# Curb-GAN: Conditional Urban Traffic Estimation through Spatio-Temporal Generative Adversarial Networks

Yingxue Zhang[$], Yanhua Li[$], Xun Zhou[†], Xiangnan Kong[$], and Jun Luo[#]
[$]Worcester Polytechnic Institute, USA
[†]University of Iowa, USA; [#]Lenovo Group Limited, Hong Kong
{yzhang31,yli15,xkong}@wpi.edu,xun-zhou@uiowa.edu,jluo1@lenovo.com

## ABSTRACT

Given an urban development plan and the historical traffic observations over the road network, the Conditional Urban Traffic Estimation problem aims to estimate the resulting traffic status prior to the deployment of the plan. This problem is of great importance to urban development and transportation management, yet is very challenging because the plan would change the local travel demands drastically and the new travel demand pattern might be unprecedented in the historical data. To tackle these challenges, we propose a novel Conditional Urban Traffic Generative Adversarial Network (Curb-GAN), which provides traffic estimations in consecutive time slots based on different (unprecedented) travel demands, thus enables urban planners to accurately evaluate urban plans before deploying them. The proposed Curb-GAN adopts and advances the conditional GAN structure through a few novel ideas: (1) dealing with various travel demands as the "conditions" and generating corresponding traffic estimations, (2) integrating dynamic convolutional layers to capture the local spatial auto-correlations along the underlying road networks, (3) employing self-attention mechanism to capture the temporal dependencies of the traffic across different time slots. Extensive experiments on two real-world spatio-temporal datasets demonstrate that our Curb-GAN outperforms major baseline methods in estimation accuracy under various conditions and can produce more meaningful estimations.

## CCS CONCEPTS

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

## KEYWORDS

generative adversarial networks; self-attention; traffic estimation; spatial-temporal data

**ACM Reference Format:**
Yingxue Zhang[$], Yanhua Li[$], Xun Zhou[†], Xiangnan Kong[$], and Jun Luo[#]. 2020. Curb-GAN: Conditional Urban Traffic Estimation through Spatio-Temporal Generative Adversarial Networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3394486.3403127


*KDD '20, August 23–27, 2020, Virtual Event, CA, USA*

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403127

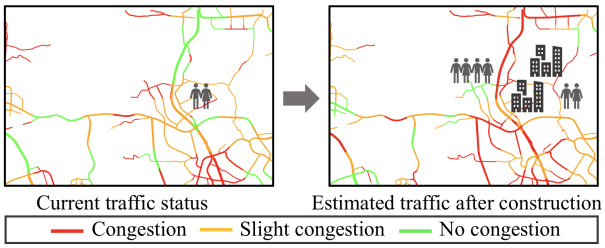



**Figure 1: Example of traffic estimation and evaluation for urban planning in Vaughan, Canada.**

## 1 INTRODUCTION

The fast urbanization in recent years has brought huge impacts on urban traffic due to the growth of urban population, which potentially increases the travel demands and the risk of worsening traffic conditions caused by the overload of the transportation infrastructures. Therefore, urban traffic estimation has acted an important role in the process of urban development, which can provide insights for urban planning, traffic management and resource allocation, and help to improve the urban transportation efficiency and living environment [39]. For example, as shown in Figure 1, new sports village was planed to be built in Vaughan, Canada by the local government in 2019, which would increase the local travel demands to a great extent. Considering the potential traffic pressure the construction would bring, the plan was finally rejected [2]. Thus, urban traffic estimation is a critical step when evaluating an urban development plan before its deployment.

Given an urban development plan (with new travel demands it would produce), the underlying road network, and the historical traffic observations, *the problem of conditional urban traffic estimation* aims at evaluating the deployment plan by estimating traffic status under the new travel demands in consecutive time slots.

The conditional urban traffic estimation problem is challenging and difficult to solve due to the following reasons:
(1) *Traffic status heavily depends on travel demands.* Major changes in travel demands due to emergencies or urban constructions (*e.g.*, a newly constructed commercial center or hospital) could drastically change the traffic status [37]. In such a scenario, data-driven approaches may not effectively estimate the traffic after the demand changes due to the lack of data.
(2) *Spatial auto-correlations.* The traffic status in nearby locations tends to correlate with each other. Capturing such auto-correlations

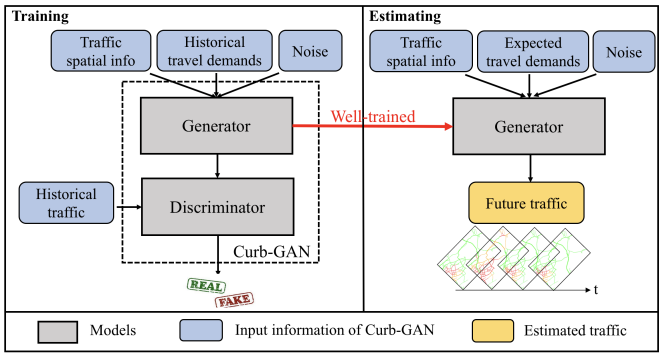



**Figure 2: Insight of the framework.**

is non-trivial. However, in a traffic network, the strength of traffic spatial auto-correlations varies at different locations and highly depends on the underlying road network structures.

(3) *Temporal auto-correlations.* Traffic status at the same location also exhibits strong auto-correlations over time. The traffic status at one location is highly correlated with its precedents. Such impacts also bring big challenges when estimating the urban traffic.

To estimate the urban traffic, many estimation methods have been proposed from different perspectives. The classic traffic estimation methods have been extensively studied in the literature [3, 11, 32, 33]. These works train machine learning models using historical traffic data trying to capture the correlations among the past traffic, environmental features and the future traffic. However, when predicting the traffic impacts of drastic increased (or decreased) travel demands, these models would fail because they cannot capture the future traffic changes caused by the travel demand changes due to the lack of training samples.

