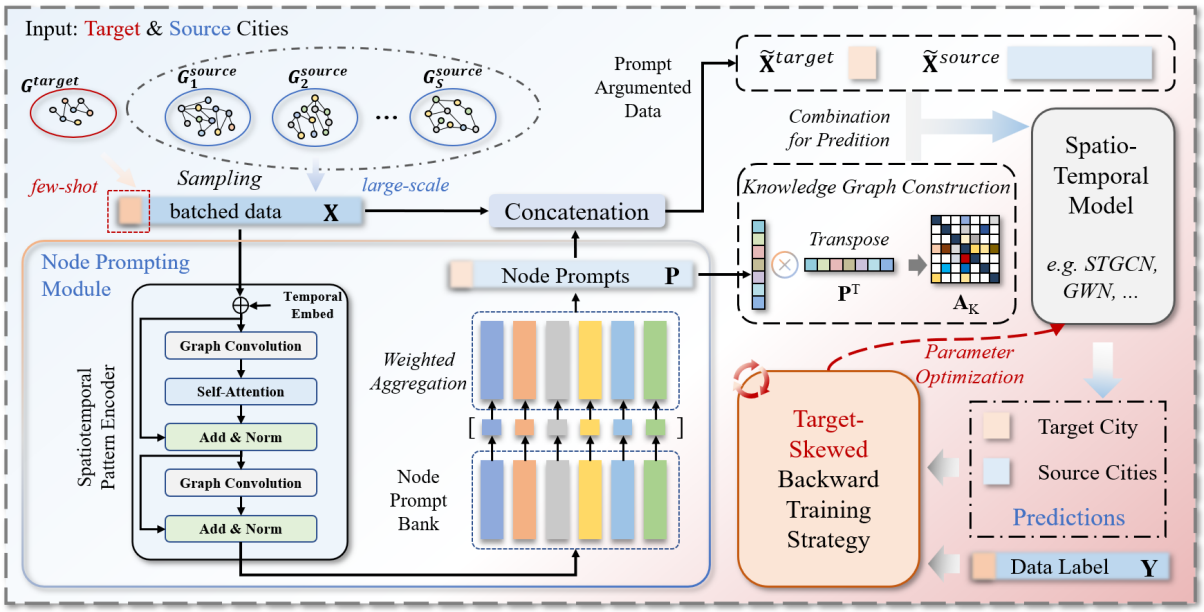


Figure 1: **Overview.** The framework comprises two key innovative components: the Node Prompting Module (NPM) and the Target-Skewed Backward (TSB) training strategy. The NPM curates a repository of shared spatiotemporal patterns, creating analogous prompts for nodes with matching patterns, thereby enhancing the TSB strategy. TSB selectively refines gradients from source cities, retaining only the elements that are advantageous for the target city. Together, NPM and TSB fortify spatiotemporal models against the challenges of few-shot learning, offering an effective solution for cross-city knowledge transfer.

Method

In this section, the proposed Target-Skewed Joint Training (TSJT) framework will be detailed. As shown in Fig.1, the proposed framework is an end-to-end transfer learning framework, which serves as an enhancement, i.e., add-on component, to vanilla spatiotemporal graph forecasting models, facilitating their application in few-shot learning scenarios on the target city. The TSJT framework encompasses two innovative components: a Target-Skewed Backward (TSB) training strategy and a Node Prompting Module (NPM). Specifically, a novel node prompting module (NPM) is introduced to encode shared spatiotemporal patterns across diverse datasets, guiding the framework to obtain similar gradients for nodes with comparable spatiotemporal patterns, thereby refining the overall quality of the gradient information. Additionally, we propose a novel target-skewed backward training strategy. This innovative approach facilitates seamless end-to-end training across both the target and source datasets, enhancing the overall learning process. As depicted in Fig.1, the NPM and TSB can be integrated with existing spatiotemporal graph learning models. This integration effectively tackles the challenges posed by few-shot learning, providing a robust solution for cross-city knowledge transfer.

Node Prompting Module

Nodes located in different cities may share similar spatiotemporal patterns. For instance, cities always have residential and commercial areas, with people periodically commuting between them. The patterns of human flow in residential areas in different cities are similar. Therefore, it is reasonable that the STF model generates similar predictions on nodes with similar spatiotemporal patterns. Inspired by such observation, a Node Prompting Module (NPM) is proposed, which maintains a bank of node prompts encoding spatiotemporal patterns shared by nodes in different cities. NPM utilizes an encoder to extract spatiotemporal patterns from the data. NPM then generates the prompt of a node by utilizing the attention mechanism, with the extracted pattern as Query and node prompts in the maintained bank as Key and Value.

