Selective Cross-City Transfer Learning for Traffic Prediction via Source City Region Re-Weighting

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

Deep learning models have been demonstrated powerful in modeling complex spatio-temporal data for traffic prediction. In practice, effective deep traffic prediction models rely on large-scale traffic data, which is not always available in real-world scenarios. To alleviate the data scarcity issue, a promising way is to use cross-city transfer learning methods to fine-tune well-trained models from source cities with abundant data. However, existing approaches overlook the divergence between source and target cities, and thus, the trained model from source cities may contain noise or even harmful source knowledge. To address the problem, we propose CrossTReS, a selective transfer learning framework for traffic prediction that adaptively re-weights source regions to assist target fine-tuning. As a general framework for fine-tuning-based cross-city transfer learning, CrossTReS consists of a feature network, a weighting network, and a prediction model. We train the feature network with node- and edge-level domain adaptation techniques to learn generalizable spatial features for both source and target cities. We further train the weighting network via source-target joint meta-learning such that source regions helpful to target fine-tuning are assigned high weights. Finally, the prediction model is selectively trained on the source city with the learned weights to initialize target fine-tuning. We evaluate CrossTReS using real-world taxi and bike data, where under the same settings, CrossTReS outperforms state-of-the-art baselines by up to 8%. Moreover, the learned region weights offer interpretable visualization.

CCS CONCEPTS

• **Computing methodologies → Transfer learning**; • **Applied computing → Transportation**; **Forecasting**.

KEYWORDS

Urban Computing; Traffic Prediction; Selective Transfer Learning

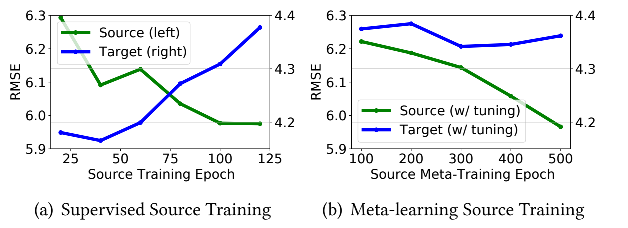
ACM Reference Format:
Yilun Jin, Kai Chen, and Qiang Yang. 2022. Selective Cross-City Transfer Learning for Traffic Prediction via Source City Region Re-Weighting. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3534678.3539250

1 INTRODUCTION

Traffic prediction is essential in various smart city applications. Timely and accurate traffic prediction is helpful not only to urban practitioners for policymaking and resource allocation, but also to urban residents for trip planning. Recently, deep learning models such as Convolutional Neural Networks (CNN) and Graph Neural Networks (GNN) achieve great success in traffic prediction tasks, such as traffic speeds [13, 27], travel demands [9, 25], and crowd flows [29]. However, deep learning models require large-scale data, which is not always accessible for urban traffic prediction tasks. For example, cities with newly deployed ride-sharing services have only several days of data, which is insufficient to train deep learning models and may lead to low-quality services. Therefore, improving traffic prediction under a lack of data is of great importance.

To address the problem, *transfer learning* [18] methods are proposed to transfer knowledge from a city with much traffic data to another city with limited data. Existing transfer learning methods for traffic prediction are generally based on *fine-tuning* [19, 20, 24]. They train models using supervised learning [19] or meta-learning [24] on abundant source data as initialization and design methods to fine-tune the models on target data. However, a common pitfall of existing works is that they ignore the gap between source training and target fine-tuning. Thus, the model learned on abundant source data may contain noise or even harmful source knowledge to the target city, which negatively impacts transfer learning performance.

We perform experiments on real-world taxi data from Chicago and Washington DC to illustrate the weakness of fine-tuning-based solutions. We vary the number of source training epochs and plot the corresponding test error on both source (Chicago) and target (DC, after fine-tuning) cities in Fig. 1. As shown, for both supervised learning and meta-learning, as the number of source training epochs increases, the error on the source city decreases, while the error on the target city remains similar (Fig. 1(b)) or even increases (Fig. 1(a)). The results suggest that knowledge learned from the source city may not help or even harm the target city, and that it is important to perform *selective transfer learning*, i.e. selecting relevant source knowledge to transfer to the target city. As cities are often divided



(a) Supervised Source Training (b) Meta-learning Source Training

Figure 1: Error on source (Chicago) and target (Washington DC, after fine-tuning) cities with various source training epochs. For source training with meta-learning, we report source error after fine-tuning.

into regions in urban computing [19], we focus on selecting helpful source *regions* to the target city. By doing so, we improve not only traffic prediction in the target city, but also the interpretability of cross-city transfer learning, as the selected regions provide intuitive understandings of the common knowledge between cities.

The problem of selective transfer learning has been studied in areas like Natural Language Processing (NLP) [4, 14, 15] where the models select the most indicative words or documents. However, selective cross-city transfer learning for traffic prediction faces unique challenges. The major difference is that sample selection in NLP focuses on tasks in *the same language*, thus sharing *the same set of words*. However, as cities are geographically disjoint, they do not share common regions, which leads to two challenges.

- **Extracting generalizable region features.** Generalizable region features between both cities must be extracted before selective transfer learning is performed. While word embeddings or language models act as generalizable features across NLP tasks, existing methods that learn region features [7, 30, 31] focus on region-wise relations in a single city, and are thus *city-specific* and may not generalize to another city.
- **Evaluating helpfulness of source regions.** Through the shared words between NLP tasks, we can evaluate words in both source and target contexts and thus select indicative words or samples across tasks. However, as no regions exist in both source and target cities, it is challenging to evaluate the helpfulness of a source region in the target context.

In this paper, we propose CrossTReS (**Cross**-City **T**ransfer via **Re**gion **S**election), a selective cross-city transfer learning framework for traffic prediction by adaptively re-weighting source regions. CrossTReS applies to general fine-tuning-based cross-city transfer learning methods. Specifically, CrossTReS consists of three components, a feature network F_{θ_f}, a weighting network F_{θ_w}, and a prediction model F_θ. We train the feature network with node- and edge-level domain adaptation techniques to learn generalizable spatial features for both source and target regions. Based on the learned features, the weighting model assigns weights λ_{r_S} to source regions and is trained via a source-target joint meta-learning. We simulate source training and target fine-tuning in the inner loop of the joint meta-learning, and learn θ_f, θ_w such that source regions that help target fine-tuning are assigned high weights. Finally, we selectively train the prediction model F_θ on the source city via λ_{r_S} to initialize target fine-tuning. Extensive experiments on real-world taxi and bike data are performed, where under the same settings, CrossTReS outperforms state-of-the-art baselines by as much as 8%. Moreover, the learned region weights offer interpretable visualization results.

To summarize, we make the following contributions.

- To our knowledge, this is the first work to study the selective cross-city transfer learning problem for traffic prediction.
- We propose CrossTReS, a selective cross-city transfer learning framework for traffic prediction. With node- and edge-level domain adaptation methods and joint meta-learning, CrossTReS extracts city-agnostic region features and adaptively re-weights source regions to improve target fine-tuning.
- We perform extensive experiments and case studies on real-world taxi and bike data to validate the effectiveness and interpretability of CrossTReS.

2 RELATED WORKS

2.1 Traffic Prediction

Traffic prediction, such as predicting urban flows [29], traffic speeds [13], and travel demands [9, 26] plays important roles in smart transportation systems. With the advances in deep learning, deep models are used for traffic prediction, such as CNN [29], recurrent networks (RNN) [26], and GNN [9, 13]. These models extract complex spatio-temporal relations and can perform accurate traffic prediction. However, deep models suffer from accuracy degradation upon a lack of data, which is the problem we tackle in this paper.

2.2 Transfer Learning for Traffic Prediction

The data insufficiency problem is common in traffic prediction. For example, cities without advanced digital infrastructures (e.g. speed sensors) would have difficulty gathering data. To mitigate this problem, transfer learning methods are proposed for deep traffic prediction models, including RegionTrans [19], MetaST [24], and ST-DAAN [20]. Given a well-trained model on the source city, RegionTrans uses auxiliary data similarity to regularize fine-tuning, MetaST extracts and transfers long-term temporal features, while ST-DAAN leverages deep adaptation networks (DAN) [17] for fine-tuning. However, all these works focus on how to fine-tune a source model but ignore how to train a generalizable source model for fine-tuning. Different from existing works, this paper aims to selectively learn from the source city to accommodate target fine-tuning.

We discuss transfer learning for other urban computing tasks besides traffic prediction in Section B in the Appendix.