Moreover, in recent years, there have been a lot of urban traffic prediction works using deep neural networks to model spatial and temporal auto-correlations. [17] used stacked autoencoders to predict the travel demands. [34] and [40] used ConvLSTM and ConvGRU to predict traffic accidents and crowd density. Others [5, 31] used the combination of CNN and LSTM to predict the road traffic speed and crowd flows. These models captured the temporal and spatial dependencies simultaneously, however, they did not consider the impact of conditions (*e.g.*, travel demand changes), cannot accurately capture the spatio-temporal auto-correlations with various travel demands, and thus fail to provide long-term predictions without any prior knowledge. A more recent work [37] proposed a TrafficGAN model to solve the traffic estimation problem. However, TrafficGAN ignores the temporal auto-correlations of traffic status and can only make snapshot estimations.

To tackle the aforementioned challenges and solve the conditional urban traffic estimation problem, in this paper, we propose a novel Conditional Urban Traffic Generative Adversarial Network (Curb-GAN), which can provide effective traffic estimations in consecutive time slots based on different travel demands. Figure 2 shows the solution framework, where the proposed Curb-GAN utilizes conditional GAN structure to control the generated traffic based on various travel demands, and the well-trained generator is used to estimate future traffic. Curb-GAN features a few novel designs, including using dynamic convolutional layers to capture the spatial auto-correlations along the road networks, and applying self-attention mechanism to capture the traffic temporal dependencies across different time slots. Our **main contributions** are summarized as follows:

- We model the conditional traffic estimation problem as a traffic data generation problem, and propose a novel deep generative model Curb-GAN, which can generate future traffic estimations in consecutive time slots based on different travel demands in any region of a city. (See Sec 3.)
- Building blocks containing dynamic convolutional layers and self-attention mechanism are designed to capture the shared patterns across spatio-temporal regions of how traffic status evolves according to time changes, travel demand changes and underlying road network structures. (See Sec 3.3.)
- We conduct extensive experiments on two real-world spatio-temporal datasets (taxi inflow and traffic speed) to evaluate our proposed Curb-GAN. The experiment results verify that Curb-GAN can significantly improve the urban traffic estimation performance and outperform all existing baseline methods on both datasets. (See Sec 4.) *We made our code and unique dataset available to contribute to the research community [1].*

**Table 1: Notations**

| Notations | Descriptions |
|---|---|
| $i, j$ | Locations(coordinates in a grid world) |
| $\mathcal{S} = \{s_{ij}\}$ | Grid cells |
| $\mathcal{R} = \{R_{ij}\}$ | All target regions |
| $N_s = \ell^2 \in \mathbb{N}$ | Number of grid cells in a region |
| $N_t \in \mathbb{N}$ | Number of time slots within a day |
| $\tau_{\text{in}} \in \mathbb{N}$ | Number of days of historical traffic observations |
| $\mathcal{D}^R = \{d_t^R\}$ | Travel demand sequence of one day in $R$ |
| $\mathcal{M}^R = \{\boldsymbol{M}_t^R\}$ | Traffic distribution sequence of one day in $R$ |
| $\mathcal{A}^R = \{\boldsymbol{A}_t^R\}$ | Traffic correlation matrix sequence in $R$ |
| $\mathcal{C}^R = \{\boldsymbol{C}_t^R\}$ | Traffic condition sequence of one day in $R$ |

## 2 PRELIMINARIES

In this section, we first introduce the preliminaries used in this paper and then formalize the conditional urban traffic estimation problem.

### 2.1 Notations and Definitions

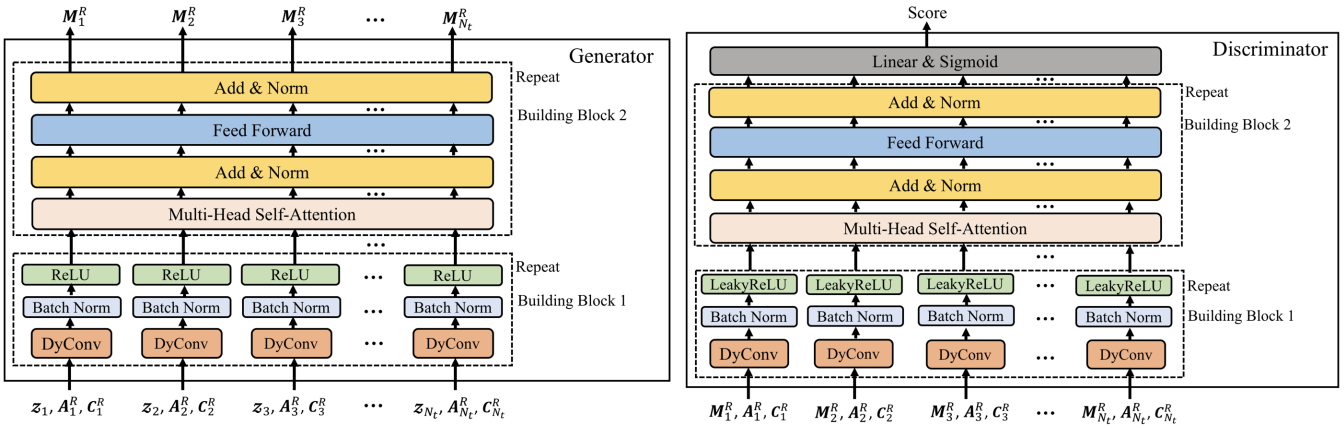
We list the notations that will be used throughout the paper in Table. 1.

**Definition 1 (Grid cells).** We split the city into $I \times J$ grid cells with equal side-length in latitude and longitude, denoted as $\mathcal{S} = \{s_{ij}\}$, where $1 \le i \le I, 1 \le j \le J$.