Spatiotemporal pattern encoder NPM extracts spatiotemporal patterns of nodes with a spatiotemporal pattern encoder. The encoder is a modified self-attention layer, where the feed-forward linear layer is replaced by a graph convolution layer. Given feature of a spatiotemporal graph $\mathbf{X} \in \mathbb{R}^{N \times T \times D}$, the pattern encoder first adds it with temporal embedding $\mathbf{E_T}$ as in (Vaswani 2017). The adding result $\mathbf{Z_0} = \mathbf{X} + \mathbf{E_T}$ is fed into a graph convolution layer followed by a self-attention layer,

$$\mathbf{Z_1} = \text{Attention}(\text{GNN}(\mathbf{Z_0})). \tag{2}$$

$\mathbf{Z_1}$ is then added with $\mathbf{Z_0}$ and normalized as,

$$\mathbf{Z_2} = \text{Norm}(\mathbf{Z_1} + \mathbf{Z_0}). \tag{3}$$

Layer normalization (Ba, Kiros, and Hinton 2016) is employed here. Then we have another graph convolution layer, adding and normalization,

$$\mathbf{Z} = \text{Norm}(\mathbf{Z_2} + \text{GNN}(\mathbf{Z_2})), \tag{4}$$

where $\mathbf{Z} \in \mathbb{R}^{N \times T \times D'}$ will be further utilized to generate node prompts.

Node prompts generation As forementioned, NPM maintains a set of K shared patterns termed as node prompt bank $\mathbf{B} = \{\mathbf{B_k} \in \mathbb{R}^{T\times D} | k \in [1, K]\}$. Each prompt is a $T \times D$ matrix encoding what we termed as a meta pattern. Given spatiotemporal pattern of a specific node extracted by pattern encoder $Z_i \in \mathbb{R}^{T\times D'}$, the prompt of this node is calculated by utilizing attention mechanism on Z_i and $\mathbf{B}$, where Z_i serves as *Query*, $\mathbf{B}$ serves as *Key* and *Value*.

As depicted in the middle of Fig.1, upon receiving a node feature X_i, NPM computes the attention score for each meta pattern B_k. Subsequently, the meta prompts within the node prompt bank $\mathbf{B}$ are aggregated through a weighted sum based on their respective attention scores. The result of weighted summation is the output node prompt $P_i \in \mathbb{R}^{T\times D}$ corresponding to node X_i. The process can be formulated as

$$P_i = \frac{Z_i \mathbf{B}^T \mathbf{B}}{\sqrt{D}}. \tag{5}$$

Noting that $\mathbf{B}$ is a 3-dimension matrix, we specify that both the transpose and matrix multiplication operations on $\mathbf{B}$ are performed on the last two dimensions of $\mathbf{B}$. Consequently, the result of multiplying X_i by $\mathbf{B}^T$ yields a tensor of dimensions as $\mathbb{R}^{K\times T\times T}$.

With incorporated prompts of all nodes denoted as $\mathbf{P} = [P_1, P_2, \cdots, P_N]$, we concatenate them with the input node feature $\mathbf{X} = [X_1, X_2, \cdots, X_N]$ along the feature dimension.

$$\widetilde{\mathbf{X}} = [X_1||P_1, X_2||P_2, \cdots, X_N||P_N], \tag{6}$$

where $||$ denotes the concatenation operation, $\widetilde{\mathbf{X}} \in \mathbb{R}^{N\times T\times (D+D')}$ is the concatenation result, serving as the input to the utilized STF backbone model.

Additionally, considering that node prompts encode spatiotemporal patterns, it is highly probable that nodes with similar prompts are correlated, as they share similar spatiotemporal patterns. Therefore, it is reasonable to construct an adjacency matrix by calculating similarities between the prompts of different nodes. Concretely, with prompts of all nodes $\mathbf{P}$, we define such knowledge graph A_K as,

$$A_K(i,j) = \frac{< P_i, P_j >_F}{||P_i||_F ||P_j||_F}, \tag{7}$$

where $< \cdot, \cdot >_F$ represents the Frobenius inner product of two matrices, and $|| \cdot ||_F$ means Frobenius norm [1]. Subsequently, the derived knowledge graph A_K is combined with $\widetilde{\mathbf{X}}$ and utilized as input to the STF backbone model.

Target-skewed backward training strategy

The cornerstone of our end-to-end framework lies in the joint training of both the source and target datasets. However, due to the limited size of the target dataset, the joint training process is susceptible to being dominated by the source datasets, leading to unsatisfying performance on the target dataset. To address this challenge, we introduce the novel Target-Skewed Backward (TSB) training strategy. **This strategy serves two critical functions:** 1) It skews the joint training toward the target dataset, thereby preventing it from being dominated by the much larger source datasets; and 2) It ensures that the model can effectively learn valuable knowledge from the source datasets.

