When Transfer Learning Meets Cross-City Urban Flow Prediction: Spatio-Temporal Adaptation Matters

Ziquan Fang[1] , **Dongen Wu**[2] , **Lu Pan**[1] , **Lu Chen**[1] and **Yunjun Gao**[1]
[1]College of Computer Science, Zhejiang University, Hangzhou, China,
[2]College of Computer Science, Zhejiang University of Technology, Hangzhou, China
[1]{zqfang, play, luchen, gaoyj}@zju.edu.cn, [2]wude@zjut.edu.cn

Abstract

Urban flow prediction is a fundamental task to build smart cities, where neural networks have become the most popular method. However, the deep learning methods typically rely on massive training data that are probably inaccessible in real world. In light of this, the community calls for knowledge transfer. However, when adapting transfer learning for cross-city prediction tasks, existing studies are built on static knowledge transfer, ignoring the fact inter-city correlations change dynamically across time. The dynamic correlations make urban feature transfer challenging. This paper proposes a novel Spatio-Temporal Adaptation Network (STAN) to perform urban flow prediction for data-scarce cities via the spatio-temporal knowledge transferred from data-rich cities. STAN encompasses three modules: i) spatial adversarial adaptation module that adopts an adversarial manner to capture the transferable spatial features; ii) temporal attentive adaptation module to attend to critical dynamics for temporal feature transfer; iii) prediction module that aims to learn task-driven transferable knowledge. Extensive experiments on five real datasets show STAN substantially outperforms state-of-the-art methods.

1 Introduction

Urban flow prediction is indispensable for constructing smart cities and intelligent traffic systems (ITS). It is a typical spatio-temporal prediction task that aims to forecast future crowd flows over citywide regions based on historical observation data (e.g., taxi and bike trip records). Accurate urban flow prediction serves a broad scope of applications in transportation [Yuan *et al.*, 2020], mobility mining [Mazimpaka and Timpf, 2016], and urban computing [Zheng, 2015]. Due to its great benefits, many studies are proposed [Jiang *et al.*, 2021; Yuan and Li, 2021; Ye *et al.*, 2020]. The essential task behind is to capture the spatio-temporal urban dependence for spatio-temporal data prediction.

Recently, inspired by the success of deep neural networks, deep learning based methods such as [Xingjian *et al.*, 2015; Zhang *et al.*, 2017; Li *et al.*, 2018a; Yao *et al.*, 2019b; Liang *et al.*, 2019; Zhang *et al.*, 2020; Yuan *et al.*, 2021] are emerging and achieve remarkable performance. While the researchers apply increasingly compelling networks (varying from RNNs to CNNs to GNNs) to improve the accuracy, it simultaneously exposes unignorable problems. Specifically, the superior performance of these methods highly relies on large-scale training data such as massive vehicle trip records or even auxiliary weather information [Liao *et al.*, 2018], which are usually inaccessible in real world. In reality, data-scare issues commonly exist due to sparse positioning stations, data privacy principles, and low city development levels [Wang *et al.*, 2019]. For instance, some cities like New York and Beijing have released taxi data for a couple of years, while others like Chengdu may only release a few days of data. Although travel platforms (e.g., Didi and Uber) may collect massive vehicle trips via navigation services, they cannot release the data under privacy issues [Tong *et al.*, 2017]. As a result, existing approaches trained on data-rich cities (e.g., NYC and BJ) are hard to accommodate those cities with sparse data [Wang *et al.*, 2018]. Correspondingly, their performance may degrade when the training samples decrease.

Targets. Instead of proposing yet another deep learning approach requiring a large volume of spatio-temporal data to further optimize the model accuracy, in this paper, we propose a transfer learning framework to perform urban flow prediction in a data-scarce target city via the knowledge transferred from a data-rich source city. The core task is to capture the shareable (common) spatio-temporal features between the source and target cities, in which case, the model trained on source city could be generalized to support target city.

Although recent studies [Wang *et al.*, 2019; Yao *et al.*, 2019a; Wang *et al.*, 2021] have made some efforts, they mainly focus on static and spatial-only feature transfer while neglecting the dynamic correlations accompanying the time dimension. The dynamic here means that the interactions of two cities at different times are supposed to be different, as highlighted by *red dotted rectangle* in Fig. 1 (b). In view of this, the temporal dimension must be transferred with the spatial dimension. Motivated by Domain Adaptation [Long *et al.*, 2015; Chen *et al.*, 2019] (a type of transfer learning methods in Computer Vision) to learn domain-invariant features across two images, we borrow its idea to bridge source domain (city) and target domain (city) in isomorphic latent space considering spatial and temporal aspects to capture the domain/city invariant features for target city prediction.

However, it is a nontrivial endeavor to design such a domain adaptation approach due to the following challenges.