**Definition 2 (Target region).** A target region $R$ is a square geographic region in the city, formed by $N_s = \ell \times \ell$ grid cells. The whole city can be split into overlapping regions $\mathcal{R} = \{R_{ij}\}$, where $R_{ij} = \langle s_{ij}, \ell \rangle$ is uniquely defined by an anchor grid cell $s_{ij}$ on its top-left corner and a number $\ell$ of grid cells on the side[1].

**Definition 3 (Travel demand).** The travel demand of an area captures the total number of departures in a period of time. Thus, we denote the travel demand of a grid cell $s$ in time slot $t$ as $d_t^s \in \mathbb{N}$. Moreover, the travel demands of a target region $R$ within a day is denoted as a sequence $\mathcal{D}^R = \{d_1^R, \ldots, d_{N_t}^R\} \in \mathbb{N}^{N_t}$, where $d_t^R$ is

[1]Note that target regions can also be defined as rectangles rather than squares. For simplicity, we use square shape of target regions in this work.




(a) Generator of Curb-GAN

(b) Discriminator of Curb-GAN

**Figure 3: Overview of Curb-GAN.**

the sum of travel demands in all grid cells within $R$ in time slot $t$, *i.e.*, $d_t^R = \sum_{s \in R} d_t^s \in \mathbb{N}$.

**Definition 4 (Traffic status and traffic distribution).** Traffic status indicates the quality of traffic, which can be measured by traffic speed, traffic inflow/outflow, traffic volume, *etc.* We denote $m_t^s$ as the average traffic status of grid cell $s$ in time slot $t$. The traffic distributions of a target region $R$ within a day is denoted as a tensor $\mathcal{M}^R = \{\boldsymbol{M}_1^R, \ldots, \boldsymbol{M}_{N_t}^R\} \in \mathbb{R}^{N_t \times \ell \times \ell}$, where we denote the $\ell \times \ell$ matrix $\boldsymbol{M}_t^R$ as the traffic distribution in $R$ in time slot $t$, each entry of $\boldsymbol{M}_t^R$ is $m_t^s$, where $s \in R$.

**Definition 5 (Traffic correlation matrix).** The traffic correlation matrices of a target region $R$ within a day is denoted as a tensor $\mathcal{A}^R = \{\boldsymbol{A}_1^R, \ldots, \boldsymbol{A}_{N_t}^R\} \in \mathbb{R}^{N_t \times N_s \times N_s}$, where $\boldsymbol{A}_t^R$ is a traffic correlation matrix of size $N_s \times N_s$ in region $R$ in time slot $t$ [37], $\boldsymbol{A}_t^R$ is non-negative and row-normalized.

Traffic correlations capture the inherent traffic dependencies between a grid cell pair (*i.e.*, auto-correlations). We use Pearson correlation coefficient of each pair of grid cells to quantify their corresponding traffic correlations. Note here we assume that traffic status at different locations are either positively correlated or independent. Negative correlations, though theoretically possible, are not considered in this paper. The reasons and details are shown in Appendix A.

## 2.2 Problem Definition

A city area is partitioned into regions $\mathcal{R}$, given $\tau_{\text{in}} \times \|\mathcal{R}\|$ samples of $\mathcal{D}$ and $\mathcal{M}$, for one specific target region $R$, we aim to estimate the traffic distributions $\hat{\mathcal{M}}^R = \{\hat{\boldsymbol{M}}_1^R, \ldots, \hat{\boldsymbol{M}}_{N_t}^R\}$ in consecutive time slots based on a given expected travel demand sequence $\hat{\mathcal{D}}^R = \{\hat{d}_1^R, \ldots, \hat{d}_{N_t}^R\}$.

# 3 METHODOLOGIES

To solve the conditional urban traffic estimation problem, we are inspired by the conditional GAN [18] model, since our traffic estimation problem is similar to conditional image sequences generation problem, where the travel demand can be treated as a condition, the traffic distribution of a region in one time slot can be treated as an"image" and the traffic status of each grid can be viewed as a "pixel" value. Thus, the conditional GAN (cGAN) structure could be potentially used to solve the conditional traffic estimation problem. However, the unique challenges (2) and (3) of our problem mentioned in Sec 1 prevent the state-of-the-art cGAN models from solving it, since simple cGAN model cannot capture the spatial and temporal auto-correlations of traffic very well. Hence, we propose a novel generative model – Curb-GAN which can more accurately capture the spatial auto-correlations and temporal dependencies of traffic, control the generation results with desired travel demands, and generate realistic traffic estimations in consecutive time slots.

In this section, we introduce the architecture of Curb-GAN for traffic estimation problem. Following the conditional GAN structure, Curb-GAN consists of a generator $G$ and a discriminator $D$, and both the generator and discriminator apply dynamic convolutional layers [37] and self-attention mechanism [26] to deal with the spatial and temporal auto-correlations of traffic.

## 3.1 Dynamic Convolutional Layer (DyConv)

In urban areas, the strength of traffic spatial auto-correlations is often heterogeneous, which mostly relies on the locations and the complex underlying road structures. Based on the First Law of Geography [24], nearby locations and closely connected roads often have stronger traffic spatial auto-correlations. Hence, we apply dynamic convolutional layers in both $G$ and $D$, which can better capture the diverse footprints of spatial auto-correlations of traffic status.