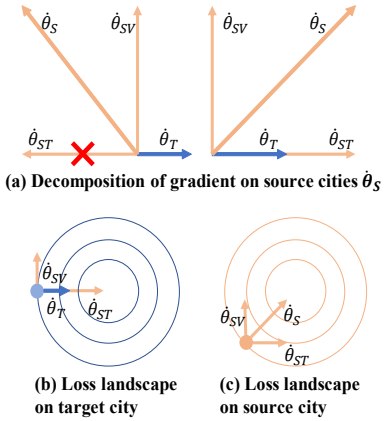


Figure 2: **Illustration of TSB.** (a) Decomposition of $\dot{\theta}_S$. $\dot{\theta}_S$ is decomposed into $\dot{\theta}_{ST}$ and $\dot{\theta}_{SV}$. When $\dot{\theta}_{ST}$ is in the reverse direction to $\dot{\theta}_T$, it will be dropped. (b) Using $\dot{\theta}_{SV}$ and $\dot{\theta}_{ST}$ will not hurt the loss on target city. (c) Using $\dot{\theta}_{SV}$ or $\dot{\theta}_{ST}$ will lead to better loss on source cities.

The target-skewed backward training strategy is a gradient-based approach that aims to preserve the beneficial aspects of the gradient of trainable weights computed on the source datasets while ensuring a reduction in loss on both the target and source datasets. Specifically, let θ represent all trainable parameters within our framework, let $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_B\}$ and $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_B\}$ denote a batch of samples and their corresponding labels, respectively, drawn from both the source and target datasets, and let L be the loss function. Initially, we segment the batched data $\mathcal{X}$ into two subsets: $\mathcal{X}_T$, which includes samples from the target dataset, and $\mathcal{X}_S$, which comprises samples from the source datasets. Similarly, we have $\mathcal{Y}_T$ and $\mathcal{Y}_S$ as the corresponding labels of $\mathcal{X}_T$ and $\mathcal{X}_S$. Subsequently, we compute the gradients for the parameters θ with respect to each subset $\mathcal{X}_T$ and $\mathcal{X}_S$,

$$\begin{aligned} \dot{\theta}_T &= \frac{\partial L(\mathcal{Y}_T, f_\theta(\mathcal{X}_T))}{\partial \theta}, \\ \dot{\theta}_S &= \frac{\partial L(\mathcal{Y}_S, f_\theta(\mathcal{X}_S))}{\partial \theta}, \end{aligned} \tag{8}$$

where $\dot{\theta}_T$ and $\dot{\theta}_S$ represent the gradients of θ with respect to each subset $\mathcal{X}_T$ and $\mathcal{X}_S$, respectively. Given that the framework consists of numerous components or neural network layers, for notational simplicity, we will henceforth consider θ as the parameters of a single component. The following procedure can be readily extended to encompass all components within the framework.

During training, the proposed TSB prioritizes the optimization of θ by predominantly considering the direction of $\dot{\theta}_T$, i.e., $\dot{\theta}_T$ is directly utilized to optimize θ. Additionally, TSB decomposes $\dot{\theta}_S$ into two components, $\dot{\theta}_{ST}$, which is parallel

[1] https://en.wikipedia.org/wiki/Frobenius_inner_product

to $\dot{\theta}_T$, and $\dot{\theta}_{SV}$, which is perpendicular to $\dot{\theta}_T$. Noting that $\dot{\theta}_T$ could be a matrix, the decomposition is implemented by applying Frobenius inner product, formalized as,

$$\begin{aligned} \dot{\theta}_{ST} &= \frac{< \dot{\theta}_S, \dot{\theta}_T >_F}{< \dot{\theta}_T, \dot{\theta}_T >_F} \theta_T, \\ \dot{\theta}_{SV} &= \dot{\theta}_S - \dot{\theta}_{ST}. \end{aligned} \tag{9}$$

TSB employs distinct approaches to utilize each component for optimizing θ:

- 1) $\dot{\theta}_{SV}$ is employed directly in the optimization of θ. Given that $\dot{\theta}_{SV}$ is perpendicular to $\dot{\theta}_T$, utilizing $\dot{\theta}_{SV}$ for optimizing θ does not adversely affect the loss on the target data $\mathcal{X}_T$, while leading to smaller loss on source data $\mathcal{X}_S$.
- 2) When $\dot{\theta}_{ST}$ is in the same direction as $\dot{\theta}_T$, it can be directly utilized to optimize θ, which typically results in a reduced loss for both the source and target datasets. Conversely, when $\dot{\theta}_{ST}$ is in the opposite direction to $\dot{\theta}_T$, it is discarded to prevent adverse effects on the optimization process.

