Fine-Grained Urban Flow Prediction

Yuxuan Liang[1], Kun Ouyang[1], Junkai Sun[3], Yiwei Wang[1], Junbo Zhang[3,4], Yu Zheng[3,4,5], David S. Rosenblum[1,2], Roger Zimmermann[1]
[1]School of Computing, National University of Singapore, Singapore
[2]Department of Computer Science, George Mason University, VA, USA
[3]JD iCity, JD Technology, Beijing, China & JD Intelligent Cities Research, Beijing, China
[4]Artificial Intelligence Institute, Southwest Jiaotong University, Chengdu, China [5]Xidian University, Xi'an, China
{yuxliang,ouyangk,y-wang,rogerz,david}@comp.nus.edu.sg;{junkaisun,msjunbozhang,msyuzheng}@outlook.com

ABSTRACT

Urban flow prediction benefits smart cities in many aspects, such as traffic management and risk assessment. However, a critical prerequisite for these benefits is having fine-grained knowledge of the city. Thus, unlike previous works that are limited to coarse-grained data, we extend the horizon of urban flow prediction to fine granularity which raises specific challenges: 1) the predominance of inter-grid transitions observed in fine-grained data makes it more complicated to capture the spatial dependencies among grid cells at a global scale; 2) it is very challenging to learn the impact of external factors (e.g., weather) on a large number of grid cells separately. To address these two challenges, we present a Spatio-Temporal Relation Network (STRN) to predict fine-grained urban flows. First, a backbone network is used to learn high-level representations for each cell. Second, we present a Global Relation Module (GloNet) that captures global spatial dependencies much more efficiently compared to existing methods. Third, we design a Meta Learner that takes external factors and land functions (e.g., POI density) as inputs to produce meta knowledge and boost model performances. We conduct extensive experiments on two real-world datasets. The results show that STRN reduces the errors by 7.1% to 11.5% compared to the state-of-the-art method while using much fewer parameters. Moreover, a cloud-based system called UrbanFlow 3.0 has been deployed to show the practicality of our approach.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; • **Computing methodologies** → *Artificial intelligence*; *Neural networks*.

KEYWORDS

Urban flow prediction; spatio-temporal data; relational learning; convolution neural networks; urban computing.

ACM Reference Format:
Yuxuan Liang, Kun Ouyang, Junkai Sun, Yiwei Wang, Junbo Zhang, Yu Zheng, David S. Rosenblum and Roger Zimmermann. 2021. Fine-Grained Urban Flow Prediction. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3442381.3449792

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449792

1 INTRODUCTION

Accurately forecasting urban flows, such as predicting the total crowd flows entering and leaving each location (i.e., grid cell) of a city during a given time interval [40, 41], plays an essential role in smart city efforts. It can provide insights to the government for decision making, risk assessment, and traffic management. For example, by foreseeing that an overwhelming crowd will stream into a region ahead of time, the government can carry out traffic control, send warnings or even evacuate people for public safety.

One key property that must be considered in grid-based urban flow prediction is *spatio-temporal (ST) dependencies*: the future of a grid cell is conditioned on its previous readings as well as neighbors' histories. Moreover, urban flows are also impacted by *external factors* such as weather conditions and events. For example, heavy snow can sharply reduce traffic flows in many regions. To address these characteristics, many existing studies [6, 20, 36, 40–42] use convolutional neural networks (CNNs) as the backbone structure to extract spatially near and distant dependencies; the temporal dependencies (e.g., at the recent, daily and weekly levels) are captured using different sub-branches. Meanwhile, the influence of external factors is encoded by some manually-designed subnetworks.

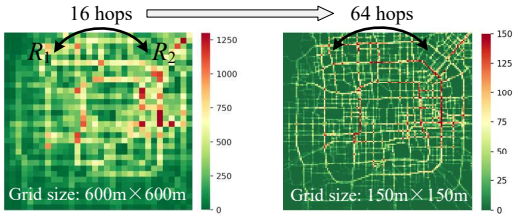
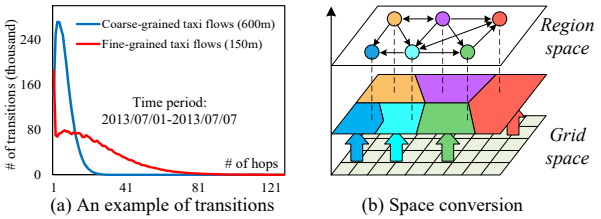


Figure 1: Coarse-grained vs. fine-grained urban flows.

In this paper, we focus on predicting urban flows at a *fine-grained* level, which is important yet unexplored in the community. Fine-grained flows can create exactness of the underlying dynamics of the city, encouraging better decision making. For instance, as shown in Figure 1, acquiring the traffic in a small area of interest with size 150m×150m can help allocate police resources more precisely while knowing that information at a district level with size 600m×600m is less useful. Notice that for a specific city, increasing the granularity (e.g., 600m→150m) is equivalent to obtaining higher resolution (e.g., 32×32→128×128). Thus, we use "high resolution" and "fine granularity" interchangeably. Although previous studies have shown promising results at coarse-grained levels (e.g., 32×32 Beijing [40]), their architectures are not suitable for predicting fine-grained urban flows due to the following specific challenges:



(a) An example of transitions (b) Space conversion

Figure 2: (a) compares the transition patterns from a certain week in Beijing. (b) shows grid- and region-based map segmentation, as well as space conversion between them.

1) *Global spatial dependencies.* Rasterizing the city with higher resolution reveals more details of urban mobility and, meanwhile, enlarges the distance (or hops) between two given grid cells. As shown in Figure 1(a), the number of hops between an office area (R1) and residence (R2) in Figure 1(b) becomes four times of that in Figure 1(a). This causes the statistics in Figure 2(a) where we can witness more long-range inter-grid communications (i.e., transitions with more hops) compared to that in the coarse-grained setting where short-range transitions often dominate. Hence, it becomes far more important to capture regional dependencies on a global scale in such fine-grained settings. In most of the existing studies [6, 40, 41], long-range spatial dependencies are captured by large receptive fields achieved by stacking many convolutional layers, where each layer captures only short-range dependencies at a local scale. Such naive repetition is computationally inefficient and causes optimization difficulties [7]. Though using dilation convolution [39] tends to alleviate this drawback to some extent, it fails to improve the predictive performance empirically (see Appendix A for more details). These facts demonstrate that simply increasing the receptive fields may not help. Recently, a new method called DeepSTN+ [20] attempted to capture the global spatial dependencies in *every layer* by explicitly modeling all pairwise relationships between grids. However, it indispensably induces a huge number of parameters with high computational costs. Hence, how to efficiently capture the global spatial dependencies remains unsolved.

2) *External factors & Land functions.* Previous studies like DeepST [41] and ST-ResNet [40] use subnetworks to map the effects of external factors onto each grid cell. Specifically, they stack several fully-connected layers upon the external factors: the former layers act as embedding layers to combine each factor and the final layer maps the short embeddings to high-dimensional features with the same shape as the flow map. However, as the resolution enhances to a fine-grained level, it will induce a large number of parameters proportional to the number of grid cells in the final layer. Furthermore, they ignore the influence of land functions such as POIs on traffic movements. To this end, DeepSTN+ [20] presents a new way to jointly consider the POIs information as well as the external factors. However, in DeepSTN+, external factors are used only to learn the weights of different kinds of POI features, while ignoring the significant difference of how external factors impact different grid cells. Thus, it is still challenging to learn the location-specific response to the external factors in fine-grained settings.

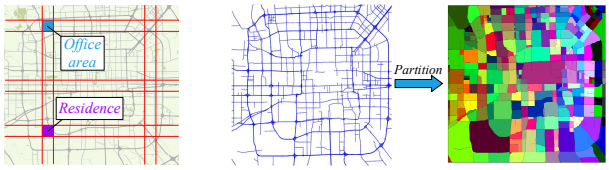
To address the above challenges, we present a Spatio-Temporal Relation Network (STRN) for fine-grained urban flow prediction. Similar to the previous and current state-of-the-art methods [20, 40], STRN follows the CPT (closeness, period and trend) paradigm to model the three types of temporal dependencies, and uses a CNN-based backbone to extract high-level ST features. To address the above challenges, we design two specific modules as follows.