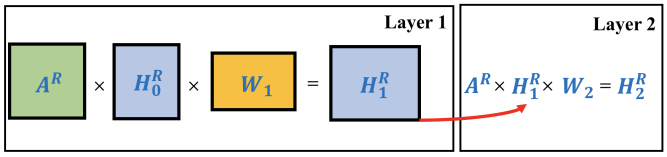
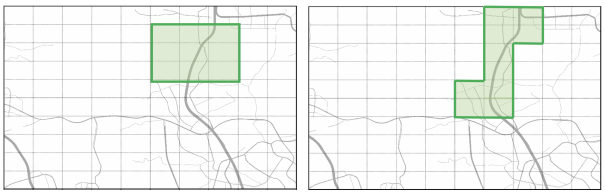



**Figure 4: Propagation rule of DyConv.**

The input of the dynamic convolutional layer includes: (1) a traffic status matrix $\boldsymbol{H}^R$ of size $N_s \times d_{\text{status}}$ ($d_{\text{status}}$: number of traffic status measures, if traffic status is represented by only one measure, *e.g.*, traffic speed, then $d_{\text{status}} = 1$) and (2) a traffic correlation matrix $\boldsymbol{A}^R$.



(a) Filter in standard Conv (b) Filter in DyConv

**Figure 5: Filter comparisons of standard Conv and DyConv.**

The layer-wise propagation rule of DyConv is presented in Eq. 1 and the output of DyConv is a new traffic status matrix:

$$H_i^R = f\left(H_{i-1}^R, A^R\right) = \sigma\left(A^R H_{i-1}^R W_i\right), \tag{1}$$

where $H_i^R$ is the output traffic status matrix of region $R$ in $i$-th layer, $W_i$ is the weight matrix and $\sigma$ is an activation function. The rule is illustrated in Figure 4.

The traffic correlation matrix $A^R$ in DyConv can also be viewed as a "filter", similar to the filter in a standard convolutional layer (Conv), which is applied to images and has fixed size and regular shape. The "filter" in DyConv created by $A^R$ is applied to the traffic status matrix $H^R$ which has irregular shape and size. As shown in Figure 5, the filter of standard Conv would cover some grids having no roads or very low traffic correlations and thus cannot capture the roads accurately, but the "filter" created by $A^R$ in DyConv exactly captures the road structures since $A^R$ can control the shape and size of the "filter" to make it only cover the grid cells which have very strong traffic correlations.

## 3.2 Self-Attention Mechanism (SA)

After applying the dynamic convolutional layer to capture the spatial auto-correlations of urban traffic, we are seeking a way to capture the temporal dependencies. Self-attention mechanism [26], which is mostly used in Seq2Seq models, achieves excellent performance when dealing with language modeling and machine translation problems. Self-attention mechanism handles sequential data including text, audios and videos, and learn the temporal dependencies from it. Compared with LSTM and GRU, self-attention mechanism is computed in parallel, and thus requires less time to train and results in higher training quality.

The input and output of a self-attention layer are two sequences of vectors. In the self-attention process, each vector in the input sequence is linearly transformed into three vectors called query, key and value. Each output vector is computed as a weighted sum of all the values, where the weights are the outputs of a softmax layer, and the inputs of the softmax layer are scaled dot products of the corresponding query with all keys. Since a sequence of queries, keys and values can be combined in matrices form $Q$, $K$ and $V$ and computed in parallel, the self-attention function is calculated in Eq. 2.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^T/\sqrt{d_k}\right)V \tag{2}$$

where $d_k$ is the dimension of $K$.

In this work, we apply multi-head self-attention mechanism, where the queries, keys and values are linearly transformed $h$ times and thus we get $h$ different attentions, which are then concatenated together and go through a linear transformation to get the final

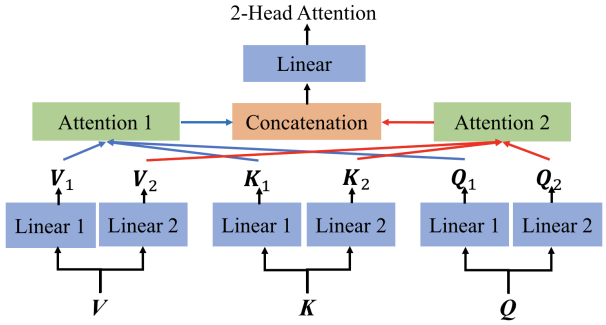



**Figure 6: Example of 2 heads attention.**

values, the multi-head self-attention is calculated with Eq. 3.

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, \ldots, \text{head}_\text{h}\right) W^O \\ \text{where head}_\text{i} &= \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right) \end{aligned} \tag{3}$$

where $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}, W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}, W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$ are parameter matrices, $d_v$ is the dimension of $V$ and $d_\text{model}$ is the dimension of the outputs. Figure 6 shows an example of 2-head attention.

## 3.3 Curb-GAN Architecture

To provide daily consecutive traffic estimations conditioned on expected travel demands, we employ the conditional GAN (cGAN) structure to make it possible to control the estimations by different travel demands. Figure 3 shows the overall structure of Curb-GAN. Curb-GAN contains a generator $G$ and a discriminator $D$. The generator $G$ aims to generate sequences of traffic distributions in consecutive time slots which are similar to the real ones so that the discriminator $D$ cannot distinguish the generated traffic distribution sequences from the real sequences well.

**The generator** $G$ aims to generate daily sequential traffic distributions with respect to the daily travel demand sequence $\mathcal{D}^R$ in a specific region. The input of the generator $G$ includes three parts, i) a noise tensor $\mathcal{Z} = \{z_1, \ldots, z_{N_t}\} \in \mathbb{R}^{N_t \times N_s \times N_s}$, randomly sampled from Gaussian distribution, ii) a condition tensor $C^R = \{C_1^R, \ldots, C_{N_t}^R\} \in \mathbb{R}^{N_t \times N_s \times 4}$, where $C_t^R$ is a matrix of size $N_s \times 4$ defining the region location of $R$, travel demand and current time slot, *i.e.*, $C_t^R = \text{Repeat}(\text{Concat}(i, j, d_t^R, t))$, where $(i, j, d_t^R, t)$ are concatenated to one vector and repeat for $N_s$ times to form the matrix $C_t^R$, and iii) a traffic correlation matrix tensor $\mathcal{A}^R$. In generator, $C^R$ is concatenated into $\mathcal{Z}$ and it builds the mapping from distribution $p_\mathcal{Z}(\mathcal{Z})$ to a traffic distribution $G(C^R, \mathcal{A}^R, \mathcal{Z})$.