2.3 Selective Transfer Learning

Selecting relevant knowledge for transfer learning is an important problem that has been studied in computer vision (CV) [2, 28], NLP [4, 14, 15], and recommendation [16]. However, selective transfer learning in these areas is assisted with 1) generalizable feature extractors, such as pre-trained CNN and word embeddings, and 2) explicit cross-domain links, such as shared vocabulary [14, 15], label sets [2, 28], or item sets [16]. On the contrary, in cities, existing works only extract city-specific features that may not be generalizable between source and target cities. Further, as no regions exist in both source and target cities, it is challenging to evaluate whether a source region will help target fine-tuning.

In a similar field, multi-task learning, researchers propose task selection methods [3, 5, 11] by aligning the gradients of each task. However, in cities with hundreds of regions, it is not affordable to compute gradients for all regions.

2.4 Regional Feature Learning

Regional feature learning, also known as *urban region embeddings*, aims to learn versatile region features that apply to various tasks, such as land usage and crime rate prediction [7, 31]. Recently, multi-view urban data, such as human mobility and POI data are incorporated into region embeddings [30]. However, they all focus on learning embeddings for a single city, while we aim to learn generalizable region features between two cities.

3 BACKGROUND & MOTIVATION

In this section, we first introduce necessary backgrounds, including notations and existing fine-tuning-based solutions. Then, we analyze the drawback of existing works and formulate the problem of selective source training for cross-city transfer learning.

3.1 Preliminaries

3.1.1 Notations. Notations and definitions are presented below.

Definition 1 (Regions). *A city C is divided into a grid map with $W_C \times H_C$ grids, where W_C, H_C denote the longitudinal and latitudinal ranges. Each grid in the map is referred to as a region r. With an abuse of notation, we also use C to denote the set of regions in city C.*

Definition 2 (Time Intervals). *We divide the time range in city C into disjoint time intervals of equal length $t = 1, \dots T_C$.*

Definition 3 (Regional Traffic Data). *Each region r in C is associated with a vector $\mathbf{x}_r = \left[x_r^{(t)}\right]_{t=1}^{T_C}$ describing the traffic data (e.g. number of taxi pickups) of r from time interval 1 to T_C.*

We then present the definition of cross-city transfer learning for traffic prediction [19]. In this paper, we mainly focus on grid-based prediction tasks [12] and leave graph-based tasks for future works.

Definition 4 (Transfer Learning for Traffic Prediction). *Given a source city $\mathcal{S}$ and a target city $\mathcal{T}$ with traffic data $\mathcal{X}_\mathcal{S} = \{\mathbf{x}_{r_\mathcal{S}}, r_\mathcal{S} \in \mathcal{S}\}, \mathcal{X}_\mathcal{T} = \{\mathbf{x}_{r_\mathcal{T}}, r_\mathcal{T} \in \mathcal{T}\}, T_\mathcal{S} \gg T_\mathcal{T}$, we aim to learn a prediction model F_θ using target data and abundant source data.*

$$\begin{aligned} &\min_\theta \mathcal{L}_\mathcal{T}(\mathcal{X}_\mathcal{T}, \theta) = \sum_{r \in \mathcal{T}} \sum_{t=1}^{T_\mathcal{T}} L\left(\hat{x}_r^{(t)}, x_r^{(t)}\right), \\ &\text{where } \hat{x}_r^{(t)} = F_\theta\left(\left[x_r^{(t-\tau)}, \dots x_r^{(t-1)}\right], \mathcal{X}_\mathcal{S}\right). \end{aligned} \tag{1}$$

$\hat{x}_r^{(t)}, x_r^{(t)}$ are the predicted and true values of target region r at time interval t. L is an error function such as squared or absolute error.

3.1.2 Fine-Tuning for Cross-City Transfer Learning. Existing works solve the problem of cross-city transfer learning for traffic prediction via fine-tuning [19, 20, 24], which consists of two steps.

- **Source Training.** The model θ is trained with abundant source data $\mathcal{X}_\mathcal{S}$ to capture knowledge from the source city. There are two ways to perform source training.
  - **Supervised Learning [19, 20],** where the error on the source city $\mathcal{L}_\mathcal{S}$ is directly minimized
$$\theta_\mathcal{S} = \arg\min_\theta \mathcal{L}_\mathcal{S}(\mathcal{X}_\mathcal{S}, \theta). \tag{2}$$
  - **Meta-Learning [24],** where the source error after a gradient step on source data is minimized
$$\theta_\mathcal{S} = \arg\min_\theta \mathcal{L}_\mathcal{S}(\mathcal{X}_\mathcal{S}, \theta - \alpha\nabla_\theta \mathcal{L}_\mathcal{S}). \tag{3}$$
- **Fine-Tuning.** The model θ is trained on limited target data $\mathcal{X}_\mathcal{T}$ starting from the trained source model $\theta_\mathcal{S}$, such that knowledge encoded in $\theta_\mathcal{S}$ is transferred to the target city,
$$\theta_\mathcal{T} = \arg\min_\theta \mathcal{L}_\mathcal{T}(\mathcal{X}_\mathcal{T}, \theta; \theta_\mathcal{S}), \tag{4}$$
where $\mathcal{L}_\mathcal{T}(\mathcal{X}_\mathcal{T}, \theta; \theta_\mathcal{S})$ denotes Eqn. 1 initialized with $\theta_\mathcal{S}$.

3.2 Motivation and Problem Definition

As shown in Eqn. 2, 3 and 4, there is an inherent gap between source training and target fine-tuning objectives. While the overall goal of cross-city transfer learning is to minimize the error on target data $\mathcal{L}_\mathcal{T}$, source training ignores $\mathcal{L}_\mathcal{T}$ and minimizes the error only on source data. Therefore, source training methods, including both supervised and meta-learning used by existing works may learn noise or harmful source knowledge, which lead to sub-optimal performances on target fine-tuning, as shown in Fig. 1.

To bridge the gap, we aim to perform selective cross-city transfer learning to rule out harmful source knowledge. For fine-tuning-based methods, source knowledge is learned during source training rather than fine-tuning. Thus, we perform selective transfer during source training. Moreover, as cities are often divided into regions [19], we perform selection at the level of regions. Concretely, we formulate the problem of selective source training as follows.

Definition 5 (Selective Source Training for Cross-city Transfer Learning). *The goal of selective source training is to learn weights $\lambda_{r_\mathcal{S}} > 0$ for source region $r_\mathcal{S}$ to perform source training, such that after minimizing the weighted error on the source city,*

$$\theta_\mathcal{S} = \arg\min_\theta \hat{\mathcal{L}}_\mathcal{S}(\mathcal{X}_\mathcal{S}, \theta; \lambda_{r_\mathcal{S}}) = \sum_{r_\mathcal{S} \in \mathcal{S}} \sum_{t=1}^{T_\mathcal{S}} \lambda_{r_\mathcal{S}} L\left(\hat{x}_{r_\mathcal{S}}^{(t)}, x_{r_\mathcal{S}}^{(t)}\right) \tag{5}$$

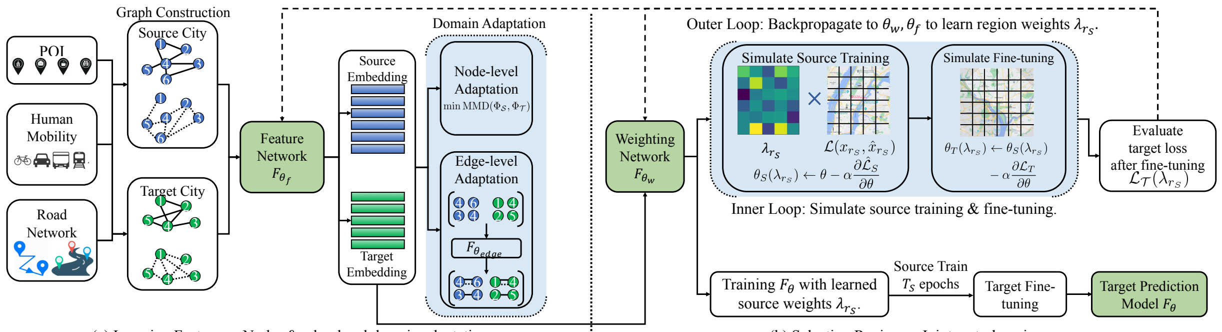
and after fine-tuning (Eqn. 4), error on the target city $\mathcal{L}_\mathcal{T}$ is minimized.