The second approach is intuitive, we here further explain the first approach. As shown in Fig. 2, there is always an acute angle between $\dot{\theta}_{SV}$ and $\dot{\theta}_S$. Therefore, moving in the direction of $\dot{\theta}_{SV}$ will decrease the loss function $L(\mathcal{Y}_S, f_\theta(\mathcal{X}_S))$. On the other hand, since $\dot{\theta}_{SV}$ is perpendicular to $\dot{\theta}_T$, moving in the direction of $\dot{\theta}_{SV}$ will not change the value of loss function $L(\mathcal{Y}_T, f_\theta(\mathcal{X}_T))$. Therefore, by optimizing parameters in the direction of $\dot{\theta}_{SV}$, we obtain a small loss on source cities and shift the searching space on target city (changing parameters without changing the loss on target city), which is absolutely we want. Next, we detail how we update the parameters.

As shown in Fig. 1, given batched data $\widetilde{\mathbf{X}}$, the framework process $\widetilde{\mathbf{X}}^{target}$ and $\widetilde{\mathbf{X}}^{source}$ separately. TSB then calculates the corresponding gradients of $\widetilde{\mathbf{X}}^{target}$ and $\widetilde{\mathbf{X}}^{source}$, obtaining $\dot{\theta}_T$ and $\dot{\theta}_S$. After decomposing $\dot{\theta}_S$ according to Eq. 9, we have $\dot{\theta}_T$, $\dot{\theta}_{SV}$ and $\dot{\theta}_{ST}$. The optimization of θ can be formulated as,

$$\theta = \theta - \gamma_1 \dot{\theta}_T - \gamma_2 (\dot{\theta}_{SV} + \mathrm{dir}(\dot{\theta}_{\mathrm{T}}, \dot{\theta}_{\mathrm{ST}}) \dot{\theta}_{\mathrm{ST}}), \tag{10}$$

where $\mathrm{dir(a, b)}$ returns 1 if a and b are with the same direction, otherwise 0. γ_1 and γ_2 are the learning rates corresponding to target city and source cities, respectively.

Spatiotemporal forecasting backbone model

As mentioned, the proposed framework is an add-on component that facilitates applications of existing spatiotemporal forecasting models in few-shot learning scenarios. In this paper, we utilize two representative and relatively simple spatiotemporal learning models, i.e., STGCN (Yu, Yin, and Zhu 2017) and GWN (Wu et al. 2019). To combine the proposed framework with existing STF models, we can simply replace the input of existing models with $\widetilde{\mathbf{X}}$ in Eq. 6 and A_K in Eq. 7 if necessary.

Experiment

Experimental Setup

Datasets. In the experiment, we take traffic speed prediction as an example to verify our proposed framework. We evaluate our proposed framework on four real-world widely used public datasets: *PEMS-BAY*, *METR-LA* (Li et al. 2017), *Chengdu*, and *Shenzhen* (Didi 2021). These datasets contain months of traffic flow data and the details of these data are listed in Table. 1.

Datasets	PEMS-BAY	METR-LA	Chengdu	Shenzhen
# of Nodes	325	207	524	627
# of Edges	2,694	1,722	1,120	4,845
Interval	5 min	5 min	10 min	10 min
# of Time Step	52,116	34,272	17,280	17,280
Mean	61.7768	58.2749	29.0235	31.0092
Std	9.2852	13.1280	9.6620	10.9694

Table 1: Statistical details of traffic datasets.

Few-Shot Setting. We use a similar few-shot traffic prediction setting to (Lu et al. 2022; Liu, Zheng, and Yu 2023). The data of these four cities are divided into source data, few-shot target data, and test data. Source data and few-shot target data are utilized to train the framework and test data is used to test. Here, source data consists of data from three cities, while few-shot target data and test data consist of data from the target city. For example, if *Shenzhen* is selected as the target city, the full data of *PEMS-BAY*, *METR-LA*, *Chengdu* constitutes source data. Then, three-day data of *Shenzhen* constitutes the few-shot target data, and the remaining data of *Shenzhen* constitutes the test data. We train our framework on source data and few-shot target data, and evaluate it on the test data. The same division method is used for other datasets and Z-score normalization is applied for data preprocessing.
Baselines. We consider 13 baselines belonging to three categories, including statistical methods, typical deep-learning methods, and transfer-learning methods. We use the model structure hyperparameters reported by the original papers of these models or frameworks. Note that the typical deep-learning methods are implemented in *Reptile* (Nichol, Achiam, and Schulman 2018), which is a meta-learning framework.