**The discriminator** $D$ tries to rise the output score if the input is real traffic distribution sequence, and lower down the score if the input is generated traffic distribution sequence. $D$ takes three inputs, i) a one day traffic distribution tensor $\mathcal{M}^R$, ii) a condition tensor $C^R$ and iii) a traffic correlation matrix tensor $\mathcal{A}^R$. $D$ outputs a scalar indicating whether the input traffic distribution tensor $\mathcal{M}^R$ is real and whether the input $\mathcal{M}^R$ and $C^R$ are matched. The detailed structures of generator $G$ and discriminator $D$ are illustrated in Figure 3(a) and Figure 3(b).

As a result, the loss function of Curb-GAN is in the form of Eq. 4, modeled as a Min-Max game with an additional L2 penalty. (See

more details in [18].)

$$\min_G \max_D V(D,G) = E_{\mathcal{M}\sim p_{data}(\mathcal{M})}[\log D(C,\mathcal{A},\mathcal{M})] + E_{\mathcal{Z}\sim p_{\mathcal{Z}}(\mathcal{Z})}[\log(1-D(G(C,\mathcal{A},\mathcal{Z})))]. \quad (4)$$

Inside the generator and discriminator, we apply dynamic convolutional layer and self-attention mechanism, which help to capture the spatio-temporal auto-correlations. As shown in Figure 3, there are two building blocks inside $G$ and $D$ – Building Block 1 and Building Block 2, both can be stacked for several times.

**Building Block 1** is composed of DyConvs followed by batch normalization and activation functions like ReLU or LeakyReLU. In Building Block 1, the number of DyConvs is equal to $N_t$, and all DyConvs can share the parameters.

**Building Block 2** is composed of a multi-head self-attention layer and a feed-forward network composed of two fully-connected layers activated by ReLU. Both the self-attention layer and the feed-forward network are followed by an addition operation and a layer normalization [26].

**Algorithm 1** Curb-GAN Training Process

**Input:** Training iteration $k$, a training set $\mathcal{P}$, initialized $G$ and $D$.
**Output:** Well trained G and D.

1: In each training iteration $iter$:
2: **repeat**
3: Sample $\mathcal{P}_0$ from training set $\mathcal{P}$.
4: Sample $\mathcal{B}$ from Gaussian distribution.
5: Generate $\tilde{O}$ with $G$.
6: Sample $\hat{O}$ from training set $\mathcal{Z}$.
7: Update $D$ with Eq. 6 to maximize Eq. 5.
8: Update $G$ with Eq. 8 to maximize Eq. 7.
9: **until** $iter > k$.

## 3.4 Curb-GAN Training

During the training process, we apply BPTT (backpropagation through time). The detailed training process is shown in Algorithm 1, where the discriminator $D$ and the generator $G$ are updated in line 3 – 7 and line 8, respectively. Denote the training set which contains $n$ samples as $\mathcal{P} = \{(C^1,\mathcal{A}^1,\mathcal{M}^1),\cdots,(C^n,\mathcal{A}^n,\mathcal{M}^n)\}$, Denote $\mathcal{P}_0 = \{(C^1,\mathcal{A}^1,\mathcal{M}^1),\cdots,(C^m,\mathcal{A}^m,\mathcal{M}^m)\}$ (line 3) as a subset of $\mathcal{P}$ containing $m$ samples, where $m < n$. Denote $\mathcal{B} = \{\mathcal{Z}^1,\mathcal{Z}^2,\cdots,\mathcal{Z}^m\}$ as a set of $m$ noise tensors sampled from Gaussian distribution (line 4), $\tilde{O} = \{\tilde{\mathcal{M}}^1,\cdots,\tilde{\mathcal{M}}^m\}$ as a set of $m$ traffic distribution tensors generated with $G$ (line 5), where $\tilde{\mathcal{M}}^i = G(C^i,\mathcal{A}^i,\mathcal{Z}^i)$. Denote $\hat{O} = \{\hat{\mathcal{M}}^1,\hat{\mathcal{M}}^2,\cdots,\hat{\mathcal{M}}^m\}$ as a set of $m$ traffic distribution tensors sampled from the training set $\mathcal{P}$ (line 6), each $\hat{\mathcal{M}}^i$ is mismatched with $(C^i,\mathcal{A}^i)$. In each training iteration, we update the parameters $\boldsymbol{\theta}_D$ of $D$ with Eq. 5 and Eq. 6, where $\eta_D$ is the learning rate.

$$\tilde{V}_D = \frac{1}{m}\sum_{i=1}^{m}\Big(\log(1-D(C^i,\mathcal{A}^i,\tilde{\mathcal{M}}^i)) + \log D(C^i,\mathcal{A}^i,\mathcal{M}^i) + \log(1-D(C^i,\mathcal{A}^i,\hat{\mathcal{M}}^i))\Big), \quad (5)$$

$$\boldsymbol{\theta}_D = \boldsymbol{\theta}_D + \eta_D \nabla \tilde{V}_{\boldsymbol{\theta}_D}(\boldsymbol{\theta}_D). \quad (6)$$

Then, we update the parameters $\boldsymbol{\theta}_G$ of $G$ with Eq.7 and Eq.8, where $\eta_G$ is the learning rate.

$$\tilde{V}_G = \frac{1}{m}\sum_{i=1}^{m}\log D(G(C^i,\mathcal{A}^i,\mathcal{Z}^i)), \quad (7)$$

$$\boldsymbol{\theta}_G = \boldsymbol{\theta}_G + \eta_G \nabla \tilde{V}_{\boldsymbol{\theta}_G}(\boldsymbol{\theta}_G). \quad (8)$$

After training, we use the well-trained generator to generate the estimated traffic distributions in consecutive time slots of target regions with expected travel demand sequences.