Learning region weights $\lambda_{r_\mathcal{S}}$ essentially performs knowledge selection by assigning low $\lambda_{r_\mathcal{S}}$ to regions with harmful knowledge, and vice versa, which leads to two advantages. On one hand, we improve traffic prediction in the target city. On the other hand, we enhance the interpretability of cross-city transfer learning, as the learned $\lambda_{r_\mathcal{S}}$ sheds light on the shared knowledge between cities.

We note that the selective source training problem is general and agnostic to specific fine-tuning methods. Thus, existing fine-tuning methods such as RegionTrans and MetaST [19, 20, 24] are orthogonal to the contribution of this paper.

4 PROPOSED METHOD

In this section, we introduce our framework, CrossTReS. We first present an overview, and then introduce detailed methods.


Graph Construction
POI
Human Mobility
Road Network
Source City
Target City
Feature Network F_{θ_f}
Source Embedding
Target Embedding
Domain Adaptation
Node-level Adaptation
min MMD($\Phi_\mathcal{S}, \Phi_\mathcal{T}$)
Edge-level Adaptation
$F_{\theta_{edge}}$
Weighting Network F_{θ_w}
Outer Loop: Backpropagate to θ_w, θ_f to learn region weights $\lambda_{r_\mathcal{S}}$.
Simulate Source Training
$\lambda_{r_\mathcal{S}}$
$\mathcal{L}(x_{r_\mathcal{S}}, \hat{x}_{r_\mathcal{S}})$
$\theta_\mathcal{S}(\lambda_{r_\mathcal{S}}) \leftarrow \theta - \alpha \frac{\partial \mathcal{L}_\mathcal{S}}{\partial \theta}$
Simulate Fine-tuning
$\theta_\mathcal{T}(\lambda_{r_\mathcal{S}}) \leftarrow \theta_\mathcal{S}(\lambda_{r_\mathcal{S}}) - \alpha \frac{\partial \mathcal{L}_\mathcal{T}}{\partial \theta}$
Evaluate target loss after fine-tuning $\mathcal{L}_\mathcal{T}(\lambda_{r_\mathcal{S}})$
Inner Loop: Simulate source training & fine-tuning.
Training F_θ with learned source weights $\lambda_{r_\mathcal{S}}$.
Source Train T_S epochs
Target Fine-tuning
Target Prediction Model F_θ

(a) Learning Features – Node- & edge-level domain adaptation (b) Selecting Regions – Joint meta-learning

Figure 2: Overview of CrossTReS. CrossTReS consists of a feature network F_{θ_f}, a weighting network F_{θ_w}, and a prediction model F_θ. The feature network captures shared spatial features between cities to uncover common urban functions. The weighting network learns to assign high weights to source regions that improve target fine-tuning. Finally, the prediction model F_θ is selectively trained with $\lambda_{r_\mathcal{S}}$ to initialize target fine-tuning.

4.1 Overview of CrossTReS

The key challenges to selective source training are two-fold. First, we need to extract city-agnostic region features that are indicative in both source and target cities. Second, based on the extracted features, we need to evaluate the helpfulness of source regions to target city fine-tuning. To achieve both goals, CrossTReS contains the following components, as illustrated in Fig. 2.

- **Feature Network F_{θ_f}.** In urban planning, each region bears certain urban functions that are related to traffic patterns. For example, traffic flows in industrial areas peak during weekdays, while those in business centers peak during weekends. Capturing generalizable spatial features uncovers shared urban functions between cities and facilitates knowledge transfer. To capture such features, we train the feature network via both node- and edge-level domain adaptation objectives.
- **Weighting Network F_{θ_w}.** F_{θ_w} learns to adaptively re-weight source regions. For example, if the target city enjoys smooth traffic flows, source regions with heavy congestion would be detrimental and should be down-weighted. To achieve this goal, we train F_{θ_f} and F_{θ_w} via source-target joint meta-learning. By simulating source training and target fine-tuning in the inner loop, both networks are learned to re-weight source regions to improve target fine-tuning.
- **Prediction Model F_θ.** F_θ is selectively trained on the source city with weights $\lambda_{r_\mathcal{S}}$ to initialize target fine-tuning.

4.2 Model Components

In this section, we introduce detailed components of CrossTReS.

4.2.1 *Feature Network.*

The feature network extracts generalizable spatial features for source and target regions upon which we identify shared urban functions and perform selective source training. Following existing works on region embeddings [7, 30], we model regions as nodes and a city as graphs based on multi-view urban data and feed them as inputs to the feature network. We build graphs based on the following region-wise relations: proximity $\mathcal{G}^C_{prox}$, road connections $\mathcal{G}^C_{road}$, POI $\mathcal{G}^C_{poi}$, and human mobility $\mathcal{G}^C_s, \mathcal{G}^C_d$. How to build the graphs is beyond the scope of this paper. We refer readers to Section D in the Appendix for details.

With the multi-view graphs $\left\{\mathcal{G}^C_v = \left(C, \mathcal{E}^C_v\right)\right\}, v \in \{prox, road, poi, s, d\}, C \in \{\mathcal{S}, \mathcal{T}\}$ where $\mathcal{E}^C_v$ denotes the edge set of $\mathcal{G}^C_v$, we leverage the feature network F_{θ_f} to project $\{\mathcal{G}^C_v\}$ into regional spatial features $\Phi_C \in \mathbb{R}^{|C| \times d_{emb}}, C \in \{\mathcal{S}, \mathcal{T}\}$, where d_{emb} is the dimension of output region features. Note that we do not specify the architecture of F_{θ_f} as long as it operates on graph data. Our detailed implementations are specified in Section 5.1.

4.2.2 *Weighting Network.*

The weighting network F_{θ_w} transforms the regional spatial features $\Phi_\mathcal{S}, \Phi_\mathcal{T}$ to source region weights $\lambda_{r_\mathcal{S}} \geq 0$. The intuition of the weighting network is that source and target regions with similar urban functions are likely to share common temporal features. Thus, we first transform both $\Phi_\mathcal{S}, \Phi_\mathcal{T}$ with F_{θ_w},

$$\hat{\Phi}_{r_\mathcal{S}} = F_{\theta_w}(\Phi_{r_\mathcal{S}}), \hat{\Phi}_{r_\mathcal{T}} = F_{\theta_w}(\Phi_{r_\mathcal{T}}), \tag{6}$$

and then apply mean pooling to all target regions to obtain the target city features $\hat{\Phi}_\mathcal{T}$. We finally obtain source region weights $\lambda_{r_\mathcal{S}}$ via inner product and a tanh activation,

$$\lambda_{r_\mathcal{S}} = \max\left(\tanh\left(\hat{\Phi}^\intercal_{r_\mathcal{S}} \hat{\Phi}_\mathcal{T}\right), 0\right), \tag{7}$$

where Φ_{r_C} denotes the features of region $r_C \in C, C \in \{\mathcal{S}, \mathcal{T}\}$.

4.2.3 *Prediction Model.*

The prediction model F_θ takes source and target data $\mathcal{X}_\mathcal{S}, \mathcal{X}_\mathcal{T}$ and the learned $\lambda_{r_\mathcal{S}}$ as inputs. It performs source training according to Eqn. 5 with weights $\lambda_{r_\mathcal{S}}$, which will be fine-tuned with target data $\mathcal{X}_\mathcal{T}$. We do not make assumptions on F_θ.

4.3 Learning to Re-Weight Source Regions

In this section, we introduce how to learn $\theta_f, \theta_w, \theta$ to address the two key challenges in Sec. 4.1, i.e. extracting city-agnostic region features and evaluating the helpfulness of source regions, and thus solve the problem of selective source training.

4.3.1 *Learning Region Features via Bi-level Domain Adaptation.*

Existing works on regional feature learning leverage intra-city region-wise links, e.g. $\mathcal{G}^C_{road}, \mathcal{G}^C_{poi}$. Consequently, the learned features Φ_C are indicative of urban functions in the same city C rather than between cities. Thus, learning $\Phi_{\mathcal{S}}, \Phi_{\mathcal{T}}$ that encodes city-agnostic region features is a crucial challenge before we select *source* regions to assist *target* fine-tuning. We design node- and edge-level domain adaptation methods to meet the challenge.

Node-Level Adaptation. Following existing works on domain adaptation [17, 23], we minimize the maximum mean discrepancy (MMD) [1] between regional features from both cities

$$\mathcal{L}_{node} = \mathrm{MMD}(\Phi_{\mathcal{S}}, \Phi_{\mathcal{T}}). \tag{8}$$

Intuitively, $\mathcal{L}_{node}$ aligns the distributions of node (i.e. region) features from both cities.