- **C1: How to capture the spatial domain-invariant correlations between the source and target cities?** As depicted in Fig. 1, the spatial correlations between two cities may vary across time. To learn sufficient features, a natural way is to use discrepancy reduction methods to explicitly compute the spatial distributions of two cities at each time step. However, in that case, massive spatial features (each time step corresponds to a feature collection) are yielded to be computed, which could result in expensive computational costs.
- **C2: How to capture the temporal domain-invariant correlations between the source and target cities?** Existing methods typically neglect the temporal correlations between source and target cities, which constrains themselves from temporal-aware feature transfer. The temporal dynamics can be viewed as a combination of multiple local temporal features corresponding to different spatial features. However, not all the local temporal features equally contribute to overall spatio-temporal feature transfer. To address this, a promising way is to focus more on those with high contributions to the domain discrepancy between the source and target cities.

With these challenges in mind, we propose a Spatio-Temporal Adaptation Network called STAN, which consists of a spatial adversarial adaptation module (SAAM), a temporal attentive adaptation module (TAAM), and a prediction module (PM). Specifically, we design the SAAM to solve the first challenge. Inspired by GANs [Goodfellow *et al.*, 2014], it distinguishes the spatial features from the source and target cities at each time step via adversarial classification to effciently capture spatio-transferable features in an efficient manner. To overcome the second challenge, we design the TAAM, which adopts attention mechanisms when performing domain discrepancy reduction. By doing it, the temporal dynamics that contribute more to the domain discrepancy will be paid more attention to, leading to more effective spatio-temporal feature transfer. Finally, under the guidance of PM, STAN enable transferring task-driven features for data-scare spatio-temporal urban flow prediction.

In summary, this paper makes the following contributions.

- *Spatio-Temporal Transfer Learning.* We propose a novel transfer learning framework to capture sufficient spatio-temporal transferable knowledge via domain adaptation. The framework enables cross-city urban flow prediction, which can effectively address data-scarce problems.
- *Spatial Domain Adaptation.* We design a spatial adversarial adaptation module to transfer spatial domain-invariant features while learning dynamic spatial correlations in an efficient adversarial-based classification way, successfully avoiding high computational costs.
- *Temporal Domain Adaptation.* We design a temporal attentive adaptation module, which simultaneously aligns temporal domains and encodes temporal dynamics into spatiotemporally-aware representations. Moreover, it utilizes attention mechanisms to capture critical temporal features for fine-grained transfer learning.

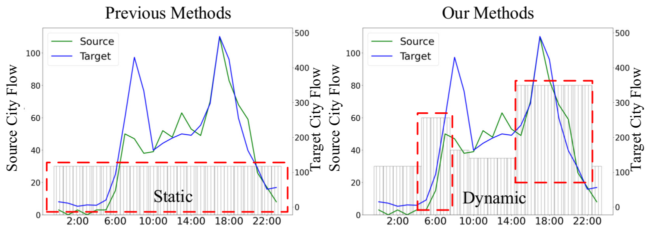

Figure 1: Example of Inter-City Static and Dynamic Modeling

- We conduct comprehensive experiments to evaluate STAN. The results show that STAN i) outperforms both fine-tuned deep learning methods and state-of-the-art transfer learning methods; ii) exhibits superior robustness to data scarcity and parameter settings; and iii) shows remarkable model efficiency.

2 Problem Formulation

Definition 1 (Region)**.** *Following previous works [Wang* et al.*, 2019; Yao* et al.*, 2019a], we spatially split a city c into a $h \times w$ (e.g., 16×16) grid image based on the longitude and latitude. Each grid is defined as a region $r_{i,j}$ that locates at i_{th} row and j_{th} column, and all the regions of the city form a region image $R_c = \{r_{1,1}, ..., r_{i,j}, ..., r_{h,w}\}$.*

Definition 2 (Inflow/Outflow)**.** *In urban context, each region $r_{i,j}$ during time interval t involves two types of flows, namely inflow and outflow, which can be computed based on the historical vehicle trip records/trajectories.*

$$\begin{aligned} x_t^{(r_{i,j},\, in)} &= \sum_{\tau \in \mathbb{T}} |g_{t-1} \notin r_{i,j} \& g_t \in r_{i,j}| \\ x_t^{(r_{i,j},\, out)} &= \sum_{\tau \in \mathbb{T}} |g_{t-1} \in r_{i,j} \& g_t \notin r_{i,j}| \end{aligned} \tag{1}$$

where $\mathbb{T}$ is a trajectory set, $\tau = \{g_1, ..., g_t, ..., g_n\}$ is a trajectory of length n, and g_t denotes t-th GPS location of τ.

Definition 3 (Urban Flow Image)**.** *Given region image $R = \{r_{1,1}, ..., r_{i,j}, ..., r_{h,w}\}$ of a city, we define the inflow and outflow during time interval t of all regions as a urban flow image donated as a matrix $X_t \in \mathcal{R}^{h \times w \times 2}$.*

Here, the in/outflow in each region can be replaced with any other spatio-temporal data such as pickup and weather data so that our proposed model can flexibly support spatio-temporal prediction tasks such as demand and environment forecasting.