Primarily, we introduce a new structure (GloNet) to capture the global spatial dependencies. We partition a city into N grid cells. Compared to DeepSTN+ that directly models all inter-grid correlations (totally N^2 correlations), we perform relational inference on a higher semantic level (i.e., region level) that is more friendly to capture such global relations. As depicted in Figure 2(b), we first perform a conversion from grid space to region space (M regions), and then infer the regional correlations globally by message passing. Since the region semantic changes over time, a new loss based on minimum cut theory enables the model to dynamically partition the map into irregular regions. Finally, we project the features back to the grid space and obtain global-aware features. In this way, our method only needs to model M^2 correlations among all region pairs[1], where typically $M \ll N$. Moreover, we present an original *Meta Learner* to produce the cell-specific responses to the time-evolving external factors based on matrix decomposition. Compared to DeepST and ST-ResNet, our Meta Learner not only considers the latent region functions but is also independent of the map resolution. Thus, it is more lightweight and practical in fine-grained settings. In contrast to DeepSTN+, our module can capture the cell-specific responses to the external factors and learn better representations. In summary, our contributions are four-fold:

- We devise a unified model that jointly considers the spatial, temporal and external relations for predicting fine-grained urban flows. A system has been deployed to show its practicality.
- We develop a GloNet structure that captures the global spatial dependencies in a more economical way than existing methods.
- We design an original Meta Learner to simultaneously learn the effects of external factors and land function.
- We conduct extensive experiments to evaluate our model on two real-world mobility datasets. Our model reduces the errors by 7.1%∼11.5% while using as few as less than 1% of the number of parameters required in the state-of-the-art method DeepSTN+.

2 FORMULATION

Definition 1 (Grid cell) As shown in Figure 3(a), we partition an area of interest (*e.g.*, a city) evenly into a $H \times W$ raster with totally $N = HW$ grid cells. Note that enlarging H or W indicates that we can obtain urban flow data with higher resolution.



(a) Map segmentation (b) Road network (c) Irregular regions

Figure 3: (a): Grid-based map segmentation. (b)-(c): We partition Beijing into irregular regions based on road networks.

[1] For example, $N = 128^2 = 16384$ while $M = 100$ in TaxiBJ+ dataset

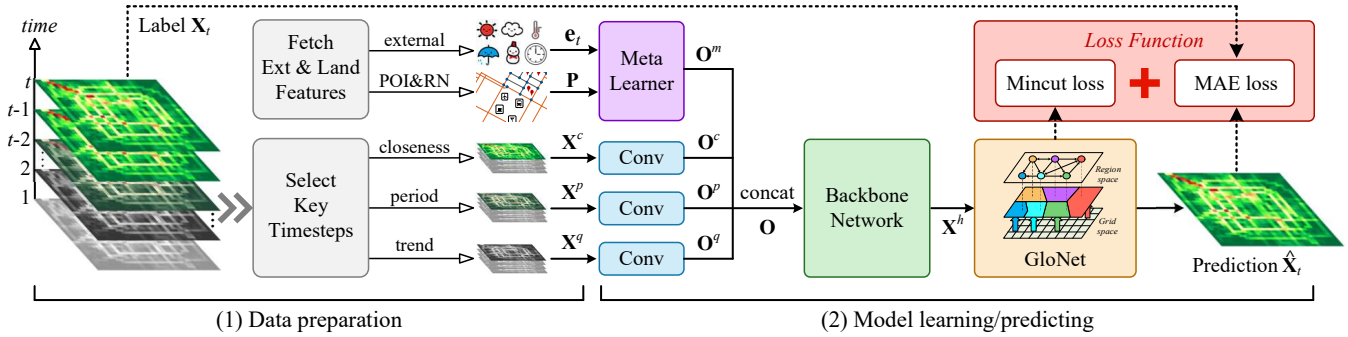


Figure 4: The framework of STRN. Closeness, period and trend are recent, daily and weekly patterns; Conv: convolution layer.

Definition 2 (Urban flow) The urban flows at a certain time t can be denoted as a 3D tensor $\mathbf{X}_t \in \mathbb{R}^{K\times H\times W}$, where K is the number of flow measurements (e.g., inflow/outflow). Each entry (k, h, w) denotes the value of the k-th measurement in the cell (h, w).

Definition 3 (Region) Land use and function endow different geographic semantics to urban areas that are bounded irregularly [43]. Figure 3(c) shows an example of irregular region segmentation based on road networks. It provides us with a more natural and semantic segmentation of urban spaces than the grid-based method. Assume that each region consists of many grid cells and we can thereby use a matrix $\mathbf{B} \in \mathbb{R}^{N\times M}$ to denote the assignment, where each element $b_{i,j}$ is the likelihood that grid cell i belongs to region j and M is the number of regions.

Definition 4 (External factors) Urban flow data have a strong correlation with external factors, such as weather conditions, time of day and events. We denote these external factors at a certain time step t as a vector $\mathbf{e}_t \in \mathbb{R}^{l_e}$, where l_e is the feature length.

Definition 5 (Land features) The category of POIs and their density in an urban grid cell indicate the land functions of the cell as well as the traffic patterns in this cell, therefore contributing to the urban flows of the grid cell [20]. Likewise, the structure of road networks (RNs) like the number of high-level road segments also provides a good complement to traffic modeling [17, 43]. Thus, we combine the land features including POIs and RNs of every cell, and denote them as $\mathbf{P} \in \mathbb{R}^{l_f\times H\times W}$, where l_f is the feature number.

Problem Statement Here, we define the problem of fine-grained urban flow prediction: Given the fine-grained historical observations of urban flows denoted as $\{\mathbf{X}_i | i = 1, 2, \cdots, t-1\}$, the corresponding external factors $\mathbf{e}_t$, and the land features $\mathbf{P}$, our target is to predict the urban flows at the future time step, denoted as $\mathbf{X}_t$.

3 METHODOLOGY

Figure 4 illustrates the framework of STRN, which consists of two major stages: *data preparation* and *model learning/predicting*. In the first stage, we first select the key timesteps (closeness, period and trend) to create the flow inputs, denoted as $\mathbf{X}^c$, $\mathbf{X}^p$ and $\mathbf{X}^q$, respectively. Meanwhile, we fetch the context of the external factors $\mathbf{e}_t$ and the land features $\mathbf{P}$. More details about the construction and dimensionality of these inputs are provided in Appendix B.

In the second stage, the prepared data are fed to learn the model, following a local to global paradigm. As shown in Figure 4, for each temporal sequence ($\mathbf{X}^c$, $\mathbf{X}^p$ and $\mathbf{X}^q$), we first use three non-shared convolutional layers to convert them to embeddings $\mathbf{O}^c, \mathbf{O}^p, \mathbf{O}^q$, each with D channels, i.e., they are all in $\mathbb{R}^{D\times H\times W}$. Meanwhile, we design a *Meta Learner* that takes the external factors and land features as inputs to learn the external impacts on each urban grid cell, where the learned representation $\mathbf{O}^m$ is of the same shape as $\mathbf{O}^c$. Next, we concatenate the three types of temporal features as well as the meta features, and feed the fusion result $\mathbf{O} \in \mathbb{R}^{4D\times H\times W}$ to the *backbone network* for feature extraction within its local receptive fields. This early fusion strategy allows different kinds of information to interact with each other in the backbone network. Once we obtain the extracted high-level features $\mathbf{X}^h \in \mathbb{R}^{C\times H\times W}$ at a local scale, we design a *GloNet* structure to capture the global spatial dependencies and generate the final predictions. Finally, we optimize the model weights by using a loss function including two parts: a Mincut loss for automatic region partition and a mean absolute error (MAE) loss for measuring the prediction errors.

3.1 Backbone Network

A powerful backbone network is crucial to urban flow prediction as it can help the model learn useful and discriminative features. For example, DeepST [41] provides the first deep learning-based solution to urban flow prediction by stacking a number of convolutional blocks for spatio-temporal feature extraction. As the network depth increases, DeepST will be hard to train due to the notorious vanishing gradient problem. To overcome this drawback, ResNet [7] is widely used as the backbone in the previous and current state-of-the-art [6, 20, 40] for urban flow prediction. However, they emphasize the dependencies in the spatial dimension and overlook the channel-wise information in the feature maps. In this paper, we employ the Squeeze-and-Excitation Networks (SENet) [10] to fuse both spatial and channel-wise information within small (i.e., local) receptive fields at each layer, which has proven to be effective in producing compacted and discriminative features of each grid cell. As shown in Figure 4, it takes the fusion result $\mathbf{O}$ as input and outputs the high-level representations of each cell. First, we use a convolutional layer to compress the dimension of input channels from $4D$ to C. Then, we stack F squeeze-and-excitation (SE) blocks [10] with C filters for feature extraction within the receptive field. Finally, a convolution layer is used to generate the output $\mathbf{X}^h$. The visualization of pipeline can be found in Appendix C.