- **Statistical Methods:** History Average (HA) and ARIMA (Williams and Hoel 2003) calculate the statistical properties of input data to predict future signals.
- **Typical Deep-learning Methods:** DCRNN (Li et al. 2018), GWN (Wu et al. 2019), DSTAGNN (Lan et al. 2022), and FOGS (Rao et al. 2022) are classical models for spatiotemporal prediction. To apply them in the transfer learning scenario, we optimize them using the *Reptile* (Nichol, Achiam, and Schulman 2018) meta-learning framework.
- **Transfer-learning Models:** AdaRNN (Du et al. 2021), ST-GFSL (Lu et al. 2022), DASTNet (Tang et al. 2022), TPB (Liu, Zheng, and Yu 2023), TransGTR (Jin, Chen, and Yang 2023), GPD (Yuan et al. 2024b), and STGP (Hu et al. 2024) are state-of-the-art time series or spatiotemporal forecasting methods for transfer learning.

Model	Target City	METR-LA								PEMS-BAY							
	Metrics	MAE(↓)				RMSE(↓)				MAE(↓)				RMSE(↓)			
	Horizon	10 min	30 min	60 min	avg.	10 min	30 min	60 min	avg.	10 min	30 min	60 min	avg.	10 min	30 min	60 min	avg.
HA	Target Only	3.62	4.23	5.13	4.33	7.33	8.56	10.17	8.69	2.42	2.89	3.7	3.00	5.46	6.52	8.25	6.74
ARIMA		3.22	3.7	5	3.97	6.28	7.8	9.8	7.96	2.28	2.49	3.66	2.81	4.52	5.35	7.45	5.77
DCRNN	Reptile	3.01	3.65	4.67	3.78	5.62	7.16	8.96	7.25	1.83	2.43	3.36	2.54	3.36	4.71	6.59	4.89
GWN		3.11	3.75	4.73	3.86	5.87	7.31	9.1	7.43	1.99	2.45	3.14	2.53	3.55	4.64	6.23	4.81
DSTAGNN		3.3	4.1	4.95	4.12	5.9	7.73	9.56	7.73	1.85	2.51	3.59	2.65	3.41	4.79	6.66	4.95
FOGS		3.26	4.11	4.88	4.08	5.95	7.5	9.47	7.64	1.89	2.38	3.37	2.55	3.49	4.54	6.01	4.68
AdaRNN	Transfer	3.05	3.68	4.51	3.75	5.66	7.15	8.6	7.14	1.79	2.33	3.04	2.39	3.38	4.6	5.98	4.65
ST-GFSL		3	3.79	4.58	3.79	5.72	7.21	8.67	7.20	1.77	2.2	2.95	2.31	3.27	4.5	5.92	4.56
DSATNet		3.03	3.66	4.51	3.73	5.7	7.15	8.78	7.21	1.64	2.16	2.88	2.23	3.26	4.36	5.89	4.50
TPB		3.07	3.8	4.66	3.84	5.69	7.03	8.52	7.08	1.62	2.12	2.83	2.19	3.24	4.33	5.76	4.44
TransGTR		3.01	3.64	4.44	3.70	5.6	7.12	8.49	7.07	**1.6**	2.13	2.79	2.17	3.04	4.35	5.68	4.36
GPD		2.96	3.58	4.29	3.61	5.58	6.9	8.21	6.90	1.72	2.18	2.69	2.20	3.19	4.26	5.6	4.35
STGP		2.97	3.54	4.23	3.58	5.48	6.77	8.19	6.81	1.74	2.13	2.7	2.19	3.21	**4.18**	5.46	4.28
TSJT(STGCN)	Joint	2.87	3.52	4.19	3.53	5.5	6.76	8.21	6.82	1.75	2.09	2.68	2.17	3.16	4.24	5.57	4.32
TSJT(GWN)		**2.82**	**3.3**	**4.01**	**3.38**	**5.24**	**6.39**	**7.98**	**6.54**	1.62	**2.01**	**2.58**	**2.07**	**3.01**	4.2	**5.38**	**4.20**