Edge-Level Adaptation. Existing works on domain adaptation focus on aligning feature distributions of individual samples. However, as we model a city with multi-view graphs, nodes (i.e. regions) in the graphs are connected with multiple types of edges. As each type of edge carries unique semantics (e.g. road connections, similar origins/destinations), we additionally propose to perform edge-level domain adaptation to align the distributions of edge features. The intuition of the edge-level adaptation is two-fold:

- **Distinguishable w.r.t. Edge Types:** Features of different types of edges should be clearly separated.
- **Indistinguishable w.r.t. Cities:** Features of the same type of edges should be similar regardless of the city.

We implement the intuition via a *shared edge classifier.* The edge classifier takes edge features as inputs, outputs potential edge types, and is shared between both cities. By learning to classify edge types, we ensure the distinguishability w.r.t. edge types. By sharing the classifier across cities, we ensure the indistinguishability w.r.t. cities, as edge features from both cities are classified via the same classifier and thus follow the same distributions. Thus, we do not need an extra domain discriminator [8].

Formally, given a region pair $r_{C,i}, r_{C,j} \in C, C \in \{\mathcal{S}, \mathcal{T}\}$, we obtain the edge features by concatenating their node features as $\Phi_{C,ij} = \left[\Phi_{C,i} \| \Phi_{C,j}\right]$, and assign edge labels $l_{C,ij} \in \{0,1\}^5$ to indicate the types of edges between the region pair,

$$l_{C,ij} = \left[\mathbb{I}\left(\left(r_{C,i}, r_{C,j}\right) \in \mathcal{E}_v^C\right)\right]_{v \in \{prox,road,poi,s,d\}}, \tag{9}$$

where $\mathbb{I}$ is the indicator function. For example, if $(r_{C,i}, r_{C,j})$ is connected in $\mathcal{G}_{poi}^C, \mathcal{G}_s^C$, $l_{C,ij} = [0,0,1,1,0]$. We use the edge classifier $F_{\theta_{edge}}$ to predict edge labels $l_{C,ij}$ given the edge features $\Phi_{C,ij}$. We learn the edge classifier by minimizing the following loss function,

$$\mathcal{L}_{edge} = \sum_{C \in \{\mathcal{S},\mathcal{T}\}} \sum_{r_{C,i}, r_{C,j} \in C} \mathrm{BCE}\left(F_{\theta_{edge}}(\Phi_{C,ij}), l_{C,ij}\right), \tag{10}$$

where BCE is the multi-label binary cross entropy loss

$$\mathrm{BCE}(\hat{y}, y) = -\sum_{i=1}^{L} \left[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right].$$

By minimizing $\mathcal{L}_{edge}$, we ensure that features of each type of edges lie in their feature spaces defined by $F_{\theta_{edge}}$. It is worth noting that the edge classifier does not require adversarial learning [8], as both F_{θ_f} and $F_{\theta_{edge}}$ minimize $\mathcal{L}_{edge}$ to learn indicative edge features and to better separate different edge types in both cities.

Finally, to learn generalizable region features between source and target cities, we share the edge classifier between both cities as a domain adaptation regularizer, and minimize the following loss,

$$\min_{\theta_f, \theta_{edge}} \mathcal{L}_f = \mathcal{L}_{emb} + \beta_1 \mathcal{L}_{node} + \beta_2 \mathcal{L}_{edge}, \tag{11}$$

where β_1, β_2 are hyperparameters for node and edge-level adaptations respectively, and $\mathcal{L}_{emb}$ is the loss of the feature network.

4.3.2 *Learning Region Weights via Joint Meta-Learning.*

With the generalizable spatial features as indicators, we now introduce how to adaptively re-weight source regions to improve target fine-tuning.

Intuitively, we should learn $\lambda_{r_{\mathcal{S}}}$ such that after optimizing the weighted source loss $\hat{\mathcal{L}}_{\mathcal{S}}$ in Eqn. 5, the initialization $\theta_{\mathcal{S}}$ leads to low target error after fine-tuning. To implement the intuition, we design a joint meta-learning approach with the following steps.

(1) *Simulating Source Training.* We optimize Eqn. 5 to simulate selective source training with current weights λ_{r_s}. We perform $K_{\mathcal{S}}$ steps of stochastic gradient updates on source data,

$$\theta_{\mathcal{S}}(\lambda_{r_{\mathcal{S}}}) = \theta - \alpha \frac{\partial \hat{\mathcal{L}}_{\mathcal{S}}(\mathcal{X}_{\mathcal{S}}, \theta; \lambda_{r_{\mathcal{S}}})}{\partial \theta}. \tag{12}$$

(2) *Simulating Fine-Tuning.* Starting from $\theta_{\mathcal{S}}(\lambda_{r_{\mathcal{S}}})$, we optimize Eqn. 4 to simulate the fine-tuning stage. We perform $K_{\mathcal{T}}$ steps of stochastic gradient updates on target data

$$\theta_{\mathcal{T}}(\lambda_{r_{\mathcal{S}}}) = \theta_{\mathcal{S}}(\lambda_{r_{\mathcal{S}}}) - \alpha \frac{\partial \mathcal{L}_{\mathcal{T}}(\mathcal{X}_{\mathcal{T}}, \theta; \theta_{\mathcal{S}}(\lambda_{r_{\mathcal{S}}}))}{\partial \theta}. \tag{13}$$

(3) *Optimizing Weights.* We sample a batch of target data $\tilde{\mathcal{X}}_{\mathcal{T}}$ to evaluate the target loss $\tilde{\mathcal{L}}_{\mathcal{T}}(\lambda_{r_{\mathcal{S}}}) = \mathcal{L}_{\mathcal{T}}(\tilde{\mathcal{X}}_{\mathcal{T}}, \theta_{\mathcal{T}}(\lambda_{r_{\mathcal{S}}}))$ with parameters $\theta_{\mathcal{T}}(\lambda_{r_{\mathcal{S}}})$ after simulation. We then optimize both the weighting and the feature networks by taking a gradient step on $\tilde{\mathcal{L}}_{\mathcal{T}}(\lambda_{r_{\mathcal{S}}})$ w.r.t. $\lambda_{r_{\mathcal{S}}}$. Denoting $\theta_s = \{\theta_f, \theta_w\}$ as all parameters contributing to $\lambda_{r_{\mathcal{S}}}$, we have

$$\theta_s = \theta_s - \gamma \frac{\partial \tilde{\mathcal{L}}_{\mathcal{T}}(\lambda_{r_{\mathcal{S}}})}{\partial \lambda_{r_{\mathcal{S}}}} \cdot \frac{\partial \lambda_{r_{\mathcal{S}}}}{\partial \theta_s}. \tag{14}$$

In Eqn. 12, 13, and 14, α, γ denote learning rates. Taking Step 1 and 2 as inner loops and Step 3 as the outer loop, our approach resembles model agnostic meta-learning (MAML) in MetaST ([6, 24], Eqn. 3) with the following differences[1].

(1) The *inner loop* of MAML optimizes on source data, while our approach first optimizes on source data, then target data in the inner loop. This indicates that MAML optimizes the error after source training, while our approach optimizes the error after both source training and target fine-tuning.
(2) The *outer loop* of MAML minimizes the error on source data, while our approach minimizes on target data. This indicates that MAML leads to fast adaptation on the source domain instead of the target (shown in Fig. 1), while our approach leads to fast adaptation to target data after fine-tuning.
(3) MAML optimizes *the prediction model* θ at the outer loop, while our approach optimizes the *weights* $\lambda_{r_{\mathcal{S}}}$. This suggests that MAML fails to select important source knowledge and may lead to sub-optimal adaptation, while our approach adaptively re-weights relevant source regions to select helpful knowledge from noise.

[1] We assume that we only have one source domain (city).

4.3.3 Training Process. The overall training process of CrossTReS follows the two steps, source training and fine-tuning, described in Section 3.1.2. In each source training epoch, we optimize θ_f and θ_w to obtain the current region weights λ_{r_S}, which are used to selectively train the prediction model θ on source data for an epoch. The weights λ_{r_S} are updated along the training process.

After selective source training with CrossTReS, target fine-tuning is performed. CrossTReS is compatible with general fine-tuning methods, such as RegionTrans [19] and STMem [24].

The pseudocode of CrossTReS is shown in Alg. 1 in the Appendix.