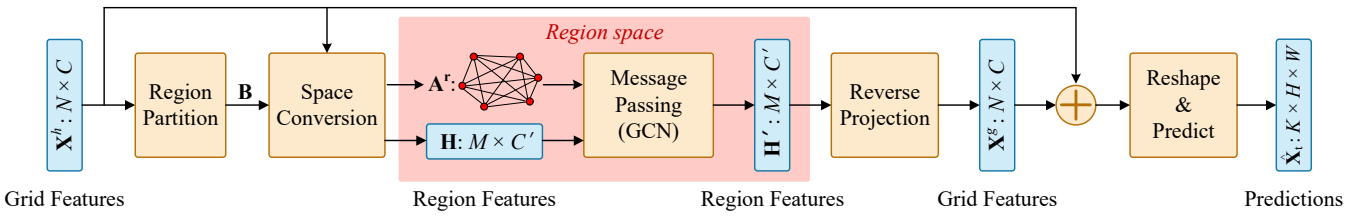


Figure 5: The pipeline of GloNet, where $N = HW$ and M are the number of grid cells and regions respectively.

3.2 Global Relation Module

After local feature extraction, we present a Global Relation Module (GloNet) to capture the global spatial dependencies in a more economical way than the previous attempts (e.g., DeepSTN+). Motivated by the relation networks [2, 14, 44] seizing relations between objects in images, we perform relational inference on a higher semantic level (i.e., region level) that is more friendly to capture global relations. Moreover, we design an unsupervised loss based on the minimum cut (Mincut) theory for region partition.

Figure 5 depicts the whole pipeline of GloNet. We first use the high-level features $\mathbf{X}^h$ to generate the assignment matrix $\mathbf{B}$ by a linear transformation. By referring to this matrix, GloNet then aggregates the grid-cell features into region space to obtain region features $\mathbf{H} \in \mathbb{R}^{M \times C'}$ and generate the connections (i.e., adjacency matrix $\mathbf{A}^r \in \mathbb{R}^{M \times M}$) between these regions. As the regions are connected in the form of a graph, we utilize Graph Convolution Networks (GCN) [13] to perform message passing on the region level. Once we obtain the global-aware features that are discriminative on the region level, the last step of GloNet is to project them back to the grid space and generate the final predictions.

3.2.1 Region Partition. Recall that the backbone network has produced a high-level abstraction of the flow history and external contexts. For convenience, we reshape this tensor to be $\mathbf{X}^h \in \mathbb{R}^{N \times C}$, where $N = HW$ is the number of grid cells. By doing this, each grid i can be represented by an embedding $\mathbf{x}_i^h \in \mathbb{R}^C$. Here, we aim to generate the grid-to-region assignment matrix $\mathbf{B} \in \mathbb{R}^{N \times M}$, where M is a hyperparameter that indicates the number of regions. Although we can perform a static region segmentation based on the road networks as mentioned in Section 2, it fails to capture the highly dynamic traffic conditions and the time-evolving external factors. To tackle this problem, we compute $\mathbf{B}$ based on the high-level representation $\mathbf{X}^h$ by means of a function δ, which maps each grid feature $\mathbf{x}_i^h$ into the i-th row of $\mathbf{B}$ as

$$\mathbf{B} = \text{softmax}\left(\delta(\mathbf{X}^h)\right), \tag{1}$$

where the softmax function guarantees the sum of each column equals to one. We parametrize δ as a feedforward neural network.

Inspired by the Mincut theory [1, 30] that aims at partitioning nodes into disjoint subsets by removing the minimum volume of edges, we view each region as a cluster containing many grid cells and regularize the assignment matrix by using a new loss. In other words, the network weights can be jointly optimized by minimizing the usual task-specific loss (e.g., MAE loss), as well as an unsupervised Mincut loss $\mathcal{L}_m$ composed of two terms:

$$\mathcal{L}_m = \underbrace{-\frac{\text{Tr}(\mathbf{B}^T \tilde{\mathbf{A}}^g \mathbf{B})}{\text{Tr}(\mathbf{B}^T \tilde{\mathbf{D}}^g \mathbf{B})}}_{\mathcal{L}_c} + \underbrace{\left\| \frac{\mathbf{B}^T\mathbf{B}}{\|\mathbf{B}^T\mathbf{B}\|_F} - \frac{\mathbf{I}_M}{\sqrt{M}} \right\|_F}_{\mathcal{L}_o}, \tag{2}$$

where $\|\cdot\|_F$ denotes the Frobenius norm; Tr is the trace of a matrix; $\mathbf{A}^g \in \mathbb{R}^{N \times N}$ is the adjacency matrix derived from the Euclidean structure and $\tilde{\mathbf{A}}^g$ is its normalization; $\tilde{\mathbf{D}}^g$ is the degree matrix of $\tilde{\mathbf{A}}^g$. $\mathbf{I}_M = \hat{\mathbf{B}}^T\hat{\mathbf{B}}$ is a rescaled clustering matrix, where $\hat{\mathbf{B}}$ assigns exactly N/M grid cells to each region. $\mathcal{L}_c \in [-1, 0]$ denotes the consistency loss that evaluates the mincut given by $\mathbf{B}$. Minimizing $\mathcal{L}_c$ enforces strongly connected grid cells to be grouped into the same region, while the other term $\mathcal{L}_o$ encourages the assignment to be orthogonal and the regions to be of similar size (see proof in Appendix D). Since the two terms in $\mathcal{L}_o$ have unitary norm, it is obvious that $0 \le \mathcal{L}_o \le 2$. Hence, $\mathcal{L}_o$ does not dominate over $\mathcal{L}_c$.

3.2.2 Space Conversion. Given the grid-cell features and the assignment matrix, we convert those grid-based embeddings to their regional counterparts $\mathbf{H} \in \mathbb{R}^{M \times C'}$ that are more friendly to capture global dependencies. Moreover, we need to find the connectivity $\mathbf{A}^r \in \mathbb{R}^{M \times M}$ between these regions. As people are capable of traveling to remote places in a short time period (*e.g.*, 30 minutes) in modern cities, we assume that all regions are mutually connected (*i.e.*, a complete digraph). Instead of using complex and time-consuming operations, we implement the space conversion by

$$\mathbf{H} = \mathbf{B}^\top \phi(\mathbf{X}^h), \quad \mathbf{A}^r = \mathbf{B}^\top \tilde{\mathbf{A}}^g \mathbf{B}, \tag{3}$$

where we generate the features of each region by directly aggregating the features of the corresponding cells that belong to this region; ϕ is a dense layer that compresses the dimension of embeddings from C to C' to avoid heavy computation; $\mathbf{A}^r$ is a symmetric matrix, whose entry $a_{i,i}$ is the total number of edges between the grid cells in the region i, while $a_{i,j}$ is the number of edges between region i and j. It can be seen easily that $\mathbf{A}^r$ comes from the numerator of L_c in Eq. 2, thus, the trace maximization yields regions with many internal grid-cell connections and weakly connected to each other.

3.2.3 Message Passing between Regions. After space conversion, we obtain a new graph where each node represents an irregular region and each edge models the interaction among two regions. To model the inter-region relationships, a natural idea is to use Graph Convolutional Networks (GCN) [13] to perform message passing between these regions based on $\mathbf{A}^r$. However, we notice that $\mathbf{A}^r$ is a diagonal-dominant matrix, describing a graph with self-loops much stronger than any other connection. As self-loops usually hamper the propagation across adjacent nodes in message passing

schemes [13], we compute a new adjacency matrix $\tilde{\mathbf{A}}^r \in \mathbb{R}^{M\times M}$ by zeroing the diagonal and applying degree normalization:

$$\hat{\mathbf{A}}^r = \mathbf{A}^r - \text{diag}\left(\mathbf{A}^r\right); \quad \tilde{\mathbf{A}}^r = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}^r\hat{\mathbf{D}}^{-\frac{1}{2}} \tag{4}$$

where $\hat{\mathbf{D}}$ denotes the degree matrix of $\hat{\mathbf{A}}^r$. Then, we employ a two-layer GCN for global relation inference on the region graph to generate the new region features $\mathbf{H}' \in \mathbb{R}^{M\times C'}$ as

$$\mathbf{H}' = f_{GCN}(\tilde{\mathbf{A}}^r, \mathbf{H}) = \tilde{\mathbf{A}}^r \,\text{ReLU}\left(\tilde{\mathbf{A}}^r\mathbf{H}\mathbf{W}_1\right)\mathbf{W}_2 \tag{5}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{C'\times C'}$ are learnable weights. By this, the regional information are passed through the graph to generate a global-aware representation for each region.