Definition 4 (Urban Flow Image Time-Series)**.** *Given the latest time interval t and historical time range k, we denote the urban flow image time-series as $\mathcal{X} = \{X_{t-k+1}, ..., X_{t-1}, X_t\} \in \mathcal{R}^{k \times h \times w \times 2}$, where $\mathcal{X}(i, j, t, *)$ denotes the inflow and outflow in region $r_{i,j}$ during t.*

Problem Definition. Given the sufficient data in a source city S and insufficient data in a target city T (i.e., $|\mathcal{X}^S| \gg |\mathcal{X}^T|$), transfer-based cross-city urban flow prediction aims to learn a function f to forecast the crowd flow in target city T of all regions during the next time interval $t + 1$:

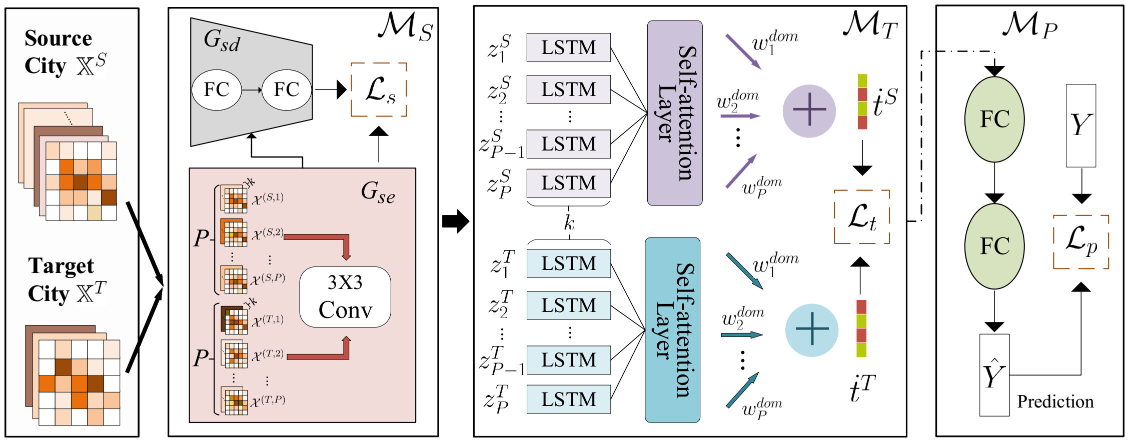

Figure 2: The Proposed STAN Framework that consists of Spatial Module $\mathcal{M}_S$, Temporal Module $\mathcal{M}_T$, and Prediction Module $\mathcal{M}_P$

$$\min_f \sum_{t+1} error\left(Y_{t+1}^T, \hat{Y}_{t+1}^T\right), \hat{Y}_{t+1}^T = f\left(\mathcal{X}^S, \mathcal{X}^T\right) \quad (2)$$

Here, $\hat{Y}_{t+1}^T$ is the predicted result while Y_{t+1}^T is the ground truth. *error* can be measured using mean absolute error (MAE), root mean squared error (RMSE), etc.

3 Methodology

3.1 Framework Overview

Fig. 2 shows the framework of STAN, which contains three major modules $\mathcal{M}_S$, $\mathcal{M}_T$, and $\mathcal{M}_P$. STAN runs three modules jointly for spatio-temporal knowledge transfer predictive learning. Three losses including spatial domain loss $\mathcal{L}_s$, temporal domain loss $\mathcal{L}_t$ and prediction loss $\mathcal{L}_p$ are aligned with three modules, respectively. A mixed loss function is further designed for model training. The detailed structure of each module is described in the following subsections.

3.2 Spatial Adversarial Adaptation Module

As shown in Fig. 2, SAAM is composed of spatial extractor G_{se} and spatial discriminator G_{sd} to learn spatial transferable features in an adversarial way.

Spatial Feature Extractor G_{se}. Following most of previous studies, we apply CNNs to extract the spatial features since we treat source/target cities like images. Specifically, given historical data of P days where each day contains k time intervals before the current time interval of the source city, the input urban flow $\mathbb{X}^S = \left\{\mathcal{X}^{(S,1)}; \mathcal{X}^{(S,2)}; ...; \mathcal{X}^{(S,P)}\right\} = \left\{X_{t-k}^{(S,1)}, ..., X_t^{(S,1)}; X_{t-k}^{(S,2)}, ..., X_t^{(S,2)}; ...; X_{t-k}^{(S,P)}, ..., X_t^{(S,P)}\right\} \in \mathcal{R}^{P\times k\times h\times w\times 2}$. Next, each $\mathcal{X}^{(S,i)}$ (the crowd flow of source city at day i) is fed into a convolution layer as below:

$$z_i^S = \text{conv}\left(\mathcal{X}^{S,i}, w\right) \quad (3)$$

According to Eq. 3, we can obtain the initial spatial feature set of the target city, i.e., $Z^S = \left\{z_1^S, ..., z_{P-1}^S, z_P^S\right\} \in \mathcal{R}^{P\times k\times N\times D}$, where k denotes the channels of kernel, N is the dimension of the two-dimensional matrix after paving, and D is the output feature dimension. Similarly, we can obtain the spatial feature set of the target city, i.e., $Z^T = \left\{z_1^T, ..., z_{P-1}^T, z_P^T\right\} \in \mathcal{R}^{P\times k\times N\times D}$ in the same way.

