Revisiting Convolutional Neural Networks for Citywide Crowd Flow Analytics

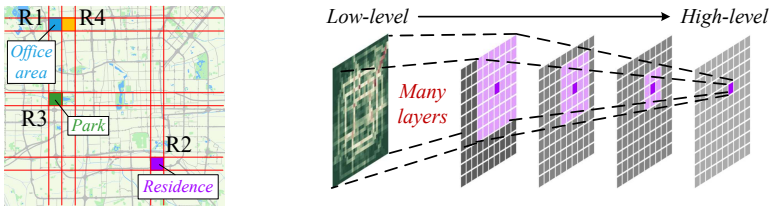
Yuxuan Liang[1] ✉, Kun Ouyang[1], Yiwei Wang[1], Ye Liu[1], Junbo Zhang[2,3,4], Yu Zheng[2,3,4], David S. Rosenblum[1]

[1] School of Computing, National University of Singapore, Singapore
[2] JD Intelligent Cities Research & JD Intelligent Cities Business Unit, Beijing, China
[3] Institute of Artificial Intelligence, Southwest Jiaotong University, China
[4] Xidian University, Xi'an, China
{yuxliang,msjunbozhang,msyuzheng}@outlook.com
{ouyangk,y-wang,liuye,david}@comp.nus.edu.sg

Abstract. Citywide crowd flow analytics is of great importance to smart city efforts. It aims to model the crowd flow (e.g., inflow and outflow) of each region in a city based on historical observations. Nowadays, Convolutional Neural Networks (CNNs) have been widely adopted in raster-based crowd flow analytics by virtue of their capability in capturing spatial dependencies. After revisiting CNN-based methods for different analytics tasks, we expose two common critical drawbacks in the existing uses: 1) inefficiency in learning global spatial dependencies, and 2) overlooking latent region functions. To tackle these challenges, in this paper we present a novel framework entitled DeepLGR that can be easily generalized to address various citywide crowd flow analytics problems. This framework consists of three parts: 1) a local feature extraction module to learn representations for each region; 2) a global context module to extract global contextual priors and upsample them to generate the global features; and 3) a region-specific predictor based on tensor decomposition to provide customized predictions for each region, which is very parameter-efficient compared to previous methods. Extensive experiments on two typical crowd flow analytics tasks demonstrate the effectiveness, stability, and generality of our framework.

1 Introduction

Citywide crowd flow analytics is very critical to smart city efforts around the world. A typical task is citywide crowd flow prediction [21, 20, 12], which aims to predict the traffic (e.g., inflows and outflows of every region) for the next time slot, given the historical traffic observations. It can help the governors conduct traffic control and avoid potential catastrophic stampede before a special event. Another important task is to infer the fine-grained crowd flows from available coarse-grained data sources, which can reduce the expense of urban systems [11, 13]. Other tasks [19, 24] are also actively studied by the community due to the vital impact of citywide crowd flow analytics.



(a) An example of Beijing

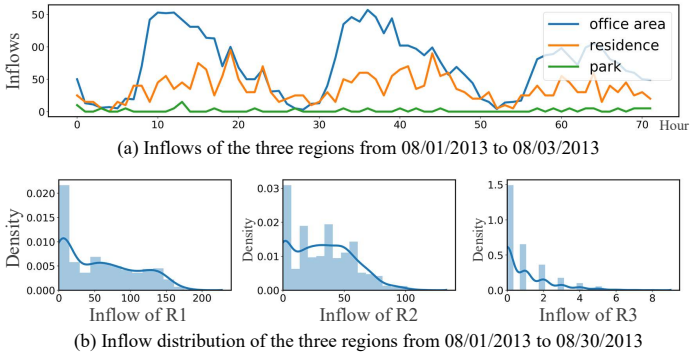
(b) Stacking CNNs for capturing global dependencies

Fig. 1. Application of CNNs for citywide crowd flow analytics (Better view in color).

Crowd flow analytics is not trivial as the traffic can be affected by multiple complex factors in spatio-temporal domains. As shown in Figure 1(a), the inflow of Region R1 is affected by outflows of nearby regions like R4 as well as distant regions, which indicates the spatial dependencies. For the temporal dependencies, crowd flow in a region is affected by recent, daily, and weekly historical traffic. To model the spatio-temporal dependencies, Convolutional Neural Networks (CNNs) have been widely used and achieved promising performance. A pioneering work [21] provided the first CNN-based method (DeepST) for modeling crowd flow, where convolution operators are used to extract spatially near and distant dependencies and the temporal dependencies are considered in different branches of networks. ST-ResNet [20] further enhanced the performance of DeepST using residual structures. Very recently, a novel ConvPlus structure in DeepSTN+ [12] was proposed to learn the long-term spatial dependencies between two arbitrary regions. These CNN-based methods are characterized by two components: a complicated ST feature learner to capture features of the measurements, and a simple task-specific predictor to generate predictions on all regions. However, they have two main drawbacks:

1) *Inefficiency in learning global spatial dependencies.* Take traveling in Beijing (Figure 1) as an example. When predicting the inflow of R1 during morning hours, the outflow of distant regions like R2 needs to be considered, since it is common that people commute from a distant residence location. As people can travel around a modern city quickly, it becomes crucial to capture global spatial dependencies in this task. To this end, existing arts employ two approaches:

- *Stacking CNNs to increase receptive fields.* Most previous studies like DeepST and ST-ResNet employ CNNs to capture information locally. But to capture global spatial dependencies, they have to stack many layers to increase the receptive field of the network (see Figure 1(b)). This is very inefficient since relationships between distant regions can only be captured by a near-top layer with a sufficiently large receptive field to cover all the regions of interest.
- *Learning long-range spatial dependencies directly.* Instead of gradually increasing receptive fields, DeepSTN+ attempts to capture global spatial dependencies in *every layer* using ConvPlus structure, which explicitly models all pairwise relationship between regions. However, a single layer of ConvPlus without pooling requires $O(n^2)$ parameters, where n is the number of regions.