3.2.4 Reverse projection. Once we obtain the global-aware features $\mathbf{H}'$ from region space, the next step is to project them back to the original space. Similar to the step of space conversion, we can also use an assignment matrix for the reverse projection. Instead of using extra operations and introducing additional overhead, we reuse the $\mathbf{B} \in \mathbb{R}^{N\times M}$ to project the region features back to grid-cell features by a linear combination as follows:

$$\mathbf{X}^g = \mathbf{B}\theta(\mathbf{H}'), \tag{6}$$

where θ is a dense layer for dimension conversion from C' to C. The new grid-cell features $\mathbf{X}^g \in \mathbb{R}^{N\times C}$ are generated by aggregating their related region features, which is achieved by multiplying matrix $\theta(\mathbf{H}')$. Until now, we have performed a grid-region-grid transformation to learn the global-aware features in this module.

3.2.5 Reshape & Predict. Lastly, the global-aware discriminative features $\mathbf{X}^g$ are reshaped to $\mathbb{R}^{C\times H\times W}$ such that the output dimension can match the input dimension $\mathbf{X}^h$ forming a residual path, and fed to a convolution layer to produce the final predictions $\hat{\mathbf{X}}_t$. In practice, the matrix multiplication procedures for projection and reverse projection are both implemented by an 1×1 convolution layer since it supports high-speed parallelization. When computing the Mincut loss, we need to store and employ two matrices (i.e., $\tilde{\mathbf{A}}^g$ and $\tilde{\mathbf{D}}^g$) with shape N by N, which dramatically increases the memory cost and computational cost in devices (such as a GPU). To overcome this problem, we notice these two matrices are sparse and thereby implement Eq. 2 based on sparse matrix multiplication.

3.3 Meta Learner

As mentioned before, existing works like DeepST [41] and ST-ResNet [40] use fully-connected layers to encode the external factors for urban flow prediction. However, as the granularity becomes larger to a fine-grained setting, it will induce massive parameters in the last layer proportional to the number of grid cells (N). In addition, they do not consider the influence of land features on the traffic movements. To this end, DeepSTN+ [20] presents a new approach to jointly consider the POIs information as well as the external factors. They notice that POIs have varied temporal influences on flow maps, they thereby transform the external factors to influence the strength of POI. However, the external factors are applied to weight on different channels (i.e., categories) of POIs while ignoring the significant difference of how external factors impact different cells. Thus, how to learn the cell-specific responses to the external factors in the fine-grained settings remains a challenge.

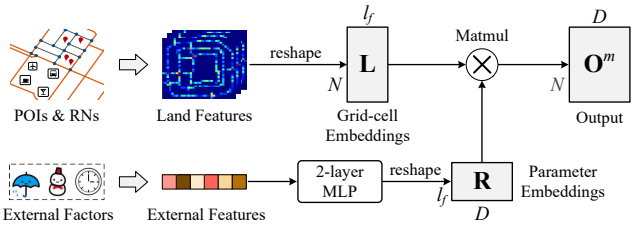


Figure 6: Pipeline of the proposed Meta Learner.

In general, grid cells with similar land functions will have similar responses to the external factors. Based on this observation, we design a novel Meta Learner to produce cell-specific responses to the external factors based on Matrix Factorization. Given the land features $\mathbf{P} \in \mathbb{R}^{l_f\times H\times W}$ and external features $\mathbf{e}_t \in \mathbb{R}^{l_e}$, we aim to compute the response of each grid cell by

$$\mathbf{O}^m = f_{ML}\left(\mathbf{P}, \mathbf{e}_t\right) \in \mathbb{R}^{D\times H\times W}. \tag{7}$$

Then, the output $\mathbf{O}^m$ will be early fused with the temporal information ($\mathbf{O}^c$, $\mathbf{O}^p$, $\mathbf{O}^q$) and fed to the backbone network.

Without introducing a large number of parameters in f_{ML}, we can reshape this target tensor as $\mathbb{R}^{N\times D}$ and decompose it into two matrices $\mathbf{L} \in \mathbb{R}^{N\times k}$ and $\mathbf{R} \in \mathbb{R}^{k\times D}$, which satisfies $\mathbf{O}^m = \mathbf{L}\mathbf{R}$. Motivated by a very recent study that learns specific predictors for each grid cell [27], we can view $\mathbf{L}$ as the grid-cell embeddings while $\mathbf{R}$ represents the parameter embeddings generated using $\mathbf{e}_t$ as meta knowledge. For simplicity, $\mathbf{L}$ is reshaped from the land features $\mathbf{P}$, where $l_f = k$ at this time. In this way, we can guarantee that cells with similar land functions will have similar responses to the external factors. For the parameter embeddings $\mathbf{R}$, we aim to make it change over time and influenced by the external factors, such as weather and time. Inspired by the meta learning approach for traffic prediction [27], we can use a two-layer MLP to generate it: the first layer transforms the external features from l_e to l_d, and the second layer further converts the dimension to be $l_f D$. Finally, we reshape the output and assign it to $\mathbf{R}$.

To help better understand the meta learner, Figure 6 further describes its whole pipeline. Compared to external components in ST-ResNet, as our meta learner is independent of the map resolution (only $l_e l_d + l_d l_f D$ parameters, which can be set far less than N), it is much more lightweight and practical in fine-grained settings. In contrast to DeepSTN+, our module can capture the cell-specific responses to the external factors and learn better representations.

3.4 Optimization

Our method provides an end-to-end solution from historical observations to fine-grained predictions, which is differentiable everywhere. Hence, the network can be trained through the back-propagation strategy and the Adam optimizer. To train our model, we aim to minimize the following loss function with two terms:

$$\mathcal{L} = \mathcal{L}_{MAE} + \alpha\mathcal{L}_m. \tag{8}$$

Here, α is a trade-off between these two losses while $\mathcal{L}_{MAE}$ is the pixel-wise Mean Absolute Error (MAE) for evaluating the errors between our prediction $\widehat{\mathbf{X}}_t$ and the corresponding ground truth $\mathbf{X}_t$.

4 EVALUATION

4.1 Experimental Settings

4.1.1 Datasets. We conduct our experiments on two real-world datasets, where TaxiBJ+ is the fine-grained version of TaxiBJ released by [41] and HappyValley is from [19].

- **TaxiBJ+**: This dataset sources from the trajectories of over 30,000 taxicabs in Beijing for four different time periods (i.e., P1 to P4). Since the taxi distributions and numbers are different in these four time periods, we evaluate our method over these periods separately. As shown in Figure 1(b), we crop the area of interest which contains most traffics, and rasterize this area into 128×128 uniform grid cells. The size of each cell is 150m×150m, which is much more fine-grained than that of the popular datasets like TaxiBJ [41] and MobileBJ [20]. We follow the previous studies [40, 41] to map the GPS points into different grid cells, each of which computes inflow and outflow volumes per half hour.
- **HappyValley**: It provides open access for the public to observe the hourly human density of a popular theme park in Beijing, allowing us to examine the model capacity in a much smaller area with a higher rate of global transitions. We partition the area into 50×100 grid cells, each of which is around 10m×10m. The human density is more distributed than the taxi flows in TaxiBJ+ since there are several popular play facilities with far more flows than their nearby regions.

The details of them are available in Table 1. We use the observations from the previous 12 time steps to predict the next step, and set the frames of period and trend patterns as 3. In each period of TaxiBJ+ and HappyValley, we recruit the first 70% as the training set, the next 20% as the validation set and the rest for the test set according to chronological order. We have removed the labels with all entries equal to zero. Besides, we use zero matrices as history for those items without sufficient precedent records. To speed up the convergence of STRN, a unique data normalization method [19, 25] for urban flow data is employed in our study.

Table 1: Description of the two datasets.

Dataset	TaxiBJ+	HappyValley
Data type	Taxi trip	Human flow
Resolution	128×128	50×100
Grid cell size	150m×150m	10m×10m
# Channels (K)	2 (inflow and outflow)	1 (population)
Sampling rate	30 minutes	1 hour
Time span	P1: 07/01/2013-10/31/2013 P2: 02/01/2014-06/30/2014 P3: 03/01/2015-06/30/2015 P4: 11/01/2015-03/31/2016	01/01/2018- 10/31/2018
External factors (meteorology, time and event)		
Weather	16 types	8 types
Temperature/℃	[-24.6,41.0]	[-24.6,41.0]
Wind speed/mph	[0,48.6]	[0,48.6]
# holidays	41	33
Ticket prize/RMB	/	[29.9,260]
Land features (POIs, road networks)		
# of POIs	651,016 (20 types)	/
# features of RNs	5	/

4.1.2 Baselines. We compare STRN with the following baselines:

- **ARIMA**: A well-known time series model.
- **VAR**: Vector Auto-Regressive (VAR) can capture the pairwise relationships among grid cells. To avoid parameter explosion, we use the history of nearby 7×7 cells as input.
- **LSTM** [9]: Long Short-Term Memory is a variant of vanilla RNN for learning long-term temporal dependencies.
- **DeepST** [41]: The first deep learning-based prediction model for grid-based spatio-Temporal data.
- **ST-ResNet** [40]: A ResNet-based method, which shows promising results on citywide crowd flows prediction.
- **ST-3DNet** [6]: It uses 3D convolution to capture the correlation of traffic data in both spatial and temporal dimensions.
- **ConvLSTM** [31]: It extends LSTM to have convolutional structures to better capture spatio-temporal correlations.
- **STDN** [36]: It employs CNNs and LSTMs to capture spatial and temporal correlations separately, and an attention mechanism to model long-term periodic temporal shifting.
- **DeepSTN+** [20]: The state-of-the-art method for urban flow prediction, which can capture long-term spatial dependencies as well as the effect of land functions.