5 EXPERIMENTS

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

- How does CrossTReS compare against state-of-the-art cross-city transfer learning methods for traffic prediction?
- How do the model components and hyperparameters impact the overall performance of CrossTReS?
- How can the learned region weights λ_{r_S} be interpreted?

5.1 Experimental Setup

Following [19, 24], we perform experiments on taxi and bike datasets. For each trip, a vehicle picks up a user from a start region and drops the user off in a destination region. Our tasks are to predict future pickup and dropoff volumes in each region. We use root mean squared error (RMSE) and mean absolute error (MAE) to evaluate performances [20]. We report the mean error on pickup and dropoff.

Datasets. We collect taxi and bike data from New York (NY), Chicago (CHI), and Washington (DC). The time range of each dataset is one year. Following [24], we divide each city into grids of 1km×1km and time intervals of 1 hour. Dataset statistics are shown in Table 2 in the Appendix. We pick NY and CHI, cities with more data as source cities, and DC as the target city. For source cities, we split the final 2 months for testing, the 2 months before for validation, and the rest (8 months) for training. For target cities, we use the same setting for test and validation data, and use the last month, week, and 3 days of data before validation for training.

We also collect POI and road data from OpenStreetMap, and derive human mobility from the taxi and bike datasets for regional feature learning. Details are stated in Section C & D in the Appendix.

Base Models. CrossTReS works with general prediction models F_θ and feature networks F_{θ_f}. For the prediction model, we choose a representative one, ST-Net [24] to perform experiments. For the feature network, we choose MVURE [30] which is an urban region embedding framework based on multi-view graphs. We also set $\mathcal{L}_{emb}$ as the loss function of MVURE (Eqn. 17 in [30]). Implementations of base models can be found in Section E in the Appendix.

Conceptually, CrossTReS can be applied to graph-based traffic prediction models [13, 27], which we leave as future work.

Baselines. We compare CrossTReS with the following transfer learning methods for traffic prediction. They all focus on fine-tuning instead of source training, and are thus *non-selective* baselines.

- **Fine-Tuning**. We first train a model on the source city with Eqn. 2, and then fine-tune it using the data on the target city.
- **RegionTrans** [19]. After source training with Eqn. 2, it computes region-wise similarity using auxiliary data to align temporal features between cities during fine-tuning.
- **MetaST** [24]. MetaST uses meta-learning (Eqn. 3) for source training and transfers the spatio-temporal memory (STMem) containing long-term temporal patterns to the target city.
- **ST-DAAN** [20]. After source training with Eqn. 2, it fine-tunes with DAN [17] by adding adaptation layers and MMD regularization to align temporal features from both cities.

For *selective* baselines, as this is the first work to study selective transfer learning for traffic prediction, there are no existing baselines. Thus, we compare with a method from other fields [16, 28].

- **Sim-Loss**, which re-weights source regions based on 1) a non-adversarial domain classifier, and 2) loss values. Regions similar to the target city with low loss values are emphasized.

We also compare with two non-transfer baselines.

- **ST-Net**. We only train ST-Net on limited target data.
- **ARIMA**, which is a statistical time-series regression model.

Details of baseline methods are in Section E in the Appendix. For fairness, all transfer learning methods use ST-Net as the base model.

In addition, we also include CrossTReS-RT and CrossTReS-Mem for comparisons to show the compatibility of CrossTReS. CrossTReS-RT and CrossTReS-Mem use RegionTrans and STMem (used in MetaST) for fine-tuning, respectively.

Implementation details and hyperparameter settings of CrossTReS can be found in Section E and F in the Appendix. We provide the link to code and data at `https://github.com/KL4805/CrossTReS`.

5.2 Performance Comparison

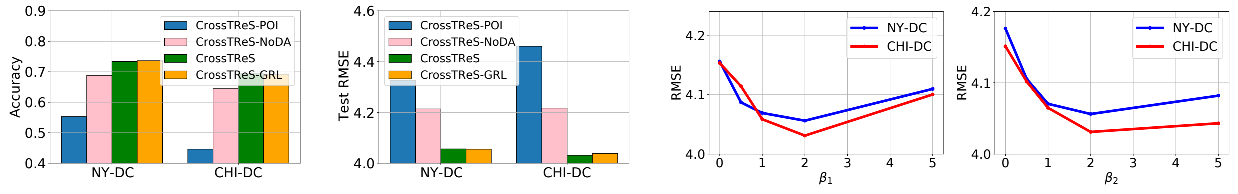
5.2.1 Quantitative Results. We evaluate CrossTReS and baselines on two tasks: DC bike and DC taxi. For each task, we transfer from NY and CHI. We report the means and standard deviations of 5 independent runs in Table 1. We make the following observations.

- The performances of ST-Net degrade significantly with limited data. For example, for taxi volume prediction in DC, ST-Net performs worse than ARIMA with fewer than 7 days of data. The degradation indicates that the problem of transferring traffic prediction models is of pressing importance.
- CrossTReS variants consistently outperform non-selective and selective baselines. Compared with the best baselines (underlined in Table 1), CrossTReS achieves an error reduction of up to 8% under the same settings. The improvements indicate that CrossTReS enables selective and effective cross-city knowledge transfer for traffic prediction tasks.
- CrossTReS-RT and CrossTReS-Mem perform similarly or better than CrossTReS, showing that CrossTReS applies to general fine-tuning methods in cross-city transfer learning.

In addition to the final prediction error on the target city, we also study the speed of adaptation for CrossTReS compared to baselines. Results are discussed in Section G in the Appendix.

Table 1: Evaluation results for bike and taxi volume prediction in Washington DC. For each target data amount and each source city, the best result is bolded and the second best is underlined.

Task	Methods	Target Data	30 days				7 days				3 days			
		Metric	RMSE		MAE		RMSE		MAE		RMSE		MAE	
DC Bike	**No Transfer**	ARIMA	3.44		1.48		3.46		1.50		3.48		1.55	
		ST-Net	2.49		1.24		2.73		1.43		3.14		1.78	
		Source	NY	CHI	NY	CHI	NY	CHI	NY	CHI	NY	CHI	NY	CHI
	Non-selective Transfer	Fine-Tuning	2.317	2.359	1.039	1.048	2.473	2.566	1.085	1.116	2.559	2.699	1.126	1.180
		RegionTrans	2.293	2.352	1.051	1.076	2.453	2.542	1.125	1.144	2.535	2.667	1.163	1.219
		MetaST	2.326	2.341	1.045	1.056	2.486	2.534	1.115	1.124	2.562	2.653	1.157	1.195
		ST-DAAN	2.339	2.371	1.021	1.033	2.475	2.591	1.102	1.091	2.633	2.759	1.120	1.167
	Selective Transfer	Sim-Loss	2.311	2.339	1.029	1.027	2.455	2.529	1.093	1.107	2.614	2.680	1.172	1.205
		CrossTReS	2.187	2.244	**0.968**	**0.984**	**2.300**	2.349	**0.988**	1.032	2.397	2.449	**1.024**	1.068
		Std. Dev.	0.033	0.017	0.010	0.008	0.031	0.009	0.017	0.012	0.018	0.018	0.016	0.031
		CrossTReS-RT	**2.177**	**2.211**	0.984	0.998	2.315	**2.315**	1.017	1.038	**2.377**	2.419	1.058	1.073
		Std. Dev.	0.004	0.021	0.013	0.013	0.029	0.026	0.012	0.015	0.036	0.009	0.013	0.014
		CrossTReS-Mem	2.179	2.231	0.974	**0.984**	**2.299**	**2.313**	1.008	**1.017**	2.391	**2.414**	1.048	**1.031**
		Std. Dev.	0.005	0.026	0.021	0.021	0.015	0.016	0.021	0.007	0.014	0.022	0.021	0.014
DC Taxi	**No Transfer**	ARIMA	5.18		1.76		5.19		1.77		5.20		1.81	
		ST-Net	4.85		2.09		5.74		2.90		6.83		3.82	
		Source	NY	CHI	NY	CHI	NY	CHI	NY	CHI	NY	CHI	NY	CHI
	Non-selective Transfer	Fine-Tuning	4.214	4.148	1.478	1.461	4.420	4.357	1.573	1.574	4.675	4.566	1.723	1.700
		RegionTrans	4.111	4.162	1.472	1.592	4.416	4.347	1.662	1.738	4.778	4.630	1.813	1.911
		MetaST	4.097	4.077	1.450	1.438	4.422	4.351	1.592	1.605	4.672	4.617	1.708	1.708
		ST-DAAN	4.250	4.171	1.482	1.472	4.459	4.408	1.550	1.555	4.898	4.705	1.697	1.679
	Selective Transfer	Sim-Loss	4.186	4.095	1.454	1.438	4.411	4.365	1.589	1.601	4.706	4.544	1.730	1.694
		CrossTReS	3.885	**3.869**	1.381	1.382	4.056	**4.031**	**1.435**	**1.449**	4.326	4.271	1.577	1.558
		Std. Dev.	0.033	0.050	0.021	0.014	0.031	0.039	0.028	0.015	0.041	0.045	0.028	0.040
		CrossTReS-RT	**3.880**	**3.867**	1.393	**1.347**	**4.052**	4.064	1.457	1.477	4.230	**4.235**	1.562	1.548
		Std. Dev.	0.025	0.031	0.019	0.022	0.014	0.031	0.015	0.039	0.017	0.038	0.019	0.040
		CrossTReS-Mem	3.883	3.873	**1.373**	1.364	4.053	4.048	1.460	1.454	**4.211**	4.241	**1.542**	**1.518**
		Std. Dev.	0.020	0.038	0.019	0.022	0.027	0.016	0.011	0.015	0.031	0.023	0.014	0.023