(a) Inflows of the three regions from 08/01/2013 to 08/03/2013

(b) Inflow distribution of the three regions from 08/01/2013 to 08/30/2013

Fig. 2. Illustration of daily patterns and inflow distribution in three regions.

Constrained by this bloated structure, DeepSTN+ cannot easily go deeper to learn higher-level representations for each region. Thus, how to learn global spatial dependencies more efficiently still remains a major challenge.

2) *Ignoring latent region functions.* Different from pixels in image processing, urban regions have different land functions according to their locations and surrounding POIs [23, 14]. Recall that R1, R2 and R3 in Figure 1 correspond to an office area, a residential area and a park zone respectively. From Figure 2(a), it can be seen easily that their daily patterns are entirely different. For instance, the office area (R1) usually reaches a traffic peak in the morning, while the residential area (R2) usually exhibits growth after dinner time. The difference between their daily flow distributions can also be seen from Figure 2(b). However, the aforementioned methods have overlooked such varying latent functions among regions and used a simple predictor with shared parameters to predict flow for all regions, which inevitably resulted in degraded performance.

To address the above problems, we make the following contributions to the community. Primarily, we introduce DeepLGR, the first-ever general framework for raster-based crowd flow analytics. It is named according to how it stratifies a given task into three major procedures: 1) **L**ocal feature extraction to learn representations for each region within small receptive fields; 2) **G**lobal context aggregation to efficiently capture the global spatial dependencies; and 3) **R**egion-specific prediction. Respectively,

- we present the first attempt to extract local region representations using Squeeze-and-Excitation networks (SENet) [5], which excels by including the channel-wise information as additional knowledge;
- we design a global context module that firstly aggregates the region representations using a specific pooling method, and then upsample the global priors back to the original scale to generate global-aware features;
- we introduce a region-specific predictor based on tensor decomposition that factorizes the region-specific parameters of the predictor into a smaller core tensor and adjoint matrices.

In addition, we evaluate our framework on two typical crowd flow analytics tasks: crowd flow forecasting [21, 20] and fine-grained crowd flow inference [11, 13]. Extensive experiments demonstrate the state-of-the-art performance and stability achieved by our framework. We have released our code at https://github.com/yoshall/DeepLGR for public use.

2 Formulation

In this section, we introduce several notations and formulate the problem of crowd flow analytics. As shown in Figure 1(a), we first follow the previous study [21] to partition an area of interest (e.g., a city) evenly into a $H \times W$ grid map based on longitude and latitude where a grid denotes a region. Thus, the crowd flow at a certain time t can be denoted as a 3D tensor $\mathcal{P}_t \in \mathbb{R}^{H \times W \times K}$, where K is the number of different flow measurements (e.g., inflow and outflow). Each entry (i, j, k) denotes the value of the k-th measurement in the region (i, j).

Without loss of generality, we use $\mathcal{X} \in \mathbb{R}^{H \times W \times C}$ and $\mathcal{Y} \in \mathbb{R}^{H' \times W' \times D}$ as the input and output for a crowd flow analytics task, where C and D are the number of channels. For example, in the task of crowd flow prediction [21, 20, 12], the input is the historical observations $\mathcal{X} = \{\mathcal{P}_i | i = 1, 2, \cdots, \tau\} \in \mathbb{R}^{H \times W \times K\tau}$ and the target is to predict $\mathcal{Y} = \mathcal{P}_{\tau+1} \in \mathbb{R}^{H \times W \times K}$.

3 Methodology

Figure 3 presents the framework of DeepLGR, which can be easily generalized to all kinds of citywide crowd flow. Compared to the previous methods composed of an ST feature learner and a shared predictor for all regions, our framework contains three major components: local feature extraction, global context module and region-specific predictor. In the first component, we employ the SENet to learn representations for each region within small (i.e., local) receptive fields from the input tensor $\mathcal{X}$. To capture global spatial dependencies, we further design the global context module that considers the full region of interest. It first extracts global contextual priors from the learned region representations using a specific pooling method, and then upsamples the priors to the original scale to generate the global features. Once we obtain features from both local view and global view, we concatenate them into a tensor and then feed it to the region-specific predictor to make customized predictions for each region respectively.

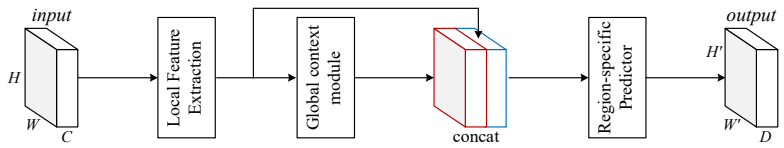


Fig. 3. The pipeline of DeepLGR, which contains three major components.