# 4 EVALUATION

In this section, We first describe the two real-world spatio-temporal datasets and then introduce baselines and the evaluation metrics. Finally, we present and analyze our experiment results in detail.

## 4.1 Dataset Descriptions

We validate the effectiveness of our model on two real-world data sets: (1) traffic speed and (2) taxi inflow.

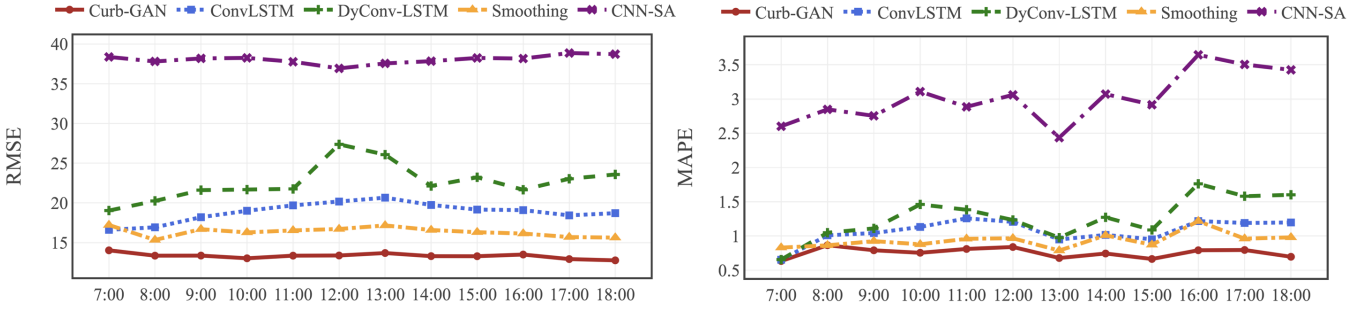
- **Traffic speed.** The hourly average traffic speed is extracted from GPS records collected from taxis in Shenzhen, China from Jul 1st to Dec 31st, 2016. In this estimation task, we first partition Shenzhen City into $40 \times 50$ grid cells. The traffic status in each grid cell is measured by average traffic speed, and there are 4416 time slots (*i.e.*, one hour) over 6 months. Then for each time slot (*i.e.*, one hour), we obtain traffic distributions and travel demands of training regions, and use the daily traffic distribution sequences and travel demand sequences of training regions to train the model. The details of training region selection are in 4.4 and Appendix A. The goal of this task is to estimate the traffic distribution sequence $\hat{\mathcal{M}}^R$ of a test region $R$ conditioned on the expected travel demand sequence $\hat{\mathcal{D}}^R$.
- **Taxi inflow.** The taxi inflow data is collected from taxis in Shenzhen, China from July 1st to Dec. 31st, 2016. In each time slot (*i.e.*, one hour) of each day, the taxi inflow is the count of all taxis that stayed or arrived at each grid cell. In this estimation task, the entire Shenzhen City is partitioned into $40 \times 50$ grid cells, and the traffic status in each grid cell is measured by taxi inflow. With the knowledge of $\mathcal{D}$ and $\mathcal{M}$ for all training regions, for a specific test region $R$, given the expected travel demand sequence $\hat{\mathcal{D}}^R$, we aim at estimating the traffic distribution sequence $\hat{\mathcal{M}}^R$.

## 4.2 Baselines

- **Spatial smoothing with neighboring regions [8].** In each time slot, this method uses the traffic distributions of 9 closest regions under the same travel demand to compute a mean distribution as the resulting estimation. Note we only use available data in the training set to estimate and we will ignore a neighboring region if its data is not available for this travel demand.
- **ConvLSTM [19, 21].** This method uses conditional GAN structure, and applies ConvLSTM inside both generator and discriminator to provide sequential estimated traffic distributions.
- **FC-SA [19, 26].** This method uses conditional GAN structure, and applies stacked fully-connected layers and self-attention layers inside both generator and discriminator.

**Table 2: Performance results on traffic speed estimation and taxi inflow estimation.**

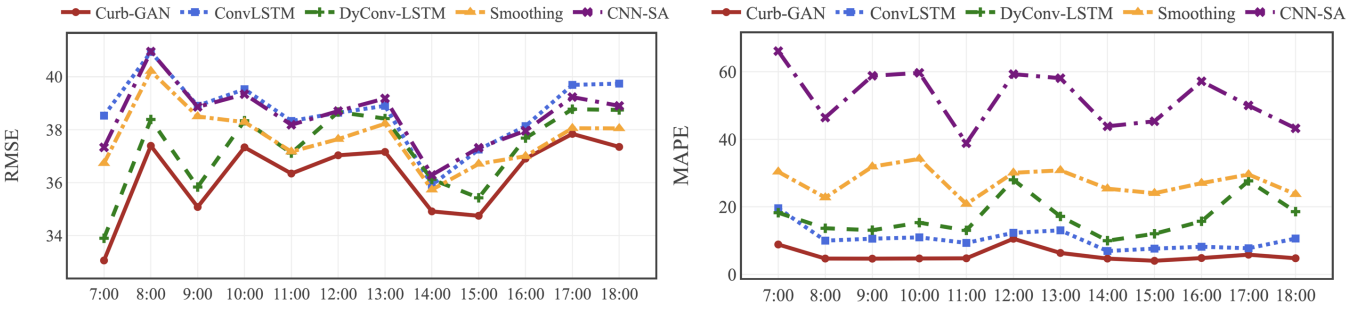
| Methods | | Smoothing | ConvLSTM | FC-SA | CNN-SA | FC-LSTM | CNN-LSTM | DyConv-LSTM | **Curb-GAN** |
|---|---|---|---|---|---|---|---|---|---|
| Traffic speed | RMSE | 16.37 | 18.90 | 44.30 | 38.06 | 128.03 | 30.15 | 22.72 | **13.34** |
| | MAPE | 0.94 | 1.07 | 3.44 | 3.02 | 3.70 | 2.27 | 1.26 | **0.76** |
| Taxi inflow | RMSE | 37.71 | 38.73 | 40.30 | 38.54 | 41.11 | 38.20 | 37.33 | **36.29** |
| | MAPE | 27.56 | 10.43 | 79.75 | 52.25 | 36.92 | 62.02 | 16.88 | **5.88** |