Model	Target City	Didi-Chengdu Dataset								Didi-Shenzhen Dataset							
	Metrics	MAE(↓)				RMSE(↓)				MAE(↓)				RMSE(↓)			
	Horizon	10 min	30 min	60 min	avg.	10 min	30 min	60 min	avg.	10 min	30 min	60 min	avg.	10 min	30 min	60 min	avg.
HA	Target Only	2.69	3.13	3.65	3.16	3.69	4.57	5.27	4.51	2.17	2.66	3.04	2.62	3.33	4.05	4.63	4.00
ARIMA		2.97	3.28	4.32	3.52	3.78	4.64	5.51	4.64	2.35	2.99	3.6	2.98	4.32	4.73	5.58	4.88
DCRNN	Reptile	2.31	3.16	3.96	3.14	3.33	4.55	5.42	4.43	1.94	2.57	3.07	2.53	2.84	3.81	4.52	3.72
GWN		2.19	2.81	3.21	2.74	3.12	4.08	4.65	3.95	1.88	2.46	2.82	2.39	2.77	3.68	4.26	3.57
DSTAGNN		2.36	3.01	3.4	2.92	3.21	4.2	4.97	4.13	1.98	2.43	2.95	2.45	2.9	3.69	4.27	3.62
FOGS		2.23	2.8	3.31	2.78	3.18	4.3	4.77	4.08	1.96	2.36	2.8	2.37	2.88	3.61	4.35	3.61
AdaRNN	Transfer	2.18	2.91	3.4	2.83	3.09	3.97	4.82	3.96	1.92	2.48	2.88	2.43	2.85	3.63	4.26	3.58
ST-GFSL		2.1	2.8	3.35	2.75	3.02	3.88	4.6	3.83	1.9	2.36	2.71	2.32	2.7	3.53	4.19	3.47
DSATNet		2.06	2.7	3.03	2.60	3.02	4.01	4.53	3.85	1.86	2.34	2.64	2.28	2.73	3.51	4	3.41
TPB		2.08	2.63	3.02	2.58	2.98	3.84	4.34	3.72	1.85	2.32	2.61	2.26	2.7	3.45	3.91	3.35
TransGTR		2.05	2.65	2.8	2.50	2.95	3.82	4.26	3.68	1.89	**2.3**	2.47	2.22	2.69	3.49	3.79	3.32
GPD		2.02	2.58	2.79	2.46	2.9	3.81	4.19	3.63	1.86	2.31	2.52	2.23	2.71	3.34	3.82	3.29
STGP		1.98	2.54	2.74	2.42	2.85	3.72	4.02	3.53	1.82	2.27	**2.42**	2.17	2.66	3.39	3.69	3.25
TSJT(STGCN)	Joint	1.91	2.42	2.66	2.33	2.88	3.74	4.1	3.57	1.86	2.38	2.53	2.26	2.59	3.45	3.74	3.26
TSJT(GWN)		**1.88**	**2.39**	**2.62**	**2.30**	**2.73**	**3.61**	**3.87**	**3.40**	**1.73**	**2.3**	2.43	**2.15**	**2.51**	**3.21**	**3.48**	**3.07**

Table 2: Performance comparison of few-shot learning on four spatiotemporal datasets. Each target city only has limited data, whereas the remaining three datasets are regarded as the source domain. The best results are indicated in **bold** with light blue and the second-best results are with light gray.

Implementation: In order to fully verify the performance of our framework, we predict the traffic flow in the next 6 time steps with 12 historical time steps. Accordingly, the time step of the METR-LA and PEMS-BAY datasets is 5 minutes, while the Didi-Chengdu and Didi-Shenzhen datasets are 10 minutes due to the availability. The framework is trained on a Tesla A100 GPU with 80GB memory. The learning rates, i.e., γ_1 and γ_2 in Eq. 10, are set to 0.0002 and 0.000001 respectively, which are determined by grid search. The batch size is set to 128 and each batch contains 4 samples from the target city and 124 samples from source cities. In the experiments, we predict the spatiotemporal data in the next 6 time steps with 12 historical time steps. Two widely used metrics are applied between the multi-step prediction and the ground truth for evaluation: Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE).

Performance Comparison

The comprehensive results are presented in Table 2, from which the following insights can be gleaned:

- Our proposed framework, TSJT, equipped with the Graph WaveNet (GWN), achieves a remarkable 4.14% performance improvement over the best baseline, STGP, and consistently secures top performance across a range of datasets for both short-term and long-term predictions. This result clearly demonstrates the superiority of our approach in cross-city transfer learning scenarios.
- TSJT, when augmented with the Spatial-Temporal Graph Convolutional Network (STGCN), maintains a competitive edge in forecasting accuracy against state-of-the-art methods. Given that both STGCN and GWN are considered relatively straightforward in terms of spatiotemporal modeling complexity, this outcome further substantiates the robustness and efficacy of our framework.
- Observing that the Reptile series models are developed within the Reptile training protocol, TSJT (GWN) demonstrates an average improvement of 13.47% over the standard GWN. This enhancement highlights the superiority of our method in cross-city transfer learning scenarios.
- The performance disparity between TSJT (GWN) and TSJT (STGCN) indicates the potential of our framework to achieve even higher forecasting accuracy if paired with more sophisticated forecasting models.

Ablation Study

Component analysis To deeply analyze the effect of different components in our framework, we come up with three variants of our model, STGCN is employed in this part.