For spatial dependencies, our framework employs the first two components which strategically capture both local-level (neighborhood) and global-level dependencies between regions. Following the mainstream CNN architectures for citywide crowd flow analytics [21, 20, 12], the temporal dependencies like closeness (recent), period (daily) and trend (weekly), if any, are considered in the channels of input. These temporal dependencies can interact with each other in the backbone network. Next, we will detail the three components respectively.

3.1 Local Feature Extraction

Recall that both the previous and current state-of-the-arts [20, 12] use residual blocks to model the spatial dependencies from nearby regions. However, these methods mainly focus on the spatial dimension and have overlooked the channel-wise information in the feature maps. Thus, we employ SENet 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. Figure 4(a) illustrates the pipeline of the module for local feature extraction. The input is fed to a convolutional layer for initialization. Then, we stack M squeeze-and-excitation (SE) blocks in Figure 4(b) for feature extraction, which is composed of three stages: 1) a residual block [3] for feature learning; 2) a squeeze operation to squeeze global spatial information into a channel descriptor by global average pooling; 3) an excitation operation to fully capture the channel-wise dependencies: it first computes the attention coefficients over each channel via two fully connected layers followed by a sigmoid function, and then rescales the channels of original inputs by these weights. Finally, we use an output convolutional layer to transform the obtained high-level feature maps to the input of the next module. In summary, the SE structure enables this module to learn better representations for each region locally within receptive fields.

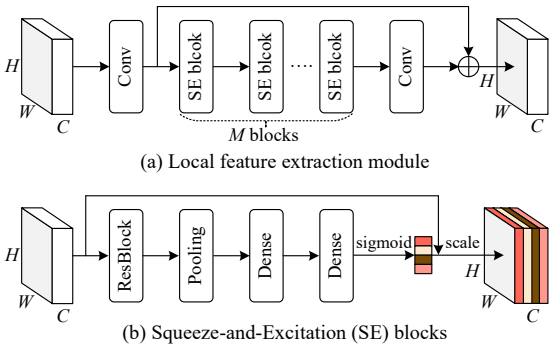


Fig. 4. The pipeline of local feature extraction, where the receptive fields depend on the number of SE blocks (M). Conv: convolutional layer. ResBlock: Residual block. Pooling: global average pooling. Dense: fully connected layer.

3.2 Global Context Module

After local feature extraction, we have designed a specific module that takes the output of the former component as input to generate global contexts for each region, so as to capture global spatial dependencies. As depicted in Figure 5, we first employ spatial pyramid pooling [2] to generate a set of the global priors, where each prior is a spatially abstract of the original input under different pyramid scales. This operation allows the module to separate the feature map into different sub-regions and build pooled representation for different locations. For example, the 1×1 prior (the red cube) denotes the coarsest level with only one single value at each channel, which is equivalent to global pooling operation that covers the whole image. In our experiments, we use a 4-level pyramid (1×1, 2×2, 4×4 and 8×8) to squeeze the input by average pooling.

Once the global priors are obtained, an 1×1 convolution layer followed by a Batchnorm layer [6] is used for dimension reduction of channels from N to $N/8$. Inspired by the study [11] aiming at inferring fine-grained crowd flow from coarse-grained counterparts, we employ the Subpixel block [15] to upsample the priors to generate new representations with the same size as the original inputs. For example, after the Subpixel block in 4×4 branch, the output feature maps grow $H/4$ and $W/4$ times larger in height and width respectively with the number of channels unchanged. Different from PSPNet [22] using bilinear interpolation for upsampling the priors, the Subpixel block considers the relationship between a super-region and its corresponding sub-regions by introducing a parametric design. Finally, we concatenate the input (i.e., region representations) with all levels of global features (i.e., context) as the output of this module.

In summary, this module first converts the input feature map into priors (e.g., 1×1 prior that encodes the information of all regions) and then upsamples the

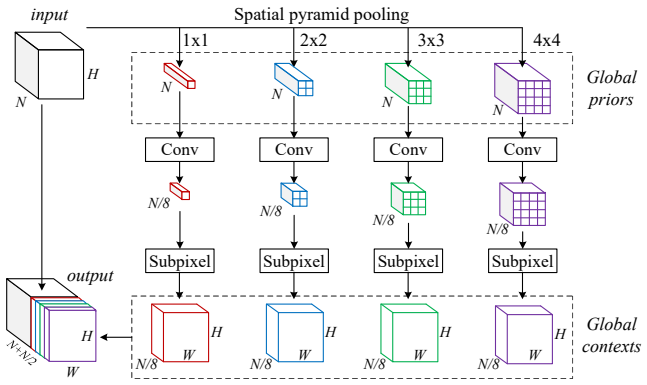


Fig. 5. The pipeline of global context module, where Conv denotes a 1×1 convolutional layer for dimension reduction, and Subpixel contains a convolutional layer and a pixelshuffle operation sequentially to upsample the contextual priors. For simplicity, we use a 4-level pyramid (1×1, 2×2, 3×3 and 4×4) for an illustration.