Spatial Feature Discriminator G_{sd}. Based on the obtained Z^S and Z^T, we further label these $P \times k$ samples (i.e., spatial features) as 1 or 0, where 1 is assigned to source city data as real data while 0 is assigned to target city data as fake data. Then, as depicted in Fig. 2, we introduce a spatial feature discriminatory G_{sd} via fully connected (FC) layers. Specifically, we mix the labeled Z^S and Z^T as the input of G_{sd}. G_{sd} then distinguishes the features from the source or target city. Its gradient training function is defined as follows:

$$\nabla W_d \frac{1}{(P)\times k}\left[\log G_{sd}\left(z_{(i)}^s\right) + \log\left(1 - G_{sd}\left(G_{se}\left(z_{(i)}^t\right)\right)\right)\right] \quad (4)$$

where W_d denotes the parameters of G_{sd}. $z_{(i)}^s$ and $z_{(i)}^t$ represent spatial feature of source and target cities, respectively. In terms of G_{sd} training, we add the gradient in Eq. 4 to get larger gradient for better classification performance. In contrast, G_{se} aims to mix the spatial features of two cities as close as possible, so it subtracts the gradient for training:

$$\nabla W_e \frac{1}{(P)\times k} \log\left(1 - G_{sd}\left(G_{se}\left(z_{(i)}^t\right)\right)\right) \quad (5)$$

where W_e denotes the parameters of G_{se}.

Spatial Domain Loss $\mathcal{L}_s$. Through the adversarial objectives, the discriminator G_{sd} is optimized to classify input spatial features into different domains, while the feature extractor G_{se} is optimized in the opposite direction. Then, we define the spatial domain loss function as:

$$\mathcal{L}_s = \text{BCELoss}\left(1, G_{sd}\left(G_{se}\left(z_{(i)}^t\right)\right)\right) \quad (6)$$

BCELoss is widely used in classification tasks. The smaller the $\mathcal{L}_s$, the closer the spatial distributions of two cities.

3.3 Temporal Attentive Adaptation Module

As depicted in Fig. 2, TAAM $\mathcal{M}_t$ is designed for time-aware feature transfer as urban discrepancies are also reflected in the temporal dimension. Specifically, we first use the self-attention mechanisms to capture the domain weight, representing the impact of local temporal features of source and target cities. Then, the discrepancy-based domain adaptation method, i.e., MMD [Long *et al.*, 2017] is employed to capture the global temporal features from source and target cities.

Self-attention Based Domain Weight. We first utilize LSTM for local temporal feature embedding as LSTM is originally designed to capture the correlations in time series. As observed, we feed $Z^S = \left\{z_1^S, ..., z_{P-1}^S, z_P^S\right\}$ and $Z^T = \left\{z_1^T, ..., z_{P-1}^T, z_P^T\right\}$ to P LSTMs. The length of each LSTM is k. After concating the output of each LSTM, we compute the initial local temporal feature set of source and target cities, i.e., $t^S = \left\{t_1^S, ..., t_{P-1}^S, t_P^S\right\} \in \mathcal{R}^{P\times k\times D}$ and $t^T = \left\{t_1^T, ..., t_{P-1}^T, t_P^T\right\} \in \mathcal{R}^{P\times k\times D}$.

Based on local temporal features, we can compute the global temporal features by summing each local temporal feature. However, not all the local temporal features equally contribute to the global temporal feature, and thus the internal correlation between local temporal features matters. To address this, as shown in Fig. 2, $\mathcal{M}_t$ introduces self-attention layers to consider both local and global aspects.

Specifically, to perform feature adaptation between source and target cities, we design a domain weight to capture those important local temporal features while discarding those local temporal features that may negatively affect domain transfer. The weight α of self-attention is defined as:

$$\begin{gathered} Q = t^S * W^Q, K = t^S * W^K, V = t^S * W^V \\ \alpha = \mathrm{softmax}\left(\frac{Q * K^{\mathrm{T}}}{\sqrt{D_k}}\right) \end{gathered} \tag{7}$$

where $Q, K, V \in \mathcal{R}^{P\times P}$ represent query vector, key vector and value vector, respectively. $W^Q, W^k, W^V \in \mathcal{R}^{D\times k\times P}$ are random initialization parameter matrices, $\alpha \in \mathcal{R}^{P\times P}$ is the weight matrix w.r.t. Q, K, V. D_k is the dimension of Q or V. Each local temporal feature can be calculated as:

$$t_p^S = \sum_{j=1}^{P} v_j * \alpha_{pj} \tag{8}$$

where v_j is j-th vector of V, α_{pj} represents correlation weight of j-th local feature to p-th local feature. Then, an initial global temporal feature of source city is:

$$\mathtt{t}^S = \sum_{p\in P} t_p^S \tag{9}$$

The initial global temporal feature of the target city, i.e., $\mathtt{t}^T$, can be obtained in the same way. Next, we consider reducing the feature discrepancies between the source and target cities in terms of local-to-local and global-to-global dimensions. Specifically, we utilize MMD that is widely used as a distribution distance measure.