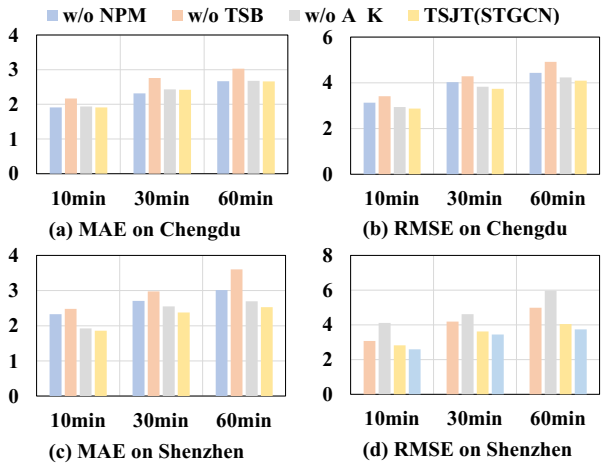


Figure 3: Impact of different components in TSJT.

- **w/o NPM:** Removing node prompting module from our framework. Directly train forecasting model with TSB.
- **w/o TSB:** Remoing target-skewed backward learning mechanism from our framework, and training forecasting model equipped with NPM in *Reptile*.
- **w/o A_K:** Removing knowledge graph in Eq.7.

The comparative analysis of TSJT variants is depicted in Fig. 3. The Target-Skewed Backward (TSB) component is revealed to be of paramount importance, as its removal from our framework results in the most significant reduction in forecasting accuracy. Similarly, the Node Prompting Module (NPM) demonstrates substantial significance; excluding it from our framework leads to an approximate 20% increase in the Mean Absolute Error (MAE) on the Shenzhen dataset. In contrast, the knowledge graph, as presented in Equation 7, exerts a negligible influence on the model's performance. These observations confirm the critical roles of the NPM and TSB in the efficacy of our proposed framework.

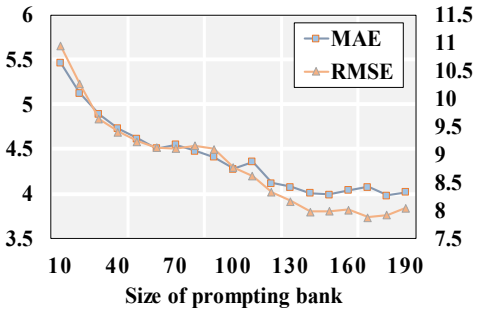
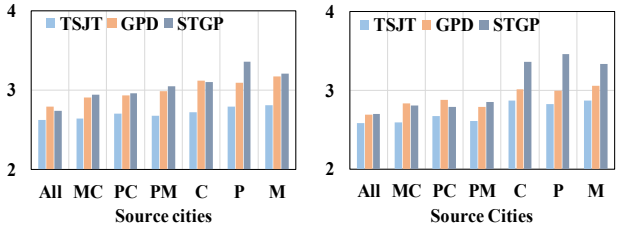


Figure 4: Impact of size of prompting bank.

Size of prompting bank The Node Prompting Module (NPM) is a critical component of our framework, with the prompting bank at its core. In this section, we assess the impact of the prompting bank's size on the framework's performance on PEMS-BAY. Specifically, we undertake a series of experiments varying the size of the prompting bank from 10 to 200, and report the MAE and RMSE. The outcomes of this investigation are illustrated in Fig. 4. Our findings indicate that when the prompting bank is relatively small, there is a significant enhancement in performance as the size increases. However, once the bank reaches a sufficient size, the rate of performance improvement diminishes, suggesting a point of diminishing returns as the bank expands further. Since a larger prompting bank leads to larger time and memory consumption, we empirically set the size of the prompting bank to half the number of nodes in our framework.



(a) MAE on Chengdu (b) MAE on METR-LA

Figure 5: Performance comparison across different sources.

Impact of source cities In the main comparison, multi source cities have been employed. We here further explore the impact of different source cities and the robustness of different methods when given diverse source cities. In this evaluation, GPD and STGP are compared, given their noteworthy performance. As shown in Fig. 5, when the number of source cities decreases, the amount of valid training data also decreases, all the three models are facing with performance drop. Taking advantage of the gradient filtering mechanism in TSB, the proposed TSJT framework has the least degradation. The result reveals the robustness of TSJT when facing with few available source cities.

Conclusion

This paper introduces a one-stage cross-city transfer learning framework to address the challenge of data scarcity. We present the Target-Skewed Joint Training (TSJT) framework, an elegant end-to-end solution for knowledge transfer from source to target cities. At the heart of TSJT is the innovative Target-Skewed Backward (TSB) training strategy, which selectively refines gradients from source city data and retains only components beneficial to the target city. The Target-Skewed Backward training strategy presented in this paper is not only tailored for the specific task of cross-city spatiotemporal forecasting but also possesses the potential to be readily applied to a variety of other transfer learning tasks and scenarios.