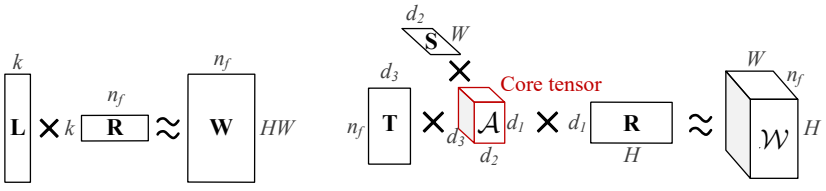
priors to learn the global-context-to-region influence (i.e., global spatial dependencies). Compared to the previous attempt (ConvPlus layer in DeepSTN+), our solution is more efficient and lightweight. Each ConvPlus layer directly models the pairwise relationships among all regions, thus demanding $O(n^2)$ parameters. With the increase of spatial granularity, it will induce extremely high computational costs due to the massive parameters. Thus, DeepSTN+ can hardly learn higher-level representations by simply increasing network depth. In contrast, as we have separated the procedures of local feature extraction and global context modeling, we can easily increase the network depth to gain better capacity.

3.3 Region-Specific Predictor

As mentioned before, each urban region has its unique land function. Previous studies [21, 20, 12] mainly employ a single fully connected layer (equivalent to a 1×1 convolution) with shared weights as the predictor for all regions, which fails to capture this critical property. Thus, it is necessary to assign region-specific predictor to each region.

Recall that the high-level feature obtained from last module is $\mathcal{Z} \in \mathbb{R}^{H \times W \times N'}$ and prediction result is $\mathcal{Y} \in \mathbb{R}^{H \times W \times D}$, where $N' = N + N/2$. Conventionally, the number of parameters in a shared fully connected layer is $n_f = N'D$. To achieve region-specific predictor, an intuitive solution is to use a customized fully connected layer for each region. However, it will induce $HW \times n_f$ parameters (denoted as a tensor $\mathcal{W} \in \mathbb{R}^{H \times W \times n_f}$), which can easily bloat up as the granularity increases. Recently, matrix factorization (MF) was used to avoid these drawbacks [14], in which the parameter tensor $\mathcal{W}$ is reshaped to a matrix $\mathbf{W} \in \mathbb{R}^{HW \times n_f}$. As shown in Figure 6(a), the authors from [14] decompose the weight matrix $\mathbf{W}$ into two *learnable* low-rank matrices, i.e., region embedding matrices $\mathbf{L} \in \mathbb{R}^{HW \times k}$ and parameter embedding matrices $\mathbf{R} \in \mathbb{R}^{k \times n_f}$. With the usage of MF, the number of the predictor parameters can be reduced to $(HW + n_f)k$, where $k \ll n_f$ and $k \ll HW$.

Nonetheless, directly flattening the parameter tensor $\mathcal{W}$ over the region dimension will lose the Euclidean structure of the flow map. For example, near things are more related than distant things according to the first law of geography, which indicates near regions should have similar prediction weights.



(a) Matrix Factorization (b) Tucker (Tensor) Decomposition

Fig. 6. Illustration of matrix factorization and tensor decomposition.

Instead, we present a new idea for decomposing $\mathcal{W}$ using Tensor Decomposition (TD) [17]. It not only preserves the spatial similarity (dependencies) between regions, but also reduces the amount of parameters. As illustrated in Figure 6(b), tensor $\mathcal{W}$ is decomposed into the multiplication of a core tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and three adjoint matrices, where d_1, d_2, and d_3 denote the number of latent factors for each matrix. The computation is as follows:

$$\mathcal{W} = \mathcal{A} \times_R \mathbf{R} \times_S \mathbf{S} \times_T \mathbf{T}, \tag{1}$$

where $\times_R$ stands for the tensor-matrix multiplication; the subscript R is the corresponding mode of the multiplication. For instance, $\mathbf{H} = \mathcal{A} \times_R \mathbf{R}$ is $\mathbf{H}_{ijk} = \sum_{i=1}^{d_1} A_{ijk} \times R_{ij}$. By this, we have changed the optimization target from $\mathcal{W}$ to the core tensor $\mathcal{A}$ as well as the three learnable matrices $\mathbf{R}$, $\mathbf{S}$ and $\mathbf{T}$. The core tensor is a low-rank representation summarising both the parametric and spatial information of the origin tensor $\mathcal{W}$. Compared to MF-based solution [14], our tensor decomposition can handle the higher-order relationships within the parameters. In addition, the number of parameters required is $d_1 d_2 d_3 + d_1 H + d_2 W + d_3 n_f$. Since d_1, d_2 and d_3 are usually very small, TD can achieve even much fewer parameters than MF, which is validated in our experiments.

3.4 Optimization

Since our framework is smooth and differentiable everywhere, it can be trained via the back-propagation algorithm. During the training phase, we use Adam optimizer to train our model by minimizing the entry-wise mean absolute error (MAE) between our prediction $\widehat{\mathcal{Y}}$ and the corresponding ground truth $\mathcal{Y}$:

$$\mathcal{L}(\mathbf{\Theta}) = \left\| \mathcal{Y} - \widehat{\mathcal{Y}} \right\|_1 \tag{2}$$

where $\mathbf{\Theta}$ denotes all learnable parameters in our framework.