(a) RMSE of selected models from 7:00 to 19:00

(b) MAPE of selected models from 7:00 to 19:00

**Figure 7: Comparisons of selected models in consecutive time slots in traffic speed estimation.**




(a) RMSE of selected models from 7:00 to 19:00

(b) MAPE of selected models from 7:00 to 19:00

**Figure 8: Comparisons of selected models in consecutive time slots in taxi inflow estimation.**

- **CNN-SA [19, 30].** This method uses conditional GAN structure and applies stacked standard convolutional layers and self-attention layers inside generator and discriminator.
- **FC-LSTM [19, 38].** This method uses conditional GAN structure and applies stacked fully-connected layers and multi-layer LSTM inside generator and discriminator.
- **CNN-LSTM [6, 19].** This method uses conditional GAN structure and applies stacked standard convolutional layers and multi-layer LSTM inside generator and discriminator.
- **DyConv-LSTM [19, 37].** This method uses conditional GAN structure and applies stacked dynamic convolutional layers and multi-layer LSTM inside generator and discriminator.

## 4.3 Evaluation Metrics

We use mean absolute percentage error (MAPE) and rooted mean square error (RMSE) to evaluate Curb-GAN:

$$\text{MAPE} = \frac{1}{N_s N_t} \sum_{s=1}^{N_s} \sum_{t=1}^{N_t} \left| y_{s,t} - \hat{y}_{s,t} \right| / y_{s,t} \tag{9}$$

$$\text{RMSE} = \sqrt{\frac{1}{N_s N_t} \sum_{s=1}^{N_s} \sum_{t=1}^{N_t} \left( y_{s,t} - \hat{y}_{s,t} \right)^2} \tag{10}$$

where $y_{s,t}$ is the ground-truth traffic status observed in the $s$-th grid cell and $t$-th time slot, and $\hat{y}_{s,t}$ is the corresponding prediction.

## 4.4 Experimental Settings

The whole Shenzhen city is divided to $40 \times 50$ grid cells with a side-length $l_1 = 0.0084°$ in latitude and $l_2 = 0.0126°$ in longitude. Each region is of size $10 \times 10$, *i.e.*, $\ell = 10$ and $N_s = 100$. Thus, there are in total $1,271$ possible target regions with size $10 \times 10$. However, it is unnecessary and too costly to use data from all $1,271$ regions to train the model. Instead, we select 63 regions covering entire Shenzhen city as target regions for training, extract their traffic distributions and travel demands over time, and use the rest of regions for testing. The more information of training region selection is in Appendix A.

The daily time interval for the data used to train all the models are from 7:00am to 7:00pm, where each hour is a time slot and we have 12 time slots per day, *i.e.*, $N_t = 12$. Thus, the sequence lengths of $\mathcal{D}^R$, $\hat{\mathcal{D}}^R$, $\mathcal{M}^R$, $\hat{\mathcal{M}}^R$ and $\mathcal{A}^R$ are 12.

To extract the travel demands, in each time slot of a day, *i.e.*, one hour, we count the total taxi pickup events within each grid cell and each region. In general, it is hard to obtain the total travel demand in a region including all transport modes. In this work, we use the demand for taxis to represent the regional travel demand, where many studies have shown that taxi demands represent the total demands quite well [9, 20].

The structure of Curb-GAN is as follows: the Building Block 1 is stacked for 4 times, the Building Block 2 is staked 3 times, where

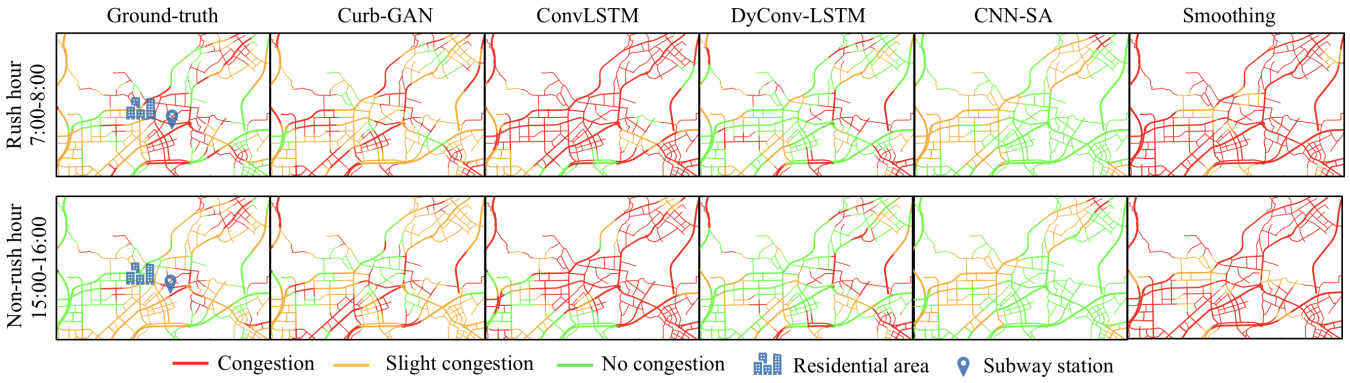



**Figure 9: Traffic status visualizations.**

2-head self-attention is used. The initial input feature of building block 1 in generator is 104, the hidden features of stacked building block 1 are {64,32,16,1}. The initial input feature of building block 1 in discriminator is 5, the hidden features of stacked building block 1 are {16,32,64,1}. Curb-GAN is trained using Adam optimizer [14] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 1500 epochs with a batch size of 64.