We test different hyperparameters for them all, finding the best setting for each over the two datasets separately. For example, for LSTM and ConvLSTM, we tune the hidden dimensionality and different numbers of layers. Notice that the original design of DeepSTN+ would introduce too many parameters (over 10G) when it is directly applied to our fine-grained datasets. To avoid the parameter explosion, we set the pooling rate 32 for DeepSTN+.

4.1.3 Training Details & Hyperparameters. We implement STRN as well as the baselines by PyTorch 1.1 with one RTX 2080 Ti. The learning rate is halved every 50 epochs, starting from 0.001, and the batch size is 16. In the backbone network, the convolution layers and SE blocks use $C = \{32, 64, 96, 128\}$ filters with kernel size 3×3. For the number of stacked SE blocks, we conduct a grid search over $F = \{3, 6, 9, 12\}$. Moreover, we set $C' = 0.5C$ for feature reduction in the GloNet and tune the region number M. In the meta learner, the two fully-connected layers have $l_d = 32$ and $l_f = 25$ hidden units respectively and the embedding length D is 64.

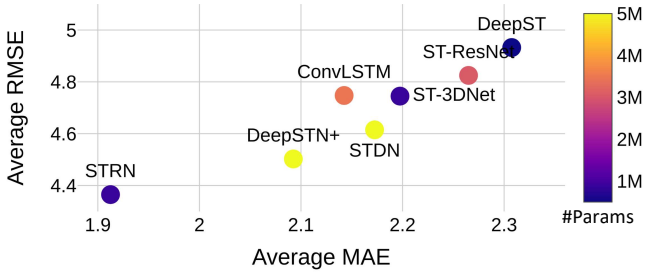


Figure 7: Model performance on TaxiBJ+ vs. # of parameters. The x-axis and y-axis indicate the average MAE and RMSE over the four time spans of TaxiBJ+. The color of each point denotes the number of parameters. It is worth noting that the number of parameters in DeepSTN+ largely exceeds 5M.

Table 2: Model comparison on TaxiBJ+, where the notation Δ indicates the reduction of MAE compared with DeepSTN+.

Method	#Param	P1			P2			P3			P4		
		MAE	Δ	RMSE	MAE	Δ	RMSE	MAE	Δ	RMSE	MAE	Δ	RMSE
ARIMA	<0.01M	2.46	+24.2%	5.37	2.91	+28.8%	6.34	3.02	+26.4%	6.55	2.08	+19.5%	4.47
VAR	<0.01M	2.41	+21.7%	5.21	2.84	+25.7%	6.18	2.92	+22.2%	6.38	2.06	+18.4%	4.35
LSTM	0.07M	2.27	+14.6%	5.04	2.68	+18.6%	6.03	2.78	+16.3%	6.21	1.88	+8.0%	4.20
ConvLSTM	3.45M	2.03	+2.5%	4.47	2.33	+3.1%	5.15	2.45	+2.5%	5.43	1.76	+1.1%	3.94
DeepST	0.46M	2.21	+11.6%	4.68	2.53	+11.9%	5.41	2.57	+7.5%	5.59	1.92	+10.3%	4.05
ST-ResNet	2.39M	2.14	+8.1%	4.58	2.48	+9.7%	5.29	2.61	+9.2%	5.55	1.83	+5.2%	3.88
ST-3DNet	0.89M	2.16	+9.1%	4.56	2.30	+1.8%	4.99	2.38	-0.4%	5.23	1.95	+12.1%	4.20
STDN	6.36M	2.08	+5.1%	4.40	2.32	+2.7%	4.98	2.44	+2.1%	5.23	1.85	+6.3%	3.85
DeepSTN+	0.27G	1.98	-	4.24	2.26	-	4.87	2.39	-	5.15	1.74	-	3.75
STRN	0.88M	**1.82**	**-8.1%**	**4.13**	**2.10**	**-7.1%**	**4.71**	**2.19**	**-8.4%**	**5.01**	**1.54**	**-11.5%**	**3.61**

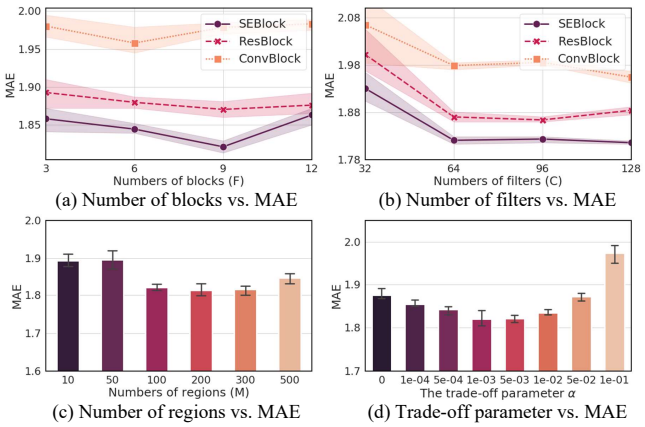
4.1.4 Evaluation Metrics. We measure model performances by two common metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). Smaller metrics indicate higher accuracy. For each dataset, we run each method 5 times and report the mean errors of each approach.

4.2 Results on TaxiBJ+

4.2.1 Model Comparison. We present the experimental results over TaxiBJ+ in Table 2. In particular, we report the result of STRN with $F = 9$, $C = 64$, $M = 100$ and $\alpha = 5e-3$ as our default setting. Our STRN *significantly* outperforms all competing baselines in terms of RMSE and MAE over all time periods according to the Students T-test at level 0.01. Compared to the state-of-the-art method (DeepSTN+), our approach reduces MAE by approximately 7.1% to 11.5% in each period, while using only 0.33% of the number of parameters required in DeepSTN+. The reasons are two-fold. First, our method captures the global spatial dependencies in a more efficient way, i.e., by GloNet structure. Second, STRN captures the cell-specific response to the external factors. Compared to ST-3DNet, our model surpasses it by a large margin while enjoying a similar parameter size. From Table 2, we can also have the following observations. ARIMA provides a lower bound of model performance. The results of VAR is much worse than the deep models due to its lower model capacity. Taking advantage of the RNN architecture, ConvLSTM and STDN slightly advance the previous state-of-the-art including DeepST, ST-ResNet and ST-3DNet. However, they both have overlooked the global spatial dependencies as well as the land functions, leading to the inferiority against DeepSTN+ and STRN. To further show the progress of our model, we also analyze the average performance of each model against the parameter size over TaxiBJ+ in Figure 7, which clearly shows that our method wins in both lightweights and effectiveness. Next, we study the effectiveness of each model component over P1 of TaxiBJ+.

4.2.2 Effects of Backbone Network. In general, a strong backbone network can significantly improve the capability of feature extraction. Here, we compare STRN with its variants using the residual block (ResBlock) [7] or the standrard convolution block (ConvBlock) as the backbone. We also attempt different numbers of these blocks (F) and filters (C) to study the effects of them. Specifically, we set $C = 64$ to explore the effects of F and let $F = 9$ to study the effects of C. As depicted in Figure 8(a)-(b), SEBlock and ResBlock outperform ConvBlock by a very large margin in all cases, which demonstrates the power of residual learning. Since SEBlock considers the channel-wise information in the feature maps, it reduces MAE by approximately 0.05 compared to ResBlock. Figure 8(a) shows that our model achieves the best performance in test set when $F = 9$. Though increasing C to 128 can bring slight improvement according to Figure 8(b), it will induce much higher computational costs. Thus, we choose $C = 64$ as our default setting.