4 Experiments

To validate the generality of DeepLGR, we conduct experiments on two typical tasks of citywide crowd flow analytics:

- *Crowd flow forecasting*: This task is to forecast the inflow and outflow of each region in a city from historical readings. Following the settings of [20], we consider the temporal dependencies (i.e., closeness, period and trend) in different channels of input, and the output is the prediction of inflow and outflow for the next timestamp. Similar to [20], we set the length of closeness (recent), period (daily) and trend (weekly) to 5, 3, 3.
- *Fine-grained flow inference*: In this task, we aim to infer fine-grained crowd flows throughout a city based on coarse-grained observations. We extend the state-of-the-art method named UrbanFM [11] using our framework. Specifically, we replace the ResNet-based feature extraction of UrbanFM by our first component (SENet). Then, we add the global context module and region-specific predictor after the subpixel blocks in UrbanFM.

4.1 Experimental Settings

Datasets Two datasets were used in our experiments, including TaxiBJ and HappyValley. The former is the fine-grained version of the ones used by [20] and the latter is provided from [11]. Specifically, TaxiBJ consists of four different time spans (denoted as P1 to P4 with different number of taxicabs and distribution), while HappyValley is the hourly observations of human flow in a theme park in Beijing from ten months. The statistics are detailed in Table 1. We select the flow data between 6am and 11pm to conduct our experiments. Using both datasets, we evaluate DeepLGR over the two aforementioned tasks: In the first task, we employ the first 80% data as training set, the next 10% as validation set and the rest for test set; In the second task, we follow all the experiment settings of [11], including training, validation and test set partition. The upscaling factors in TaxiBJ and HappyValley are 4 and 2 respectively.

Evaluation Metrics We employ two widely-used criteria to evaluate our model from different aspects, including mean absolute error (MAE) and symmetric mean absolute percentage error (SMAPE). They are defined as:

$$\text{MAE} = \frac{1}{z}\sum_{i=1}^{z}|y_i - \tilde{y}_i|\,,\ \ \text{SMAPE} = \frac{1}{z}\sum_{i=1}^{z}\frac{|y_i - \tilde{y}_i|}{|y_i| + |\tilde{y}_i|},$$

where y and $\tilde{y}$ are ground truth and predicted value respectively; z is the total number of all entries. Smaller metric scores indicate better model performance.

Baselines In the first task, we compare our framework with heuristics, time series methods and CNN-based baselines. Specifically, a naive method (**Last**) simply uses the last observation as the prediction result, and another heuristic (**CA**) leverages the closeness property to predict the future crowds by averaging the values from the previous 5 time steps. **ARIMA** is a well-known model for forecasting future values in a time series. Besides, the CNN-based baselines (including **DeepST** [21], **ST-ResNet** [20], **ConvLSTM** [15] and **DeepSTN+** [12]) have been introduced in Section 1.

Table 1. Dataset description.

Dataset	**TaxiBJ**	**HappyValley**
Data type	Inflow and outflow	Staying flow
Resolution	(128, 128)	(50,100)
Sampling rate	30 minutes	1 hour
Time Span (mm/dd/yyyy)	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

The second task was introduced only very recently by [11], where the authors presented the state-of-the-art method named **UrbanFM**. It considers the unique characteristics of this task, including the spatial hierarchy and external factors. Other strong baselines included in this work are related to image super-resolution, such as **VDSR** [7] and **SRResNet** [8]. We mainly use these three baselines for model comparison in this task. It is worth noting that all baselines are implemented with their default settings in both tasks.

Training Details & Hyperparameters Our framework, as well as the above baselines, are fully implemented by Pytorch 1.1.0 with one GTX 2080TI. During the training phase, the learning rate is 0.005 and the batch size is 16. For the number of stacked SE blocks (denoted as M) in the first component, we conduct a grid search over $\{3, 6, 9, 12\}$. For simplicity, we use the same hidden dimension (i.e., number of channels) at each 3×3 convolutional layer in SE blocks, and conduct a grid search over $F = \{32, 64, 128\}$.

4.2 Results on Crowd Flow Forecasting

Model Comparison Here, we compare our framework with the baselines over the two datasets. We report the result of DeepLGR with $M = 9$ and $F = 64$ as our default setting. Further results regarding different M will be discussed later.

Table 2. Prediction results on TaxiBJ over different time spans (P1-P4), where the bold number indicates the best performance of the column. We train and test each method five times, and present results using the format:"mean ± standard deviation".

Method	P1		P2	
	MAE	SMAPE	MAE	SMAPE
CA	3.43	0.290	4.23	0.288
Last	3.39	0.242	4.09	0.241
ARIMA	3.08	0.403	3.53	0.385
DeepST	2.59 ± 0.05	0.41 ± 0.01	2.94 ± 0.05	0.39 ± 0.01
ST-ResNet	2.53 ± 0.05	0.38 ± 0.05	2.93 ± 0.06	0.34 ± 0.07
ConvLSTM	2.42 ± 0.02	0.41 ± 0.01	2.77 ± 0.01	0.39 ± 0.01
DeepSTN+	2.33 ± 0.04	0.35 ± 0.08	2.67 ± 0.02	0.32 ± 0.05
DeepLGR	**2.15 ± 0.00**	**0.19 ± 0.00**	**2.46 ± 0.00**	**0.18 ± 0.00**
Method	P3		P4	
	MAE	SMAPE	MAE	SMAPE
CA	4.17	0.286	2.81	0.286
Last	4.07	0.240	2.82	0.239
ARIMA	3.68	0.363	2.61	0.420
DeepST	2.97 ± 0.04	0.39 ± 0.01	2.16 ± 0.04	0.43 ± 0.02
ST-ResNet	2.91 ± 0.06	0.33 ± 0.05	2.15 ± 0.04	0.32 ± 0.06
ConvLSTM	2.87 ± 0.01	0.39 ± 0.01	2.09 ± 0.02	0.43 ± 0.02
DeepSTN+	2.82 ± 0.04	0.38 ± 0.05	2.05 ± 0.01	0.34 ± 0.05
DeepLGR	**2.56 ± 0.02**	**0.19 ± 0.04**	**1.84 ± 0.01**	**0.19 ± 0.00**