(a) Region Classification (b) Target Performance (c) Parameter β_1 (d) Parameter β_2

Figure 3: Analysis results on node- and edge-level domain adaptations for spatial feature learning.

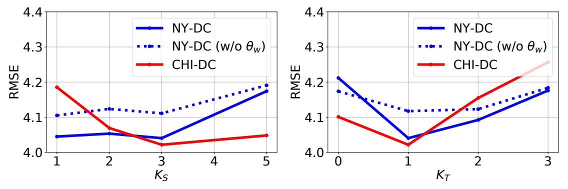
5.3 Model Analysis

In this section, we analyze both parts of CrossTReS, generalizable region feature learning and region re-weighting. We perform analyses using both NY-DC and CHI-DC city pairs and taxi volume prediction tasks with 7-day target data.

5.3.1 Domain Adaptations for Spatial Features. In this section, we study how the learned spatial features affect the overall performance. We use two experiments to show its impact.

- **Region Classification.** We perform region classification experiments to show the quality of the learned region features. We label regions in each city according to the total number of pickups and dropoffs in the dataset. Regions in each city with top 25%, 50% pickups and dropoffs are labeled 0, 1, etc. We mix $\Phi_{\mathcal{S}}, \Phi_{\mathcal{T}}$ from both cities and report a 5-fold cross-validation score using logistic regression.
- **Target Performance.** We follow the evaluation protocol in Sec. 5.2.1 and report performance on the target city (DC).

We compare the following variants of CrossTReS as ablation studies:



(a) Source Update Steps $K_{\mathcal{S}}$ (b) Target Update Steps $K_{\mathcal{T}}$

Figure 4: Analysis results on the joint meta-learning for region re-weighting.

- **CrossTReS-POI**: We do not learn spatial features and use POI vectors (details in Section C in the Appendix) as inputs of the weighting network;
- **CrossTReS-NoDA**: We set $\beta_1 = \beta_2 = 0$ to remove domain adaptation. In this case, $\Phi_{\mathcal{S}}, \Phi_{\mathcal{T}}$ may follow different distributions, which may compromise region re-weighting.
- **CrossTReS**: The full CrossTReS with $\beta_1 = \beta_2 = 2$.
- **CrossTReS-GRL**: We add an adversarial domain discriminator with gradient reversal layer (GRL) [8] along the shared edge classifier $F_{\theta_{edge}}$.

We show results in Fig. 3(a) and 3(b) with the following findings.

- Comparing with CrossTReS-POI, variants that learn spatial features are more precise in both region classification and traffic prediction on target. This shows that the feature network learns indicative spatial features that lead to more effective selective transfer learning.
- Comparing with CrossTReS-NoDA, CrossTReS achieves better performance in both experiments, which shows that by learning generalizable region features via domain adaptation, we better identify functionally similar regions across cities to help selective transfer learning.
- Comparing CrossTReS and CrossTReS-GRL, we observe very similar performances, indicating that the shared edge classifier well performs the edge-level domain adaptation, and that an extra domain discriminator is not required.

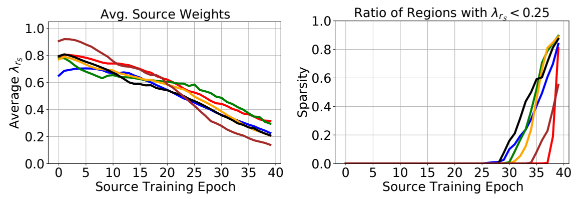
We also study the impact of domain adaptation parameters β_1, β_2. We show the results in Fig. 3(c) and 3(d). As shown, we observe a degradation in performances when either level of domain adaptation is absent ($\beta_1 = 0$ or $\beta_2 = 0$), which indicates that both levels of domain adaptation techniques contribute to learning generalizable spatial features and further performing selective source training.

5.3.2 Joint Meta-Learning for Learning Region Weights. In this section, we analyze the impact of the weighting network to the overall performance via the following experiments.

- We remove the weighting network θ_w and directly use $\Phi_{\mathcal{S}}, \Phi_{\mathcal{T}}$ to compute $\lambda_{r_{\mathcal{S}}}$ via Eqn. 7.
- We vary the number of source and target updates $K_{\mathcal{S}}, K_{\mathcal{T}}$ in the inner loop (Eqn. 12 and 13) to analyze their impacts.

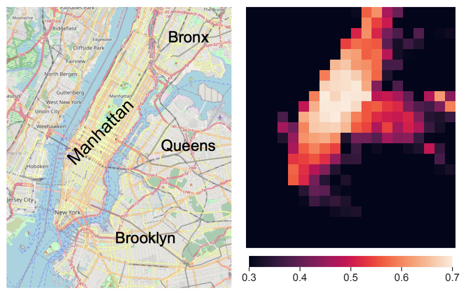
We show results in Fig. 4 and make the following observations.

- The performances slightly degrade without the weighting network θ_w, showing that the weighting network helps further capture indicative features for selective source training.



(a) Mean Values (b) Sparsity

Figure 5: Case study of $\lambda_{r_{\mathcal{S}}}$ during source training.



(a) Map of NY (b) Visualization of $\lambda_{r_{\mathcal{S}}}$

Figure 6: Visualization of source region weights $\lambda_{r_{\mathcal{S}}}$ over NY.

- The best results are obtained with $K_{\mathcal{T}} = 1$, which suggests that with an extra simulation of fine-tuning, helpful source knowledge to target fine-tuning is better identified.

5.4 Case Study

In this section, we analyze the learned $\lambda_{r_{\mathcal{S}}}$ by performing a case study using NY-DC taxi volume prediction tasks.

5.4.1 Mean Values and Sparsity. We analyze the trend of two properties of $\lambda_{r_{\mathcal{S}}}$ during source training: mean values and sparsity. We run experiments for 6 times independently and plot the following values at each source training epoch in each run in Fig. 5,

- **Mean Values**, i.e. the average $\lambda_{r_{\mathcal{S}}}$ over $\mathcal{S}$;
- **Sparsity**, i.e. the ratio of source regions with low weights $\lambda_{r_{\mathcal{S}}} < \varepsilon$; we set $\varepsilon = 0.25$.

We make the following observations:

- The mean weights of source regions gradually decline as source training proceeds, which shows that during source training, the model gradually learns source knowledge that is irrelevant to the target city, leading to the decline in weights.
- The sparsity of $\lambda_{r_{\mathcal{S}}}$ gradually increases as source training proceeds. Moreover, we observe high sparsity ($60 - 80\%$ regions have $\lambda_{r_{\mathcal{S}}} < 0.25$) at the end of source training. Both facts indicate that CrossTReS gradually learns to rule out irrelevant source regions by assigning them low weights.
- Both metrics show similar trends across different runs, showing that CrossTReS can stably select helpful source regions.

5.4.2 Visualization of λ_{r_S}. We visualize the learned weights to illustrate the selection by CrossTReS. We show both the map of NY and a heatmap of the average final λ_{r_S} in Fig. 6. As shown, Manhattan has much higher weights than New Jersey, Brooklyn, Bronx, and Queens. The phenomenon can be interpreted in two ways. On one hand, similar to Washington DC, Manhattan has many tourist attractions, which leads to a similarity in popular destinations. On the other hand, Washington DC enjoys high economic development, while Manhattan is the most economically developed county in NY.[2] We thus conclude that the selection by CrossTReS well corresponds with various notions of cross-city region similarity.