4.2.3 Effects of GloNet. Table 3 shows the comparison between STRN and its variants over P1. It can be seen easily that the integration of GloNet improves the model performance from 1.93 to 1.82 while only using very few extra parameters (0.05M), since GloNet enables our model to capture global spatial dependencies more efficiently. Moreover, GloNet with dynamic region partition (Mincut-based) can outperform static partition (road-network-based in Figure 3(b)) according to the comparison between STRN and STRN w/o dynamic. There are two hyperparameters in this module: the number of regions M and the parameter α for balancing $\mathcal{L}_{MAE}$ and $\mathcal{L}_m$. As shown in Figure 8(c), the performance degrades when M is small such as 10 and 50, since it is hard to aggregate over ten thousand grid cells into such few regions. We also notice that increasing



(a) Number of blocks vs. MAE (b) Number of filters vs. MAE

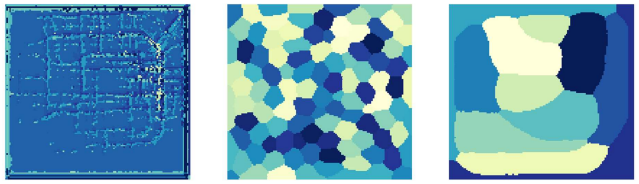
(c) Number of regions vs. MAE (d) Trade-off parameter vs. MAE

Figure 8: Effects of hyperparameters in STRN over P1.

Table 3: Results of different variants over P1 of TaxiBJ+, where ML denotes Meta Learner and w/o indicates without; Δ indicates the reduction of MAE compared with Backbone.

Variant	#Param	MAE	Δ	RMSE
Backbone	0.82M	1.97	-	4.22
STRN w/o GloNet	0.83M	1.93	-2.0%	4.20
STRN w/o ML	0.87M	1.85	-6.1%	4.15
STRN w/o dynamic	0.88M	1.87	-5.1%	4.16
STRN	0.88M	**1.82**	**-7.6%**	**4.13**

M does not give significant gain and instead slow down the training phase. Therefore, we select $M = 100$ in our STRN. Several examples of the partition results at different epochs are shown in Figure 9. At the beginning of the training phase, most of the grid cells are clustered into one region like Figure 9(a). After training, we can obtain a discriminative partition, see Figure 9(b). In contrast, we also present a counterpart when $M = 10$ in Figure 9(c).

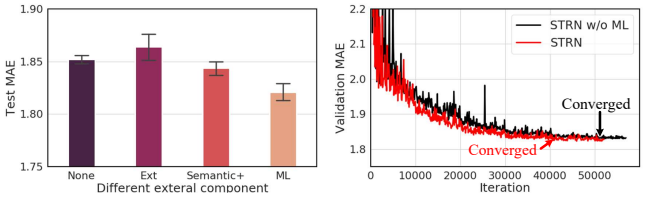


(a) Results at epoch 1 ($M = 100$) (b) Results at epoch 150 ($M = 100$) (c) Results at epoch 150 ($M = 10$)

Figure 9: Examples of region partition, where different colors indicate the regions with highest likelihood to each cell.

To study the effects of α, we try different scales of α from 1e-4 to 1e-1. We also set $\alpha = 0$ (i.e., no Mincut loss) for comparison. The results are given in Figure 8(d), from which we can see that increasing α to a large level (over 0.05) dilutes the effect of $\mathcal{L}_{MAE}$, leading to more predictive biases. Using very small α also degrades the performances as it cannot effectively supervise the region partition. Besides, we achieve the lowest MAE when α is around 1e-3. Note that we choose $\alpha = 5e-3$ as our default setting due to its best performance on the validation set rather than the test set.

4.2.4 Effects of Meta Learner. As a practical component of STRN, this module provides additional prior knowledge (external factors and land functions) to boost the predictive performance. The comparison between STRN and STRN w/o ML in Table 3 shows the effectiveness of Meta Learner (ML). To further investigate it, we also compare it with the modules for similar targets from other models. As depicted in Figure 10(a), "Ext" is the variant of STRN replacing ML by the external module from ST-ResNet, and "Semantic+" is the variant of STRN using the semantic module from DeepSTN+. Besides, "None" denotes STRN w/o ML as a baseline. Compared to None, Ext degrades the model performance since it suffers from overfitting problems caused by its large parameter size in our fine-grained settings. Semantic+ can bring slight improvements but less than our Meta Learner because Semantic+ has overlooked the cell-specific response to the external factors. Moreover, Figure 10(b) presents the validation curves during the training phase over P1. Remarkably, Meta Learner not only accelerates the training process but also makes it more stable. Specifically, STRN converges at iteration 40700 (epoch 201) while STRN w/o ML early-stops at iteration 50900 (epoch 250). The reason is that our Meta Learner successfully encodes the external factors and land functions and provides them as prior knowledge of the model.



(a) Meta Learner vs. Other models (b) The curves of validation MAE

Figure 10: Effects of Meta Learner over P1 of TaxiBJ+.

4.3 Results on HappyValley

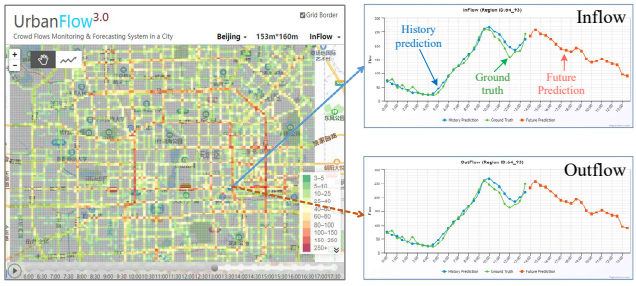
We also evaluate STRN on HappyValley with skewed human flow distribution, where only a few grid cells with popular play facilities contain dense human flow. Since the samples in HappyValley are fewer than TaxiBJ+, we reduce the network depth for each model to avoid overfitting. Noticing that there are no available land features in this dataset, we remove the corresponding components of DeepSTN+ and STRN. Table 4 presents the results over HappyValley, from which we know that: 1) STRN with such few parameters can present the state-of-the-art performance over both metrics, which reveals that our model is practical in real-world systems. It brings around 7.6% improvements against DeepSTN+ in terms of MAE, where the improvements are significant according to the Student T-test at level 0.01. 2) The fact that STRN and DeepSTN+ significantly outperform ST-ResNet over MAE verifies the necessity of capturing the global relations in such a small area. The enhancement of STRN beyond its variant (w/o GloNet) further verifies this point. 3) Unlike the results in TaxiBJ, ConvLSTM performs much better on MAE and slightly better on RMSE than most of the baselines, which validates the importance of the LSTM structure for explicitly modeling the temporal patterns. 4) Though CNN-based baselines (DeepST, ST-ResNet and ST-3DNet) achieve promising results on RMSE, they perform much poorer in terms of MAE, demonstrating the skewed distribution of this dataset.

Table 4: Results on HappyValley dataset. We omit ARIMA, VAR and LSTM due to their poor performances.

Model	#Param	MAE	Δ	RMSE
DeepST	0.26M	2.21	+10.6%	7.98
ST-ResNet	0.63M	2.20	+10.1%	7.91
ST-3DNet	0.52M	2.16	+8.1%	7.95
ConvLSTM	0.63M	1.98	-1.0%	7.86
STDN	0.97M	2.05	+2.5%	7.89
DeepSTN+	15.69M	2.00	-	7.88
STRN w/o GloNet	0.36M	1.97	-1.5%	7.90
STRN	0.37M	**1.85**	**-7.6%**	**7.80**

5 PRACTICALITY

To further verify its practicality, we deploy a cloud-based system called UrbanFlow$^{3.0}$ for monitoring the real-time urban flows and forecasting future flows in Beijing. Specifically, it supports us to predict the urban flows at a fine-grained level by using our STRN as the bedrock model. Figure 11(a) presents the main interface of it. Each grid cell denotes an area of interest with 150m×150m size. The color of each cell indicates its flow density, where "red" means dense and "green" means sparse. A user can click any grid cell on the interface to see the flow details like Figure 11(b), including the historical ground truth as well as the previous and future prediction results. Furthermore, users can watch the movie-style heatmaps such as Figure 11(c) by clicking the "play button" at the bottom left of the main interface in Figure 11(a). In summary, the benefits from *application layers* lie in two aspects. First, our model is scalable to high-resolution flow data. For example, as the data granularity increases to 256*256, DeepSTN+ will induce 40G parameters and is no longer feasible in online deployment, but our model still works well. Second, the lightweight property is crucial to mobile deployment, e.g., our model can be easily integrated into mobile apps such as Google Maps for urban flow prediction.



(a) Interface of our system (b) Inflow/Outflow of a cell

(c) Flow heatmap in the history and in future

Figure 11: Interface of UrbanFlow$^{3.0}$.