$$\mathrm{MMD}\left(F^S, F^T\right) = \left\| \frac{1}{n}\sum_{i=1}^{n} E\left(f_i^S\right) - \frac{1}{m}\sum_{j=1}^{m} E\left(f_j^T\right) \right\|_{\mathcal{H}}^2 \tag{10}$$

where $F^S = f_*^S$ and $F^T = f_*^T$ denote the features of two cities, respectively. n and m denote feature dimensions. $E(\cdot)$ represents kernel function such as Gaussian kernel, and $\mathcal{H}$ represents Hilbert space. According to Eq. 10, the local temporal feature discrepancy of day p is:

$$d_p(S,T) = \frac{1}{M}\sum_{i=1}^{M} \left\| E\left(t_{p,i}^S\right) - E\left(t_{p,i}^T\right) \right\|_{\mathcal{H}}^2 \tag{11}$$

where M denotes feature dimensions. The global temporal feature d_g is calculated by bring $\mathtt{t}^S$ and $\mathtt{t}^T$ into Eq. 11. Finally, based on the local temporal discrepancy d_p and global discrepancy d_g, we design a domain weight strategy to evaluate the correlations between local temporal feature discrepancy and the global temporal feature discrepancy as below.

$$d_p^g = |d_g - d_p|, \quad w_p^{dom} = \frac{e^{d_p^g}}{\sum_{p\in P} d_p^g} \tag{12}$$

where d_p^g measures the discrepancy between each local temporal feature of day p and the global temporal feature.

Overall Temporal Domain Adaptation. Final, the overall temporal feature of source city and target city can be obtained by bridging Eq. 12 and Eq. 9.

$$\dot{\mathtt{t}}^S = \sum_{p\in P} w_p^{dom} t_p^S, \quad \dot{\mathtt{t}}^T = \sum_{p\in P} w_p^{dom} t_p^T \tag{13}$$

These two global temporal features focus on domain-invariant temporal features via the proposed domain weight.

Temporal Domain Loss $\mathcal{L}_t$. Overall, bring $\dot{\mathtt{t}}^S$ and $\dot{\mathtt{t}}^\mathcal{T}$ into Eq. 10, we define the temporal domain loss function as:

$$\mathcal{L}_t = \frac{1}{M}\sum_{i=1}^{M} \left\| E\left(\dot{\mathtt{t}}_i^S\right) - E\left(\dot{\mathtt{t}}_i^T\right) \right\|_{\mathcal{H}}^2 \tag{14}$$

where M denotes feature dimensions.

3.4 Prediction Module

Referring to *RevGrad* [Pei *et al.*, 2018], we design a prediction module $\mathcal{M}_p$ to exact task-driven spatio-temporal features. In *RevGrad*, $\mathcal{M}_p$ is a classification task. In this paper, we design $\mathcal{M}_p$ as a FC network for crowd flow prediction.

Prediction Loss $\mathcal{L}_p$. Here, we utilize the Root Mean Squared Error (RMSE) as the prediction loss function, which is widely used in various spatio-temporal prediction studies.

$$\mathcal{L}_p = \mathrm{RMSE}(|Y - \hat{Y}|) \tag{15}$$

where Y is the ground truth and $\hat{Y}$ is the predicted result.

3.5 Overall Loss Function

Given the spatial domain loss $\mathcal{L}_s$, temporal domain loss $\mathcal{L}_t$, and prediction loss $\mathcal{L}_p$, we bridge them together as:

$$\mathcal{L}_{overall} = \beta\mathcal{L}_s + \gamma\mathcal{L}_t + \mathcal{L}_p \tag{16}$$

where β and γ control the weight of spatial/temporal aspects.

4 Experiments

Datasets	**NYCTaxi**	**NYCBike**	**CHIBike**	**BJTaxi**	**Chengdu**
# of Trips	About 120 million	About 10 million	About 3 million	/	About 7 million
Longitude	(-74.02, -73.93)	(-74.02, -73.93)	(-87.73, -87.55)	(116.36, 116.50)	(103.80, 104.30)
Latitude	(40.65, 40.79)	(40.65, 40.79)	(41.76, 42.01)	(39.85, 39.99)	(30.50, 30.80)
Time range	2015.1-2015.12	2015.1-2015.12	2015.1-2015.12	2015.11-2016.4	2016.11
# of Regions	16×16	16×16	16×16	32×32	32×32
Time interval	1 hour	1 hour	1 hour	30 minutes	1 hour