Table 2 shows the experimental results over P1 to P4 in TaxiBJ. We can observe that our framework clearly outperforms all baselines over both metrics. For instance, DeepLGR shows 10.2% and 44.1% improvements on MAE and SMAPE beyond the state-of-the-art method (DeepSTN+) in P4. The conventional model ARIMA performs much worse than deep learning models in these datasets, since it only considers the temporal dependencies among time series. Apart from the CNN-based methods, ConvLSTM advances DeepST and ST-ResNet because of the positive effect of its LSTM structure. However, it overlooks the global spatial dependencies between regions, which leads to inferiority compared to DeepSTN+ and DeepLGR. Another interesting observation is that the heuristics including CA and Last achieves much less SMAPE than previous CNN-based methods. Recall that SMAPE prefers to penalize the errors in regions with lower flow volumes. This observation reveals the importance of the temporal dependencies in such regions since CA and Last only consider the temporal closeness for forecasting. Only our method performs better than the heuristics on SMAPE with the usage of tensor decomposition, which will be detailed in the ablation study. Last but not least, DeepLGR is also more stable than the baselines according to the standard deviation observations.

Compared to TaxiBJ with a citywide scale, HappyValley focuses on a local area with a highly skewed flow distribution, where only a few regions contain dense populations. Table 3 presents a comprehensive comparison of each model over this dataset. First, it can be seen easily that our framework shows great superiority against the CNN-based methods and slightly outperforms ConvLSTM in terms of both metrics, while using as little as 6.2% of the amount of parameters required in the state-of-the-art method (DeepSTN+). This fact demonstrates that our model is more practical than other CNN-based solutions in real-world systems. Second, similar to the results in TaxiBJ, DeepLGR performs more stable than the baselines according to the standard deviation in multiple experiments. Third, the heuristic method (Last) achieves the lowest SMAPE but the second-highest MAE, which can prove the skew distribution of this dataset. Last, the fact that DeepLGR and DeepSTN+ outperform ST-ResNet verifies the necessity of modeling global context in such a small area.

Table 3. Prediction results of various methods on the HappyValley dataset, where #Params is the number of parameters and M denotes million.

Method	#Params	MAE	SMAPE
CA	x	2.23	0.46
Last	x	2.20	**0.38**
ARIMA	0.00M	2.14	0.47
DeepST	0.59M	2.02 ± 0.05	0.56 ± 0.05
ST-ResNet	2.73M	1.98 ± 0.05	0.53 ± 0.04
ConvLSTM	5.98M	1.86 ± 0.01	0.48 ± 0.10
DeepSTN+	15.70M	1.92 ± 0.01	0.54 ± 0.06
DeepLGR	0.97M	**1.84 ± 0.01**	0.40 ± 0.02

Ablation Study To further investigate the effectiveness of each component, we compare DeepLGR with its variants over TaxiBJ-P1. For simplicity, we use the terms as local, global and TD to denote the three components in our framework respectively. Based on them, DeepLGR and its variants can be denoted as:

- **local+global+TD**: The original implementation of DeepLGR.
- **local+global+MF**: To show the effectiveness and lightweight property of TD against MF, we replace TD in the region-specific predictor by MF.
- **local+global**: Similar to the CNN-based baselines [21, 20, 12], this variant uses shared parameters (i.e., not region-specific) as the predictor.
- **local+TD**: The variant of DeepLGR without global context module.
- **local+MF**: We first remove global context module from DeepLGR and then replace TD in region-specific predictor by MF.
- **local+bilinear**: We employ bilinear interpolation rather than Subpixel block to upsample the global priors, so as to obtain new global representations.
- **local**: The last two components are removed from DeepLGR.

Table 4 illustrates the variant comparison over TaxiBJ-P1. We discuss the effects of each model component as follows:

- *Local feature extraction*: A powerful ST feature extractor enables the capability of extracting useful representations for each region. Compared to previous attempts like ST-ResNet based on residual blocks, our feature extraction module largely improves the performance (e.g., local vs. ST-ResNet in Table 2 and 4). We further investigate the effects of the number of SE blocks in this module. As shown in Figure 7, it achieves the best performance when $M = 6$ in the test set. Noted that we choose $M = 9$ as the default setting of DeepLGR because of its best performance on the validation set rather than the test set. Besides, we replace the SE blocks in this module by residual blocks to show the advantages of SE blocks, where the results are also in Figure 7.
- *Global context module*: As a vital component in our framework, this module provides the global information to boost the performance. As illustrated in Table 4, the comparison between local and local+global (also local+TD and local+global+TD) can verify the effectiveness of this module. With the usage of Subpixel block with a parametric design, local+global brings an improvement beyond local+bilinear.

Table 4. Results of different variants over TaxiBJ-P1 (trained/tested five times).