6 RELATED WORK

6.1 Grid-based Urban Flow Prediction

With recent advances in ubiquitous devices [33], urban flow prediction has been an appealing domain. Several pioneering studies [3, 32] were made to forecast massive individual mobility traces. Apart from analyzing mobility on an individual level, many data-driven models were proposed for city-scale traffic prediction by aggregating flows into corresponding regions [8, 16]. However, these models highly depend on hand-crafted features, which require extensive expert efforts. Thereafter, researchers started to use deep learning for urban flow prediction. [41] presented the first CNN-based architecture to forecast the urban flows. Inspired by residual learning [7], they further presented ST-ResNet [40] to collectively predict inflow and outflow of crowds in every city grid cell. However, ST-ResNet simply treats information in adjacent time intervals as multiple channels, losing the temporal information in the intermediate layers. To tackle this issue, [6] first presented a framework based on 3D convolution to jointly model the ST correlations. Besides, many studies [21–23, 31, 36, 37] combined the CNNs with RNNs to capture the dynamic ST dependencies. In addition, [4, 42] exploited the flow transitions between regions as auxiliary features to boost the predictive performance. One problem is that, these studies are inefficient to capture the global spatial dependencies, since they rely on large receptive fields achieved by stacking many convolution layers. Therefore, [20] proposed DeepSTN+ to directly model the relationships between two arbitrary grid cells, showing state-of-the-art performances in urban flow prediction.

However, none of them can be directly applied to fine-grained data as capturing long-range dependencies would incur a huge increase in the network depth and parameters. Instead, our STRN is able to capture the global dependencies at a much lower memory cost and provides superior performances. In addition, STRN can capture the cell-specific responses to external factors in the fine-grained settings. Besides, there are several studies that aim to predict other types of ST data in a fine-grained level [28, 34].

6.2 Deep Learning for ST Prediction

Recently, deep learning models have been the dominant class of spatio-temporal forecasting. For instance, CNNs have been widely used as a basic building block for capturing spatial dependencies in grid-based ST data, such as urban flows [5, 20, 40], abnormal events [11, 12] and taxi demand [37]. Another paradigm is to use RNNs and attention mechanisms to model the temporal dependencies [18, 26, 36]. Recently, graph convolution networks (GCNs) [13] have attracted many interests by virtue of its capability of modeling non-euclidean structure. [15] devised a diffusion convolution to model the spatial dependencies between traffic sensors and integrate it with RNNs for traffic forecasting. Later works [35, 38] combined GCNs with temporal convolutions for parallel training. However, GCNs *cannot* be directly applied to grid-based urban flow prediction since there is no explicit graph structure. In this study, we use GCNs to perform global relation inference in the region space.

6.3 Relation Networks

Relation networks enable us to resolve complexities within spatio-temporal data by capturing and exploiting the underlying semantic structures, which overcomes the pitfalls of short-range dependencies in traditional CNNs. In this strand, [29] presented the first effort in modeling inter-region relations in images using a simple relation network. In the temporal domain, [44] captured frame-level correlations in video streams by a temporal relation network. Moving from the limited competence of processing only regular data (*i.e.*, pixel-level), recently, [2, 14] developed novel relation network modules to reason on semantic-level relations by employing a latent semantic graph structure. Motivated by the ubiquitous dependencies between regions in ST domains [24], we present the first attempt to infer the relations between different urban locations in both grid-cell and region level based on the Mincut theory.

7 CONCLUSION AND FUTURE WORK

In this paper, we present a Spatio-Temporal Relation Network for fine-grained urban flow prediction. Our model can jointly learn spatial (local and global) and temporal (CPT) dependencies as well as external relations (e.g. weather and POIs). Unlike previous methods, we avoid to model all inefficient operations (e.g., for cell-specific responses or pairwise transitions), which allows us immune to the change of granularity and thus avoid parameter blowing up. We evaluate our model on two real-world fine-grained datasets, where our model can achieve state-of-the-art performances while using very few parameters. In the future, we plan to extend our flow prediction system to an online learning framework.

ACKNOWLEDGMENTS

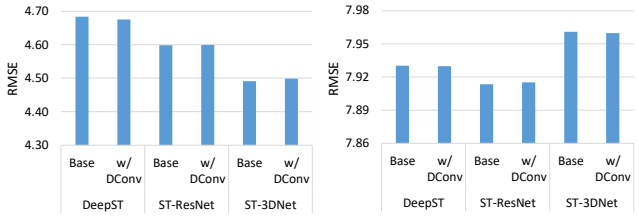
This study is mainly supported by Singapore Ministry of Education Academic Research Fund Tier 2 under MOE's official grant number MOE2018-T2-1-103, also supported by the National Key R&D Program of China (2019YFB2103201), the NSFC (No. 62076191, 72061127001), and the Beijing Nova Program (Z201100006820053). We thank all reviewers for their advice in improving this paper.

REFERENCES

[1] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2019. Mincut pooling in graph neural networks. *arXiv preprint arXiv:1907.00481* (2019).
[2] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. 2019. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.
[3] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. 2015. CityMomentum: an online approach for crowd behavior prediction at a citywide level. In *UbiComp*. 559–569.
[4] Jie Feng, Ziqian Lin, Tong Xia, Funing Sun, Diansheng Guo, and Yong Li. [n.d.]. A Sequential Convolution Network for Population Flow Prediction with Explicitly Correlation Modelling. In *IJCAI*.
[5] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention Based Spatial-Temporal Graph Convolutional Networks for Traffic Flow Forecasting. In *AAAI*.
[6] Shengnan Guo, Youfang Lin, Shijie Li, Zhaoming Chen, and Huaiyu Wan. 2019. Deep Spatial-Temporal 3D Convolutional Neural Networks for Traffic Data Forecasting. *IEEE Transactions on Intelligent Transportation Systems* (2019).
[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[8] Minh X Hoang, Yu Zheng, and Ambuj K Singh. 2016. FCCF: forecasting citywide crowd flows based on big data. In *SIGSPATIAL*. 1–10.
[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[10] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7132–7141.
[11] Chao Huang, Chuxu Zhang, Jiashu Zhao, Xian Wu, Dawei Yin, and Nitesh Chawla. 2019. Mist: A multiview and multimodal spatial-temporal learning framework for citywide abnormal event forecasting. In *The World Wide Web Conference*. 717–728.
[12] Chao Huang, Junbo Zhang, Yu Zheng, and Nitesh V Chawla. 2018. DeepCrime: Attentive hierarchical recurrent networks for crime prediction. In *ACM International Conference on Information and Knowledge Management*. 1423–1432.
[13] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[14] Yin Li and Abhinav Gupta. 2018. Beyond grids: Learning graph representations for visual recognition. In *Advances in Neural Information Processing Systems*.
[15] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.
[16] Yexin Li, Yu Zheng, Huichu Zhang, and Lei Chen. 2015. Traffic prediction in a bike-sharing system. In *SIGSPATIAL*.
[17] Yuxuan Liang, Zhongyuan Jiang, and Yu Zheng. 2017. Inferring traffic cascading patterns. In *SIGSPATIAL*. 1–10.
[18] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. GeoMAN: Multi-level Attention Networks for Geo-sensory Time Series Prediction. In *IJCAI*.
[19] Yuxuan Liang, Kun Ouyang, Lin Jing, Sijie Ruan, Ye Liu, Junbo Zhang, David S. Rosenblum, and Yu Zheng. 2019. UrbanFM: Inferring Fine-Grained Urban Flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 3132–3142.
[20] Ziqian Lin, Jie Feng, Ziyang Lu, Yong Li, and Depeng Jin. 2019. Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In *AAAI*, Vol. 33. 1020–1027.
[21] Lingbo Liu, Zhilin Qiu, Guanbin Li, Qing Wang, Wanli Ouyang, and Liang Lin. 2019. Contextualized spatial–temporal network for taxi origin-destination demand prediction. *IEEE Transactions on Intelligent Transportation Systems* 20, 10 (2019), 3875–3887.
[22] Lingbo Liu, Ruimao Zhang, Jiefeng Peng, Guanbin Li, Bowen Du, and Liang Lin. 2018. Attentive crowd flow machines. In *Proceedings of the 26th ACM international conference on Multimedia*. 1553–1561.
[23] Lingbo Liu, Jiajie Zhen, Guanbin Li, Geng Zhan, Zhaocheng He, Bowen Du, and Liang Lin. 2020. Dynamic spatial-temporal representation learning for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* (2020).
[24] Philippe Muller. 2002. Topological spatio–temporal reasoning and representation. *Computational Intelligence* 18, 3 (2002), 420–450.
[25] K. Ouyang, Y. Liang, Y. Liu, Z. Tong, S. Ruan, D. Rosenblum, and Y. Zheng. 2020. Fine-Grained Urban Flow Inference. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–1.
[26] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1720–1730.
[27] Zheyi Pan, Zhaoyuan Wang, Weifeng Wang, Yong Yu, Junbo Zhang, and Yu Zheng. 2019. Matrix Factorization for Spatio-Temporal Neural Networks with Applications to Urban Flow Prediction. In *CIKM*. 2683–2691.
[28] Zhongang Qi, Tianchun Wang, Guojie Song, Weisong Hu, Xi Li, and Zhongfei Zhang. 2018. Deep air learning: Interpolation, prediction, and feature analysis of fine-grained air quality. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2285–2297.
[29] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *NeurIPS*. 4967–4976.
[30] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.
[31] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*. 802–810.
[32] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. 2014. Prediction of human emergency behavior and their mobility following large-scale disaster. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 5–14.
[33] My T Thai, Weili Wu, and Hui Xiong. 2016. *Big Data in Complex and Social Networks*. CRC Press.
[34] Xian Wu, Chao Huang, Chuxu Zhang, and Nitesh V Chawla. 2020. Hierarchically Structured Transformer Networks for Fine-Grained Spatial Event Forecasting. In *Proceedings of The Web Conference 2020*. 2320–2330.
[35] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *IJCAI*.
[36] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *AAAI*, Vol. 33. 5668–5675.
[37] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Li Zhenhui. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In *AAAI*.
[38] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *IJCAI*.
[39] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* (2015).
[40] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In *AAAI*. 1655–1661.
[41] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. 2016. DNN-based prediction model for spatio-temporal data. In *SIGSPATIAL*. 1–4.
[42] Junbo Zhang, Yu Zheng, Junkai Sun, and Dekang Qi. 2019. Flow prediction in spatio-temporal networks based on multitask deep learning. *IEEE Transactions on Knowledge and Data Engineering* (2019).
[43] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 1–55.
[44] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. 2018. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 803–818.