Table 1: Statistics of the Evaluated Datasets

Methods		NYCTaxi $\rightarrow$ *NYCBike*		CHIBike $\rightarrow$ *NYCBike*		NYCBike $\rightarrow$ *CHIBike*		BJTaxi $\rightarrow$ *Chengdu*	
		RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Deep-Learning	ConvLSTM	–	–	0.1058	0.0807	0.1040	0.0802	0.0756	0.0389
	STResNet	–	–	0.0649	0.0417	0.0896	0.0577	0.0541	0.0283
Fine-Tuned	ConvLSTM-FT	0.0613	0.0454	0.0505	0.0345	0.0219	0.0105	0.0361	0.0214
	STResNet-FT	0.0497	0.0292	0.0480	0.0281	0.0319	0.0176	0.0444	0.0230
	DCRNN-FT	0.0411	0.0292	0.0392	0.0279	0.0392	0.0276	–	–
Transfer-Learning	MetaST	0.0526	0.0349	0.0453	0.0226	0.1003	0.0656	0.0390	0.0275
	ST-DAAN	0.0381	0.0116	0.0381	0.0116	0.0234	0.0070	0.0909	0.0557
	STAN-No-SAAM	0.0344	0.0119	0.0371	0.0117	0.0215	0.0068	0.0357	0.0202
	STAN-No-TAAM	0.0345	0.0119	0.0362	0.0116	0.0215	0.0069	0.0368	0.0209
	STAN	**0.0327**	**0.0111**	**0.0361**	**0.0114**	**0.0196**	**0.0049**	**0.0345**	**0.0196**

Table 2: The Performance Comparison of All Methods

Datasets. We adopt five popular open-source urban crowd datasets, namely NYCTaxi, NYCBike, CHIBike, BJTaxi, and Chengdu, which are commonly used in related studies. Table 1 summarizes the statistics of the datasets. To validate the genericity of STAN for spatio-temporal knowledge transfer, we conduct experiments considering both intra-city and inter-city cases. Specifically, we select NYCTaxi as the source while NYCBike as target; CHIBike as the source while NYCBike as target, NYCBike as the source while CHIBike as target, and BJTaxi as the source while Chengdu as target. For the source city, we use 9 months of data (NYCTaxi, NYCBike, and CHIBike) and 3 months of data (BJTaxi) for training and the rest for testing. For the target city, we use 1 month of data (NYCBike and CHIBike) and 10 days of data (Chengdu) for training and the rest for testing. Note that we use NYCTaxi $\rightarrow$ NYCBike as default due to limited space.

Parameter Settings. First, we split the whole city into region/grid maps and cut the time dimension into non-overlapping time intervals (see Tab. 1). Then, we select 3 days of historical data where each day contains 9 time intervals. Further, we compute inflow/outflow of each region based on Eq. 1 and then normalize the flow data into [0, 1]. In terms of SAAM, the convolution kernel size is set to 3×3, and the domain discriminator contains two fully-connected layers with sizes are 64 and 1. In terms of TAAM, the length of LSTM and hidden feature size are set to 9 and 128. In terms of PM, two connected layers' sizes are 256 and 512. Then, we set β and γ as 0.1. As for model training, the epoch size, dropout, and learning rate are set to 32, 0.5, and $1 \times e^{-6}$. Besides, after 50 epochs, we reduce the learning rate to 0.9 times the original after every five epochs. STAN is implemented with Pytorch framework on GTX-3090 24G GPU.

Comparison Baselines. We compare STAN with: i) Deep learning methods. We choose ConvLSTM [Xingjian *et al.*, 2015] and STResNet [Zhang *et al.*, 2017] as they are remarkable deep crowd flow prediction models. Here, we only use target city data for model training. ii) Fine-tuned methods. These methods first pre-train the deep models on the source city and then fine-tune models on the target city. We select ConvLSTM, STResNet, and DCRNN [Li *et al.*, 2018b] and denote them as ConvLSTM-FT, STResNet-FT, and DCRNN-FT. iii) Transfer learning methods. MetaST [Yao *et al.*, 2019a] and ST-DAAN [Wang *et al.*, 2021] achieve state-of-the-art performance for across-city urban flow predictions. We adopt RMSE and MAE as **Evaluation Metrics**.

4.1 Effectiveness Study

As shown in Tab. 2, we compare different methods for both intra-city (i.e., NYC $\rightarrow$ *NYC*) and inter-city (i.e., CHI $\rightarrow$ *NYC*, NYC $\rightarrow$ *CHI*, and BJ $\rightarrow$ *Chengdu*) transfer cases. As expected, deep learning methods perform worst as the target city has scarce samples to guide model learning. Although the fine-tuned baselines perform better than non-transfer deep learning models, they are still inferior than transfer-learning methods. In addition, STAN consistently outperforms all baselines. For example, STAN achieves the reduction of RMSE by up to 15%, which verifies the benefits of our dynamic spatio-temporal modeling over static spatial-only correlation modeling of compared baselines.

Ablation Study. To evaluate SAAM and TAAM, we proceed to compare STAN with STAN-No-SAAM and STAN-No-TAAM of STAN that remove SAAM and TAAM, respectively. It shows that RMSE and MAE slightly increase but still lower than baselines in both experiments, which proves the effectiveness of designed SAAM and TAAM.