Variants	#Params	MAE	SMAPE
local	0.72M	2.21 ± 0.01	0.37 ± 0.03
local+MF	0.89M	2.19 ± 0.02	0.36 ± 0.03
local+TD	0.74M	2.19 ± 0.01	0.32 ± 0.03
local+bilinear	0.73M	2.20 ± 0.02	0.35 ± 0.03
local+global	2.30M	2.17 ± 0.02	0.29 ± 0.03
local+global+MF	2.46M	2.15 ± 0.00	0.27 ± 0.01
local+global+TD	2.31M	**2.15 ± 0.00**	**0.19 ± 0.00**

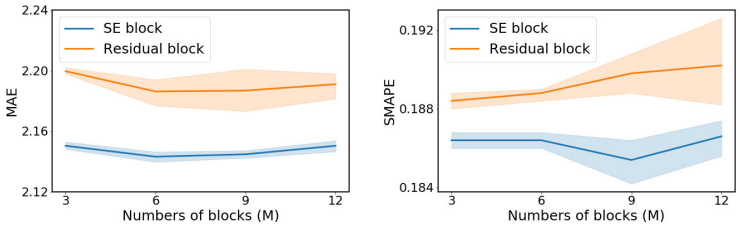


Fig. 7. SE vs. residual block over P1, where the shade area is the standard deviation.

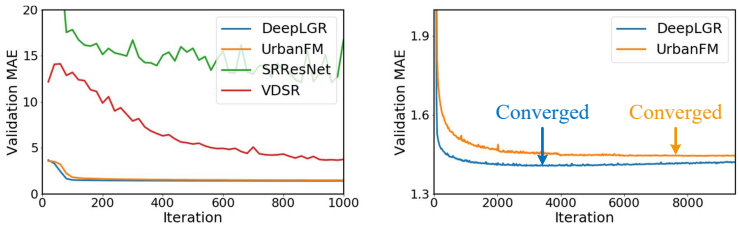
– *Region-specific predictor*: This module is used to determine the region-specific parameters for predictions. Thus, we compare it with a shared fully connected layer with n_f parameters (local+global), and the matrix decomposition method. From the last three rows of Table 4, we observe that TD demonstrates very competitive accuracy while using as little as 6.3 % of the number of parameters required in MF (i.e., 0.01M vs. 0.16M). Moreover, TD significantly outperforms MF over SMAPE since it allows the model to capture spatial dependencies between regions.

4.3 Results on Fine-grained Flow Inference

Experimental results on the second task have demonstrated the superiority of our framework again. From Table 5, we have the following observations: 1) UrbanFM equipped with our framework (denoted as local+global+TD) shows considerable improvements against its original version on both datasets, validating its great generality in different applications. For example, DeepLGR achieves 5.8% lower MAE and 28.0% lower SMAPE than UrbanFM in the TaxiBJ-P1 dataset. 2) The three components of DeepLGR are effective according to the advancement of performance (only except local vs. UrbanFM in HappyValley). 3) Compared to VDSR and SRResNet for image-resolution , UrbanFM outperforms them by considering the domain knowledge, i.e., spatial hierarchy and external influence [11]. From above discussions, we can see that existing approaches like UrbanFM can be easily integrated with our framework.

Table 5. Results of various models for fine-grained flow inference. We train/test each method five times, and present results using the format:"mean ± standard deviation".

Method	TaxiBJ-P1		HappyValley	
	MAE	SMAPE	MAE	SMAPE
VDSR	2.23 ± 0.05	0.54 ± 0.03	2.13 ± 0.04	0.61 ± 0.02
SRResNet	2.20 ± 0.05	0.52 ± 0.03	1.89 ± 0.05	0.61 ± 0.03
UrbanFM	2.07 ± 0.03	0.25 ± 0.02	1.80 ± 0.02	0.41 ± 0.02
local	1.98 ± 0.01	0.20 ± 0.01	1.83 ± 0.01	0.43 ± 0.01
local+global	1.96 ± 0.00	0.20 ± 0.01	1.78 ± 0.01	0.38 ± 0.01
local+global+TD	**1.95 ± 0.00**	**0.18 ± 0.01**	**1.76 ± 0.01**	**0.35 ± 0.00**



(a) Validation MAE curve of various models (b) DeepLGN vs. UrbanFM

Fig. 8. Convergence speed of various methods over P1.

We further investigate the efficiency of DeepLGR. Figure 8 plots the MAE on the validation set during the training phase using TaxiBJ-P1. Remarkably, UrbanFM and DeepLGR converge much smoother and faster than the others as shown in Figure 8(a). A more detailed comparison between UrbanFM and DeepLGR lies in Figure 8(b). From this figure, we can see that DeepLGR converges at iteration 3540 (epoch 37) while UrbanFM early-stops at iteration 7720 (epoch 81). This fact demonstrates that our framework can also accelerate the training phase of existing method.

5 Related Work

Citywide crowd flow analytics has attracted considerable attention of researchers in recent years. A series of studies have explored forecasting millions or even billions of individual mobility traces [16, 1]. Different from analyzing crowd behaviors on an individual level, several works started to forecast citywide crowd flow by aggregating the crowds into corresponding regions [10, 4]. Among them, statistical learning was employed to capture inter-region relationship. With interest in obtaining fine-grained regional data, several studies [11, 24, 13] presented techniques to recover fine-grained crowd flow from coarse-grained data.