A EFFECTS OF DILATION CONVOLUTION

To validate the effects of dilation convolution for capturing the global spatial dependencies in fine-grained urban flow prediction, we integrate the existing CNN-based methods (DeepST [41], ST-ResNet [40] and ST-3DNet [6]) with it. Figure 12 presents the variant comparison in terms of Root Mean Square Error (RMSE) on two real-world datasets (the details of these datasets are provided in Section 4.1.1), where base denotes the base model while base w/ DConv means integrating the dilation convolution. We can easily observe that the integration of dilation convolution cannot bring consistent improvements over both datasets.



(a) Results on P1 of TaxiBJ+ (b) Results on HappyValley

Figure 12: The effects of dilation convolution.

B DETAILS OF DATA PREPARATION

The urban flows in a grid cell is affected by recent time intervals, both near and far. Thus, we use the CPT paradigm (closeness, period and trend) [40] to consider the three types of temporal dependencies. For a specific future time t as the prediction target, we select the corresponding recent, daily and weekly timesteps as the key timesteps to construct the input sequences $\mathbf{X}^c$, $\mathbf{X}^p$ and $\mathbf{X}^q$ as:

$$\mathbf{X}^c = \left[\mathbf{X}_{t-l_c}, \mathbf{X}_{t-(l_c-1)}, \cdots, \mathbf{X}_{t-1}\right] \in \mathbb{R}^{Kl_c \times H \times W},$$
$$\mathbf{X}^p = \left[\mathbf{X}_{t-l_p \cdot p}, \mathbf{X}_{t-(l_p-1)\cdot p}, \cdots, \mathbf{X}_{t-p}\right] \in \mathbb{R}^{Kl_p \times H \times W},$$
$$\mathbf{X}^q = \left[\mathbf{X}_{t-l_q \cdot q}, \mathbf{X}_{t-(l_q-1)\cdot q}, \cdots, \mathbf{X}_{t-q}\right] \in \mathbb{R}^{Kl_q \times H \times W},$$

where l_c, l_p, l_q are the input length of closeness, period and trend respectively. To better illustrate it, we have shown an example of the input preprocessing in Figure 13.

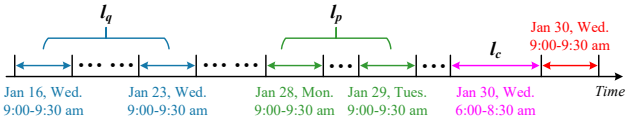
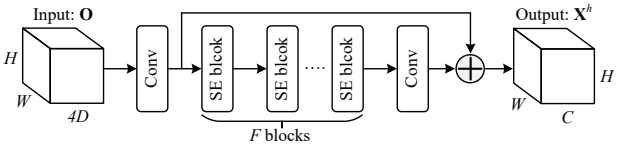


Figure 13: An example of input construction.

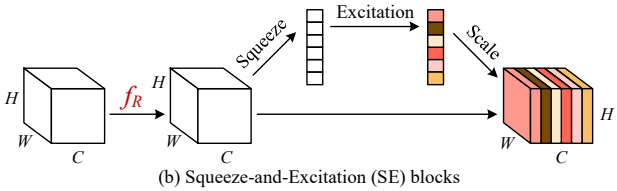
Besides, we fetch the external factors $\mathbf{e}_t$ as well as land features $\mathbf{P}$. The external factors can be classified into two groups: continuous and categorical. Continuous features like humidity and temperature can be directly concatenated to be a vector $\mathbf{e}_t^{con}$. Categorical features like weather (e.g, sunny, rainy) and events are transformed into low-dimensional embeddings by feeding them into different embedding layers separately, and then concatenate those embeddings to construct the categorical vector $\mathbf{e}_t^{cat}$. Finally, we concatenate them to obtain the external features at time interval t as $\mathbf{e}_t = [\mathbf{e}_t^{con}; \mathbf{e}_t^{cat}]$.

C DETAILS OF BACKBONE NETWORK

Figure 14(a) presents the pipeline of backbone network as detailed in Section 3.1. Figure 14(b) further shows the details of SE block, which contains three steps: 1) a function f_R for feature transformation; 2) a squeeze operation to squeeze global spatial information into a channel descriptor; 3) an excitation operation to fully capture the channel-wise dependencies: it first computes the attention coefficients over each channel via a feedforward network followed by a sigmoid function, and then rescales the channels of original inputs by these weights. In our study, f_R is a residual block [7] and the squeeze operation is implemented by global average pooling. In summary, the SE-based backbone allows our model to learn better representations for each grid cell locally within its receptive fields.



(a) Pipeline of the backbone network



(b) Squeeze-and-Excitation (SE) blocks

Figure 14: Pipeline of backbone network, where the number of filters in each convolution layer and SE block is C.

D EXPLANATORY OF MINCUT LOSS

Here, we introduce the insight behind using Mincut theory to conduct region partition in our study. Recall that the unsupervised Mincut loss $\mathcal{L}_m$ is defined as:

$$\mathcal{L}_m = \underbrace{-\frac{\mathrm{Tr}(\mathbf{B}^T\tilde{\mathbf{A}}^g\mathbf{B})}{\mathrm{Tr}(\mathbf{B}^T\tilde{\mathbf{D}}^g\mathbf{B})}}_{\mathcal{L}_c} + \underbrace{\left\| \frac{\mathbf{B}^T\mathbf{B}}{\|\mathbf{B}^T\mathbf{B}\|_F} - \frac{\mathbf{I}_M}{\sqrt{M}} \right\|_F}_{\mathcal{L}_o},$$

$\mathcal{L}_c \in [-1, 0]$ denotes the consistency loss that evaluates the mincut given by $\mathbf{B}$. Minimizing $\mathcal{L}_c$ enforces strongly connected grid cells to be grouped into the same region, since the denominator is a constant and the inner product $\langle \mathbf{b}_i, \mathbf{b}_j \rangle$ increases when $\tilde{a}_{i,j}^g$ is large. $\mathcal{L}_c$ will reach its maximum 1 when the region assignments are orthogonal for each pair of connected cells. On the contrary, the minimum occurs only if $\mathbf{B}^T\tilde{\mathbf{A}}^g\mathbf{B} = \mathbf{B}^T\tilde{\mathbf{D}}^g\mathbf{B}$. In other words, it happens when in a graph with M disconnected components, the region assignments are equal for all the grid cells in the same component and orthogonal to the region assignments of cells in different components. However, minimizing $\mathcal{L}_c$ may lead to incorrect or naive solutions. For example, given a connected graph, a trivial yet optimal solution is the one that assigns all grid cells to the same regions. To avoid the degenerate solution by optimizing $\mathcal{L}_c$, the other term (i.e., the orthogonality loss $\mathcal{L}_o$) encourages the assignment to be orthogonal and the regions to be of similar size.