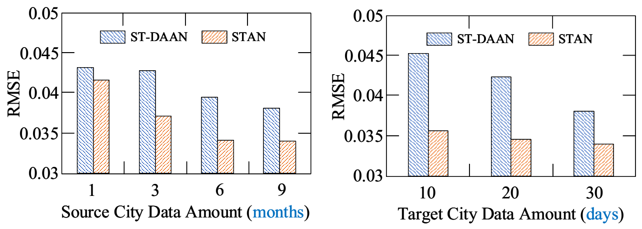

Figure 3: Performance vs. Data Amounts

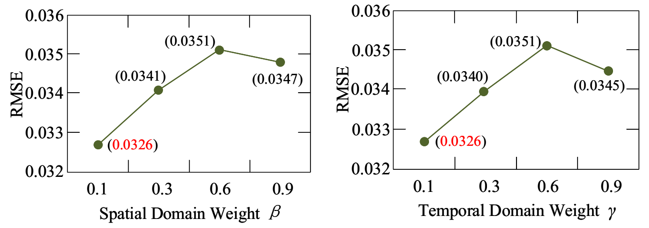

Figure 5: Tuning β and γ

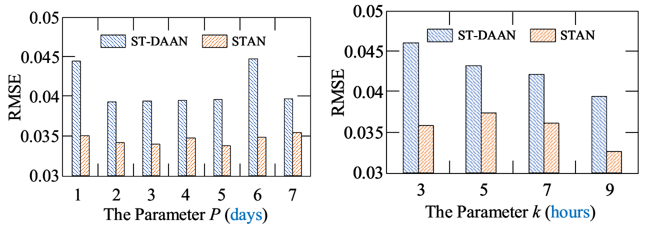

Figure 4: Sensitivity to Parameters P and k

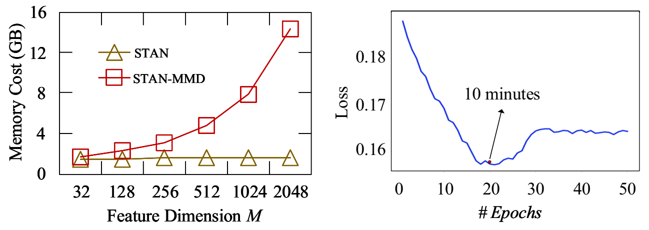

Figure 6: Efficiency Evaluation

4.2 Robustness Study

Effect of Data Amount. Here, we compare STAN with ST-DAAN, as the latter shows state-of-the-art performance. The results are shown in Fig. 3. First, STAN outperforms ST-DAAN under varying data samples of source and target cities. Second, more source/target data lead to better performance as more helpful knowledge can be transferred. Third, when the training data of the source/target city is smaller, the improvement of STAN is usually more significant than competitors, showing that STAN is more robust to data-scarce cases.

Sensitivity to P and k. As shown in Fig. 4, compared with ST-DAAN, STAN performs more stably with different P and k settings, further proving the superior robustness of STAN.

Tuning β and γ. Finally, we show how to set weight β and γ in Eq. 16. We set β (resp. γ) as fixed, and we change γ (resp. β) from 0.1 to 0.9. As shown in Fig. 5, reasonable settings for β and γ are 0.1. In addition, STAN always performs better than baselines with varying weight preference.

4.3 Efficiency Study

Last but not the least, we study the model efficiency. In terms of the feature embedding, as depicted in Fig. 6 (a), we replace the SAAM of STAN with MMD to embed the spatial features at each time step, resulting in STAN-MMD. As observed, STAN outperforms STAN-MMD by a large margin on memory occupy, as STAN adopts efficient adversarial classification to avoid massive feature computations.

In terms of the model training, the loss curve of the STAN training process is shown in Fig. 6 (b). One can see that the model converges quickly. Specifically, within around 20 epochs (corresponding to 10 minutes), the training loss first drops quickly and then becomes stable. This running time efficiency is acceptable in real-life deployments.

5 Related Work

Deep learning has become the most popular method for prediction studies [Guo *et al.*, 2021; Yuan *et al.*, 2021]. In terms of spatial modeling, CNN [Zhang *et al.*, 2016] based models split a city into equal-size regions. Then, convolution kernels can be utilized to capture the spatial correlations across regions. Recently, GCN-based models [Ye *et al.*, 2020; Bai *et al.*, 2020; Fang *et al.*, 2021] are proposed. GCN constructs a graph contains nodes and edges where a node may represent a road intersection and the edge represents the relations. Obviously, GCN-based models focus on road network modeling rather than urban modeling for citywide flow prediction. In terms of temporal modeling, RNNs [Xingjian *et al.*, 2015; Ye *et al.*, 2019] are widely applied. [Yuan and Li, 2021; Jiang *et al.*, 2021; Ye *et al.*, 2020] provide surveys on deep urban flow prediction. However, the deep learning approaches require massive training data and hard to generalize to support urban prediction with data-scare problems.

Transfer Learning. [Wang *et al.*, 2019] propose the first cross-city transfer learning for traffic prediction by matching similar regions between the source and target cities. [Yao *et al.*, 2019a] learn the period information from multiple cities for spatio-temporal learning. Obviously, they perform knowledge and prediction learning separately. To this end, [Wang *et al.*, 2021] perform urban knowledge transfer and predictive learning jointly, which also gains state-of-the-art performance. However, they all build the models via static and spatial-only feature transfer, while ignoring the fact inter-city correlations change dynamically across time.