6 CONCLUSION

We present CrossTReS, a selective cross-city transfer learning framework for traffic prediction by adaptively re-weighting source regions to improve target fine-tuning. CrossTReS applies to general fine-tuning-based methods. We first learn generalizable features for urban regions via node and edge-level domain adaptation methods to uncover shared urban functions between cities. We further adaptively re-weight source regions via a source-target joint meta-learning, such that selective source training with the learned weights leads to low target error after fine-tuning. We perform experiments on real-world taxi and bike datasets, where CrossTReS outperforms state-of-the-art baselines by up to 8%. We further visualize the learned region selection in an interpretable manner.

ACKNOWLEDGMENTS

We are grateful to Hao Liu and Xu Geng for helpful discussions and comments. 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 (TACC) [22].

REFERENCES

[1] 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.

[2] Zhangjie Cao, Kaichao You, Mingsheng Long, Jianmin Wang, and Qiang Yang. 2019. Learning to transfer examples for partial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision & Pattern Recognition.* 2985–2994.

[3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning.* PMLR, 794–803.

[4] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. 2007. Boosting for Transfer Learning. In *International Conference on Machine Learning.* ACM, 193–200.

[5] Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. 2018. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224* (2018).

[6] 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.

[7] Yanjie Fu, Pengyang Wang, Jiadi Du, Le Wu, and Xiaolin Li. 2019. Efficient region embedding with multi-view spatial networks: A perspective of locality-constrained spatial autocorrelations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 906–913.

[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] Xu Geng, Yaguang Li, Leye Wang, Lingyu Zhang, Qiang Yang, Jieping Ye, and Yan Liu. 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3656–3663.

[10] Bin Guo, Jing Li, Vincent W Zheng, Zhu Wang, and Zhiwen Yu. 2018. Citytransfer: Transferring inter-and intra-city knowledge for chain store site recommendation based on multi-source urban data. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–23.

[11] Xueting Han, Zhenhuan Huang, Bang An, and Jing Bai. 2021. Adaptive Transfer Learning on Graph Neural Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 565–574.

[12] 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.

[13] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations.*

[14] Zheng Li, Ying Wei, Yu Zhang, and Qiang Yang. 2018. Hierarchical attention transfer network for cross-domain sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. 5852–5859.

[15] Zheng Li, Yun Zhang, Ying Wei, Yuxiang Wu, and Qiang Yang. 2017. End-to-End Adversarial Memory Network for Cross-domain Sentiment Classification.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence.* 2237–2243.

[16] Yan Liu, Bin Guo, Daqing Zhang, Djamal Zeghlache, Jingmin Chen, Ke Hu, Sizhe Zhang, Dan Zhou, and Zhiwen Yu. 2021. Knowledge Transfer with Weighted Adversarial Network for Cold-Start Store Site Recommendation. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15, 3 (2021), 1–27.

[17] 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.

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

[19] 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.

[20] Senzhang Wang, Hao Miao, Jiyue Li, and Jiannong Cao. 2022. Spatio-Temporal Knowledge Transfer for Urban Crowd Flow Prediction via Deep Attentive Adaptation Networks. *IEEE Transactions on Intelligent Transportation Systems* 23, 5 (2022), 4695–4705.

[21] Ying Wei, Yu Zheng, and Qiang Yang. 2016. Transfer knowledge between cities. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 1905–1914.

[22] 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:2110.01556* (2021).

[23] Shuwen Yang, Guojie Song, Yilun Jin, and Lun Du. 2020. Domain Adaptive Classification on Heterogeneous Information Networks. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence.* 1410–1416.

[24] 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.

[25] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5668–5675.

[26] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32. 2588–2595.

[27] 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.

[28] Jing Zhang, Zewei Ding, Wanqing Li, and Philip Ogunbona. 2018. Importance weighted adversarial nets for partial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 8156–8164.

[29] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31. 1655–1661.

[30] Mingyang Zhang, Tong Li, Yong Li, and Pan Hui. 2020. Multi-View Joint Graph Representation Learning for Urban Region Embedding. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence.* 4431–4437.

[31] Yunchao Zhang, Yanjie Fu, Pengyang Wang, Xiaolin Li, and Yu Zheng. 2019. Unifying inter-region autocorrelation and intra-region structures for spatial embedding via collective adversarial learning. In *Proceedings of 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1700–1708.

[2] According to the Bureau of Economic Analysis (https://bea.gov), in terms of annual income per capita, Washington DC ranks 46 in 3140 US counties, while Manhattan ranks 2, Bronx ranks 1987, Brooklyn ranks 453, and Queens ranks 568.

Algorithm 1 Selective Cross-City Transfer Learning for Traffic Prediction with CrossTReS

Input: Source and target traffic data $\mathcal{X}_\mathcal{S}, \mathcal{X}_\mathcal{T}$,
Multi-view urban data $\{\mathcal{G}_v^C\}, v \in \{prox, road, poi, s, d\}, C \in \{\mathcal{S}, \mathcal{T}\}$,
Output: A deep prediction model $\theta_\mathcal{T}$ for $\mathcal{T}$

```
1:  Set $s\_epoch = 0, tune\_epoch = 0$.
2:  while $s\_epoch < T_s$ do
3:      Train feature network $\theta_f$ via Eqn. 11.
4:      Train $\theta_s = \{\theta_f, \theta_w\}$ via Eqn. 12, 13, and 14.
5:      Obtain source weights $\lambda_{r_\mathcal{S}}$.
6:      Train $\theta$ on source data $\mathcal{X}_\mathcal{S}$ via Eqn. 5 with weights $\lambda_{r_\mathcal{S}}$.
7:      $s\_epoch = s\_epoch + 1$.
8:  end while
9:  while $tune\_epoch < T_{tune}$ do
10:     Train model $\theta$ on target data $\mathcal{X}_\mathcal{T}$.
11:     $tune\_epoch = tune\_epoch + 1$.
12: end while
13: return Trained model $\theta_\mathcal{T}$.
```

Table 2: Detailed statistics of the datasets. # Taxis, # Bikes denotes the number of taxi/bike trips. Longitude and Latitude show the detailed spatial ranges of the selected datasets.

City	Longitude (W)	Latitude (N)	Time	# Taxis	# Bikes
Abbr.	W_c	H_c			
New York	[74.059, 73.863]	[40.645, 40.848]		133M	13.8M
NY	20	23			
Chicago	[87.740, 87.576]	[41.766, 42.013]	1/1-31/12, 2016	24.5M	3.5M
CHI	17	28			
Washington	[77.127, 76.926]	[38.798, 38.969]		10M	2.7M
DC	21	20			

A PSEUDOCODE OF CROSSTRES

The detailed pseudocode of CrossTReS is shown in Algorithm 1.

B FURTHER RELATED WORKS

B.1 Transfer Learning in Urban Computing

In addition to traffic prediction, transfer learning is also studied for other tasks in urban computing, including FLORAL for air quality prediction [21], CityTransfer and WANT [10, 16] for site recommendation. However, FLORAL focuses on tree classifiers, while WANT and CityTransfer focus on recommendation models. Therefore, none of them can be directly applied to our problem.

C DATA PREPARATION

We collect bike and taxi data from New York[3], Chicago[4], and Washington DC[5]. We show the exact spatial range of the selected datasets in Table 2. All datasets contain taxi/bike trips that include time and geographical coordinates of pickups and dropoffs. We accordingly process them into arrays recording the number of pickups/dropoffs in each time interval. In addition, the taxi datasets in Chicago and Washington DC involve both geographical and temporal rounding, i.e. the exact time is rounded to the nearest 15/30 minutes, and the exact locations are rounded to centers of census tracts. In these cases, we follow the approximate time and location information provided by the data. Regions that do not record any pickup/dropoffs will not be included in computing errors.