Recently, there have been many attempts focusing on end-to-end deep learning solutions such as CNNs for citywide crowd flow analytics. A pioneering study by [21] presented a general framework based on CNNs for citywide crowd flow prediction. By using a CNN architecture, their method can capture the spatio-temporal correlations reasonably and accurately. To overcome the gradient vanishing problem, they further integrated their framework using deep residual learning [20]. Similar insight has been applied in taxi demand prediction [19]. Moreover, there are also several studies [25, 18] using RNNs to model the periodic temporal dependencies. Very recently, a ConvPlus structure [12] showed the state-of-the-art performance by directly modeling the long-range spatial dependencies between region pairs. However, as detailed in Section 1, these methods are very inefficient in learning global spatial dependencies and none of them considers latent land function. To tackle these drawbacks, we have presented a general framework that can be easily generalized to all kinds of crowd flow data.

6 Conclusion and Future Work

In this paper, we have carefully investigated existing CNN-based methods for citywide crowd flow analytics, and exposed their inefficiency in capturing global spatial dependencies and incapability in generating region-specific predictions. Based on our discovery, we have presented the DeepLGR framework which decouples the local feature extraction and global context modeling, and provides a parameter-efficient solution for customizing regional outputs. We have evaluated DeepLGR over two real-world citywide crowd flow analytics tasks. In the prediction task, DeepLGR outperforms the state-of-the-art (DeepSTN+) by average 8.8% and 45.9% on TaxiBJ dataset, and 4.2% and 25.9% on HappyValley dataset in terms of MAE and SMAPE metrics respectively. Moreover, our framework is more lightweight than the state-of-the-art methods, which is very important in real practice. In the second task, we have verified that the existing approach can be easily integrated with our framework to boost its performance. In the future, we will explore two directions. First, we notice that manually designing neural networks requires amount of expert efforts and domain knowledge. To overcome this problem, we can follow a very recent study [9] to study Neural Architecture Search (NAS), which can automatically construct a general neural network for diverse spatio-temporal tasks in cities. Second, we will extend our framework to a much broader set of spatio-temporal tasks by using graph convolutions.

Acknowledgement

We thank all reviewers for their constructive and kind suggestions. This work was supported by the National Key R&D Program of China (2019YFB2101805) and Beijing Academy of Artificial Intelligence (BAAI).

References

1. Fan, Z., Song, X., Shibasaki, R., Adachi, R.: Citymomentum: an online approach for crowd behavior prediction at a citywide level. In: Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (2015)
2. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence **37**(9), 1904–1916 (2015)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
4. Hoang, M.X., Zheng, Y., Singh, A.K.: FCCF: forecasting citywide crowd flows based on big data. In: SIGSPATIAL. p. 6 (2016)
5. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
6. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
7. Kim, J., Kwon Lee, J., Mu Lee, K.: Accurate image super-resolution using very deep convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1646–1654 (2016)

8. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4681–4690 (2017)
9. Li, T., Zhang, J., Bao, K., Liang, Y., Li, Y., Zheng, Y.: Autost: Efficient neural architecture search for spatio-temporal prediction. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2020)
10. Li, Y., Zheng, Y., Zhang, H., Chen, L.: Traffic prediction in a bike-sharing system. In: SIGSPATIAL. pp. 1–10 (2015)
11. Liang, Y., Ouyang, K., Jing, L., Ruan, S., Liu, Y., Zhang, J., Rosenblum, D.S., Zheng, Y.: Urbanfm: Inferring fine-grained urban flows. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 3132–3142 (2019)
12. Lin, Z., Feng, J., Lu, Z., Li, Y., Jin, D.: Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1020–1027 (2019)
13. Ouyang, K., Liang, Y., Liu, Y., Tong, Z., Ruan, S., Zheng, Y., Rosenblum, D.S.: Fine-grained urban flow inference. arXiv preprint arXiv:2002.02318 (2020)
14. Pan, Z., Wang, Z., Wang, W., Yu, Y., Zhang, J., Zheng, Y.: Matrix factorization for spatio-temporal neural networks with applications to urban flow prediction. In: CIKM. pp. 2683–2691 (2019)
15. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1874–1883 (2016)
16. Song, X., Zhang, Q., Sekimoto, Y., Shibasaki, R.: 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. pp. 5–14 (2014)
17. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
18. Yao, H., Tang, X., Wei, H., Zheng, G., Li, Z.: Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In: AAAI (2019)
19. Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J., Li, Z.: Deep multi-view spatial-temporal network for taxi demand prediction. In: AAAI (2018)
20. Zhang, J., Zheng, Y., Qi, D.: Deep spatio-temporal residual networks for citywide crowd flows prediction. In: Thirty-First AAAI Conference (2017)
21. Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X.: DNN-based prediction model for spatio-temporal data. In: SIGSPATIAL. p. 92 (2016)
22. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)
23. Zheng, Y., Capra, L., Wolfson, O., Yang, H.: Urban computing: concepts, methodologies, and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **5**(3), 1–55 (2014)
24. Zong, Z., Feng, J., Liu, K., Shi, H., Li, Y.: DeepDPM: Dynamic population mapping via deep neural network. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 1294–1301 (2019)
25. Zonoozi, A., Kim, J.j., Li, X.L., Cong, G.: Periodic-crn: A convolutional recurrent model for crowd density prediction with recurring periodic patterns. In: IJCAI. pp. 3732–3738 (2018)