Acknowledgments

This work was supported by the NSFC under Grants No. (62025206, 61972338, and 62102351), and also supported by AZFT. Yunjun Gao is the corresponding author of the work.

References

[Bai *et al.*, 2020] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In *NeurIPS*, 2020.

[Chen *et al.*, 2019] Min-Hung Chen, Zsolt Kira, Ghassan Alregib, Jaekwon Yoo, Ruxin Chen, and Jian Zheng. Temporal attentive alignment for large-scale video domain adaptation. In *ICCV*, pages 6320–6329, 2019.

[Fang *et al.*, 2021] Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. Mdtp: A multi-source deep traffic prediction framework over spatio-temporal trajectory data. *VLDB*, 14(8):1289–1297, 2021.

[Goodfellow *et al.*, 2014] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014.

[Guo *et al.*, 2021] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *TKDE*, 2021.

[Jiang *et al.*, 2021] Renhe Jiang, Du Yin, Zhaonan Wang, Yizhuo Wang, Jiewen Deng, Hangchen Liu, Zekun Cai, Jinliang Deng, Xuan Song, and Ryosuke Shibasaki. Dl-traff: Survey and benchmark of deep learning models for urban traffic prediction. In *CIKM*, pages 4515–4525, 2021.

[Li *et al.*, 2018a] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

[Li *et al.*, 2018b] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

[Liang *et al.*, 2019] Yuxuan Liang, Kun Ouyang, Lin Jing, Sijie Ruan, Ye Liu, Junbo Zhang, David S. Rosenblum, and Yu Zheng. Urbanfm: Inferring fine-grained urban flows. In *KDD*, pages 3132–3142, 2019.

[Liao *et al.*, 2018] Binbing Liao, Jingqing Zhang, Chao Wu, Douglas McIlwraith, Tong Chen, Shengwen Yang, Yike Guo, and Fei Wu. Sequence learning with auxiliary for traffic prediction. In *KDD*, pages 537–546, 2018.

[Long *et al.*, 2015] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with adaptation networks. In *ICML*, pages 97–105, 2015.

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

[Mazimpaka and Timpf, 2016] Jean Damascène Mazimpaka and Sabine Timpf. Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*, 2016(13):61–99, 2016.

[Pei *et al.*, 2018] Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. In *AAAI*, 2018.

[Tong *et al.*, 2017] Yongxin Tong, Lei Chen, and Cyrus Shahabi. Spatial crowdsourcing: Challenges, techniques, and applications. *VLDB*, 10(12):1988–1991, 2017.

[Wang *et al.*, 2018] Leye Wang, Bin Guo, and Qiang Yang. Smart city development with urban transfer learning. *Computer*, 51(12):32–41, 2018.

[Wang *et al.*, 2019] Leye Wang, Xu Geng, Xiaojuan Ma, Feng Liu, and Qiang Yang. Cross-city transfer learning for deep spatio-temporal prediction. In *IJCAI*, 2019.

[Wang *et al.*, 2021] Senzhang Wang, Hao Miao, Jiyue Li, and Jiannong Cao. Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks. *TITS*, 2021.

[Xingjian *et al.*, 2015] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NeurIPS*, 2015.

[Yao *et al.*, 2019a] Huaxiu Yao, Yiding Liu, Ying Wei, Xianfeng Tang, and Zhenhui Li. Learning from multiple cities: A meta-learning approach for spatial-temporal prediction. In *WWW*, pages 2181–2191, 2019.

[Yao *et al.*, 2019b] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *AAAI*, pages 5668–5675, 2019.

[Ye *et al.*, 2019] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, Xinran Tong, and Hui Xiong. Co-prediction of multiple transportation demands based on deep spatio-temporal neural network. In *KDD*, pages 305–313, 2019.

[Ye *et al.*, 2020] Jiexia Ye, Juanjuan Zhao, Kejiang Ye, and Chengzhong Xu. How to build a graph-based deep learning architecture in traffic domain: A survey. *TITS*, 2020.

[Yuan and Li, 2021] Haitao Yuan and Guoliang Li. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *DSE*, 6(1):63–85, 2021.

[Yuan *et al.*, 2020] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. Effective travel time estimation: When historical trajectories over road networks matter. In *SIGMOD*, pages 2135–2149, 2020.

[Yuan *et al.*, 2021] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. Effective joint prediction model for travel demands and traffic flows. In *ICDE*, pages 348–359, 2021.

[Zhang *et al.*, 2016] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. Dnn-based prediction model for spatio-temporal data. In *SIGSPATIAL*, 2016.

[Zhang *et al.*, 2017] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, 2017.

[Zhang *et al.*, 2020] Junbo Zhang, Yu Zheng, Junkai Sun, and Dekang Qi. Flow prediction in spatio-temporal networks based on multitask deep learning. *TKDE*, 32(3):468–478, 2020.

[Zheng, 2015] Yu Zheng. Trajectory data mining: an overview. *TIST*, 6(3):1–41, 2015.