For multi-view urban data, in addition to human mobility extracted by bike and taxi trips, we collect road information and POI attributes from OpenStreetMap[6]. For road information, we process it into a graph $\mathcal{G}_{road}^C$. Two regions $(r_1, r_2) \in \mathcal{G}_{road}^C$ if they are connected by highways. For POI information, we process it into a matrix $\mathbf{P}_C \in \mathbb{R}^{|C| \times 14}$. $\mathbf{p}_r \in \mathbb{R}^{14}$ denotes the number of POIs of each class in region $r \in C$. We consider the following POI classes:

- 0: Scenic spots;
- 1: Medical and health services;
- 2: Domestic services;
- 3: Residential areas;
- 4: Financial institutions (e.g. banks);
- 5: Sports and leisure services;
- 6: Cultural and educational services;
- 7: Shopping;
- 8: Housing services (e.g. hotels);
- 9: Governments and organizations;
- 10: Corporations;
- 11: Catering;
- 12: Transportation;
- 13: Public services.

D MULTI-VIEW URBAN GRAPH CONSTRUCTION

We provide construction rules of the multi-view urban graphs.

- **Proximity $\mathcal{G}_{prox}^C$.** We follow [9] and link each region r with 8 regions that are within the 3×3 grid centered at r. Regions at the boundaries will link to fewer than 8 regions.
- **Road Connections $\mathcal{G}_{road}^C$.** Regions connected with highways are connected with undirected edges.
- **POI $\mathcal{G}_{poi}^C$.** Regions with similar POI distributions indicate similar urban functions. Thus, given region r, we connect it with regions with top k cosine similarity
$$\text{POISim}(r, r') = \text{CosSim}(\mathbf{p}_r, \mathbf{p}_{r'}), r, r' \in C, \tag{15}$$
where $\mathbf{p}_r$ is the POI vector of r with $\mathbf{p}_{r,i}$ denoting the number of POIs of class i in region r.
- **Human Mobility $\mathcal{G}_s^C, \mathcal{G}_d^C$.** Human mobility patterns shed light on regional functions. For example, traveling between a residential area and a business area should be more frequent than that between residential areas. To this end, given origin-destination (OD) pairs $\mathcal{M}_C = \{(r_s, r_d)\}$ from human mobilities, we compute weights between regions $r, r' \in C$ as $w(r, r') = |\{(r_s, r_d) \in \mathcal{M}_C | r_s = r, r_d = r'\}|$, and further compute the source and destination distribution of region r to model its mobility patterns as:
$$\mathbf{s}(r_i|r) = \frac{w(r_i, r)}{\sum_{r'} w(r', r)}, \mathbf{d}(r_i|r) = \frac{w(r, r_i)}{\sum_{r'} w(r, r')} \tag{16}$$

[3]https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
https://www.citibikenyc.com/system-data
[4]https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew
https://www.divvybikes.com/system-data
[5]https://opendata.dc.gov/search?q=taxi%20trips
https://www.capitalbikeshare.com/system-data

[6]https://openstreetmap.org

Finally, we connect each region r with regions of top k similarity (in KL divergence) in source distribution $\mathbf{s}(\cdot|r)$ to form $\mathcal{G}_s^C$, and similarly $\mathcal{G}_d^C$.

E IMPLEMENTATION DETAILS

We implement CrossTReS with PyTorch and DGL. We implement ST-Net by stacking 3 3×3 residual blocks with 64 channels [29] and a single-layer LSTM with 128 hidden units. We set the prediction horizon $\tau = 6$, i.e. the input data consist of observations from 6 previous intervals. We implement the feature network F_{θ_f} as an MVURE model [30] with 2-layer GATs with 2 attention heads. We use POI vectors $\mathbf{P}_C$ as input features and set hidden and embedding dimensions d_{emb} as 64. We set $k = 10$ for linking top k similar regions as suggested. The implementation of MVURE follows the official code [7]. We implement both the weighting network F_{θ_w} and the edge classifier $F_{\theta_{edge}}$ as 2-layer MLPs with 64 hidden units.

Implementation details of baseline methods are as follows:

- We use ARIMA models with 6 AR steps, 1 MA step, and 1 integration step. We use the official code and hyperparameters for MetaST[8] and ST-DAAN[9].
- We implement RegionTrans as we fail to find codes for it. The original RegionTrans uses check-in data for region matching. However, as we do not have this type of data, we use POI data $\mathbf{P}_C$ for region matching instead and tune the w parameter (Eqn. 14 in [19]) to achieve optimal results.
- The implementation of Sim-Loss follows Eqn. 11-16 in WANT [16]. The weight $\lambda_{r_\mathcal{S}}$ of region $r_\mathcal{S}$ is the product of two terms:
  - The output of a non-adversarial domain classifier $DISC(r_\mathcal{S}) \in [0,1]$. A higher $DISC(r_\mathcal{S})$ indicates a higher similarity to the target city.
  - Whether the loss value on $r_\mathcal{S}$ is sufficiently low
  $$\mathbb{I}(L(\hat{x}_{r_\mathcal{S}}, x_{r_\mathcal{S}}) < \gamma) \in \{0,1\}$$

  We fail to find codes for WANT, and thus we implement the selection rules. We implement the non-adversarial domain classifier with 2 3×3 residual blocks and manually tune the threshold γ to achieve the optimal results.

We replace the base model for all baselines with ST-Net for a fair comparison. For all fine-tuning-based baselines, we train the model for 100 epochs on source data and choose the model with the lowest source validation error to initialize fine-tuning.

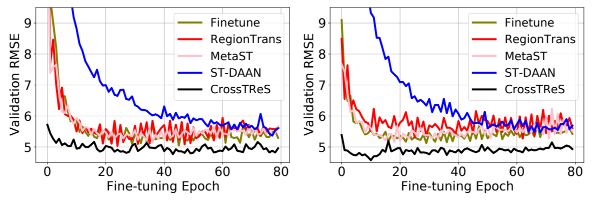
We choose ST-Net [24] as the base model because ST-Net is built upon stacking convolutional layers for spatial features, and recurrent layers for temporal features, both of which are commonly used in traffic prediction tasks. In addition, stacking the two types of layers to form a prediction model is also a common practice [19, 20]. We thus consider ST-Net as a representative base model.

All experiments are performed on a single NVIDIA Tesla V100 GPU with 32GB memory.

F HYPERPARAMETERS

We tune hyperparameters of CrossTReS using the NY-DC city pair on validation data. To learn the feature network via Eqn. 11, we set $\beta_1 = \beta_2 = 2$, learning rate 10^{-3}. In addition, to improve stability in the first epochs, we train the feature network for 80 epochs before performing source region selection via joint meta-learning. For joint meta-learning. We set the inner learning rate $\alpha = 5 \times 10^{-5}$, outer learning rate $\gamma = 10^{-4}$, $K_\mathcal{S} = 3, K_\mathcal{T} = 1$. For the prediction model, we perform source training for $T_s = 100$ epochs with early stopping and fine-tune for $T_{tune} = 80$ epochs use learning rate 8×10^{-4} with weight decay 5×10^{-5}. We use Adam optimizer.

[7]https://github.com/mingyangzhang/mv-region-embedding
[8]https://github.com/huaxiuyao/MetaST
[9]https://github.com/MiaoHaoSunny/ST-DAAN



(a) NY-DC (b) CHI-DC

Figure 7: Speed of adaptation on taxi volume prediction tasks.

Table 3: Experimental results on Hong Kong taxi data.

Methods		RMSE		MAE	
No-Transfer	ST-Net	0.350		0.150	
	Source	NY	CHI	NY	CHI
Transfer	Fine-Tuning	0.337	0.331	0.136	0.139
	RegionTrans	0.332	0.328	0.142	0.144
	MetaST	0.331	0.330	0.137	0.138
	ST-DAAN	0.342	0.336	0.135	0.133
	CrossTReS	**0.325**	**0.322**	**0.131**	**0.130**

G ADDITIONAL EXPERIMENTAL RESULTS

G.1 Speed of Adaptation

In addition to the final prediction error on the target city, how fast the model adapts to target data is also an important metric in transfer learning. We compare the speed of adaptation for all methods on taxi datasets with 7-day target data. We evaluate validation error on the target city for each fine-tuning epoch and plot the curve in Fig. 7. As shown, CrossTReS achieves both the lowest error and the fastest adaptation, which further verifies that CrossTReS selects relevant knowledge for effective transfer learning.

H DISCUSSIONS & FUTURE WORK

We discuss future work by performing experiments using taxi data from Hong Kong. The dataset contains 6 months of data. We take Hong Kong as the target city with 7 days of training data and show results in Table 3. One major difference between the Hong Kong dataset and those in Table 2 is that POI data in Hong Kong is highly sparse, which negatively impacts regional feature learning. Therefore, CrossTReS only marginally outperforms baselines (up to ~3%). We thus consider it important to deal with missing and low-quality data views in future work.