Acknowledgments

This paper is partially supported by the National Natural Science Foundation of China (No.12227901, No.62072427), the Project of Stable Support for Youth Team in Basic Research Field, CAS (No.YSBR-005), Academic Leaders Cultivation Program, USTC.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Chen, J.; Liu, T.; and Li, R. 2023. Region Profile Enhanced Urban Spatio-Temporal Prediction via Adaptive Meta-Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 224–233. Birmingham United Kingdom: ACM. ISBN 9798400701245.

Chen, K.; Han, T.; Gong, J.; Bai, L.; Ling, F.; Luo, J.-J.; Chen, X.; Ma, L.; Zhang, T.; Su, R.; et al. 2023. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv preprint arXiv:2304.02948*.

Didi, G. i. 2021. Didi Chuxing data. https://gaia.didichuxing.com.

Du, Y.; Wang, J.; Feng, W.; Pan, S.; Qin, T.; Xu, R.; and Wang, C. 2021. Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 402–411.

Hu, J.; Liu, X.; Fan, Z.; Yin, Y.; Xiang, S.; Ramasamy, S.; and Zimmermann, R. 2024. Prompt-Enhanced Spatio-Temporal Graph Transfer Learning. *arXiv preprint arXiv:2405.12452*.

Jin, Y.; Chen, K.; and Yang, Q. 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.

Jin, Y.; Chen, K.; and Yang, Q. 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*, 1032–1043.

Lan, S.; Ma, Y.; Huang, W.; Wang, W.; Yang, H.; and Li, P. 2022. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *International conference on machine learning*, 11906–11917. PMLR.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

Liu, Z.; Zheng, G.; and Yu, Y. 2023. Cross-city few-shot traffic forecasting via traffic pattern bank. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 1451–1460.

Lu, B.; Gan, X.; Zhang, W.; Yao, H.; Fu, L.; and Wang, X. 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*, 1162–1172.

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Ouyang, X.; Yang, Y.; Zhou, W.; Zhang, Y.; Wang, H.; and Huang, W. 2024. CityTrans: Domain-Adversarial Training With Knowledge Transfer for Spatio-Temporal Prediction Across Cities. *IEEE Transactions on Knowledge and Data Engineering*, 36(1): 62–76. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Rao, X.; Wang, H.; Zhang, L.; Li, J.; Shang, S.; and Han, P. 2022. FOGS: First-Order Gradient Supervision with Learning-based Graph for Traffic Flow Forecasting. In *31st International Joint Conference on Artificial Intelligence, IJCAI 2022*, 3926–3932. International Joint Conferences on Artificial Intelligence.

Tang, Y.; Qu, A.; Chow, A. H.; Lam, W. H.; Wong, S. C.; and Ma, W. 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*, 1905–1915.

Vaswani, A. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, X.; Wang, P.; Wang, B.; Zhang, Y.; Zhou, Z.; Bai, L.; and Wang, Y. 2023. Latent Gaussian Processes based Graph Learning for Urban Traffic Prediction. *IEEE Transactions on Vehicular Technology*.

Wei, Y.; Zheng, Y.; and Yang, Q. 2016. Transfer knowledge between cities. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1905–1914.

Williams, B. M.; and Hoel, L. A. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6): 664–672.

Wu, H.; Zhou, H.; Long, M.; and Wang, J. 2023. Interpretable weather forecasting for worldwide stations with a unified deep model. *Nature Machine Intelligence*, 5(6): 602–611.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 1907–1913.

Yao, H.; Liu, Y.; Wei, Y.; Tang, X.; and Li, Z. 2019. Learning from Multiple Cities: A Meta-Learning Approach for Spatial-Temporal Prediction. In *The World Wide Web Conference*, 2181–2191. San Francisco CA USA: ACM. ISBN 978-1-4503-6674-8.

Yu, B.; Yin, H.; and Zhu, Z. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Yuan, Y.; Ding, J.; Feng, J.; Jin, D.; and Li, Y. 2024a. UniST: A Prompt-Empowered Universal Model for Urban Spatio-Temporal Prediction. ArXiv:2402.11838 [cs].

Yuan, Y.; Shao, C.; Ding, J.; Jin, D.; and Li, Y. 2024b. Spatio-Temporal Few-Shot Learning via Diffusive Neural Network Generation. In *The Twelfth International Conference on Learning Representations*.

Zhang, Y.; Wang, P.; Wang, B.; Wang, X.; Zhao, Z.; Zhou, Z.; Bai, L.; and Wang, Y. 2024. Adaptive and Interactive

Multi-Level Spatio-Temporal Network for Traffic Forecasting. *IEEE Transactions on Intelligent Transportation Systems*.