## 4.5 Results

*4.5.1 Average performance results.* The performances of the competing baselines and Curb-GAN are shown in Table 2. For each dataset, we randomly pick one test region to calculate RMSE and MAPE with $N_t = 12, N_s = 100$, and similar results are got for other test regions. For deep models, we train and test each of them five times, the statistics are calculated using average generation results conditioned by the same travel demand sequence as the ground-truth.

In traffic speed estimation, Curb-GAN outperforms all the baselines on both metrics. Specifically, Curb-GAN shows 52.77% and 55.38% improvements on RMSE and MAPE beyond all baselines on average, respectively. Compared with FC-LSTM, FC-SA, CNN-LSTM and CNN-SA, Curb-GAN achieves significant improvements, because it explicitly models the relationships between different locations using DyConv. Smoothing seems to have low RMSE by simply averaging the traffic distributions of nearby regions, but it produces higher MAPE, which indicates bad spatio-temporal auto-correlations learned since the traffic of nearby regions cannot provide accurate estimates for the target region.

DyConv-LSTM simultaneously captures the spatial and temporal auto-correlations and it uses LSTM to handle the temporal dependencies, but its estimations are not as good as Curb-GAN due to the limitations of model expressiveness and non-parallel computations of LSTM and thus lead to more training time and lower generation quality.

In taxi inflow estimation, similar to traffic speed estimation, Curb-GAN significantly outperforms the baseline models by 6.48% and 77.63% improvements on average on RMSE and MAPE, respectively. The most competitive models are smoothing, ConvLSTM and DyConv-LSTM, but Curb-GAN can better learns the spatio-temporal patterns and thus obtain lower errors.

*4.5.2 Performance in consecutive time slots.* Since Curb-GAN is able to provide traffic estimations in consecutive time slots, to illustrate the effectiveness of Curb-GAN in traffic estimations in each time slot, we conduct experiments on Curb-GAN and four most competitive baseline models including ConvLSTM, DyConv-LSTM, CNN-SA and smoothing. The statistics are calculated in each time slot with $N_t = 1, N_s = 100$ using the average generation results of a specific test region based on the same travel demand sequence as the ground-truth.

In traffic speed estimation, as shown in Figure 7(a) and Figure 7(b), the Curb-GAN has the best performance in each hour from 7:00 to 19:00. In taxi inflow estimation, we got similar results in Figure 8(a) and Figure 8(b) that Curb-GAN outperforms the other four competitive baselines in both metrics from 7:00 to 19:00. These evaluations prove that the Curb-GAN can better capture the spatio-temporal auto-correlations and thus has a stable and excellent ability in estimating urban traffic in consecutive time slots.

*4.5.3 Traffic estimation visualizations.* To clearly validate the estimated traffic by Curb-GAN against the ground-truth, we visualize the traffic distributions over the road networks. As shown in Figure 9, we pick two time slots of a day, *i.e.*, rush hour (7:00-8:00) and non-rush hour (15:00-16:00), and visualize the traffic status on the road map of a specific region in Shenzhen. The ground-truth visualizations are compared with the estimation visualizations of Curb-GAN and the other four competitive baseline models. Due to page limit, we only use traffic speed to measure the traffic status but we got similar results using taxi inflow.

In Figure 9, there are obvious traffic changes around the residential area and subway station (marked) between rush and non-rush hours in ground-truth visualizations. However, Smoothing, CNN-SA and DyConv-LSTM did not capture such spatio-temporal auto-correlations very well, as there is no traffic changes between the two hours, and the traffic status along the roads is different from the ground-truth. Even though smoothing and DyConv-LSTM got very competitive statistic results in Table 2 and Figure 7, they still cannot produce good estimations due to bad spatio-temporal patterns learned. ConvLSTM shows some traffic changes between two hours, but it produces worse spatial patterns compared with Curb-GAN. By contrast, our Curb-GAN generates reasonable traffic changes around residential area and subway station between rush and non-rush hours, which suggests that Curb-GAN can capture the spatial and temporal auto-correlations of traffic very well and produce more reliable traffic estimations then all baselines.

**Table 3: Variants of Curb-GAN evaluations.**

| Methods | | 2DyConv+1SA | 3DyConv+1SA | 4DyConv+1SA | 4DyConv+2SA | 4DyConv+3SA |
|---|---|---|---|---|---|---|
| Traffic speed | RMSE | 219.10 | 19.43 | 17.73 | 20.67 | **13.34** |
| | MAPE | 5.44 | 1.09 | 0.91 | 0.95 | **0.76** |
| Taxi inflow | RMSE | 62.05 | 59.98 | 41.66 | 37.03 | **36.29** |
| | MAPE | 138.21 | 33.15 | 28.16 | 16.85 | **5.88** |

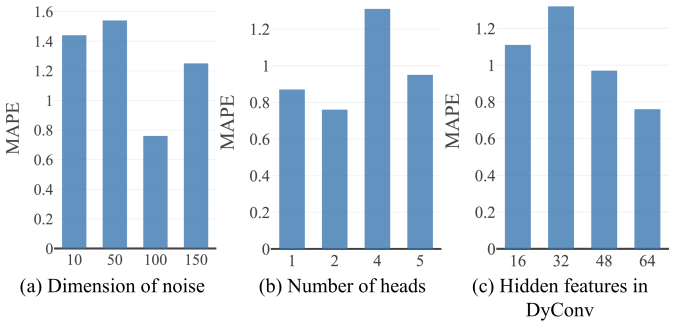



**Figure 10: Impact of parameters in traffic speed estimation.**

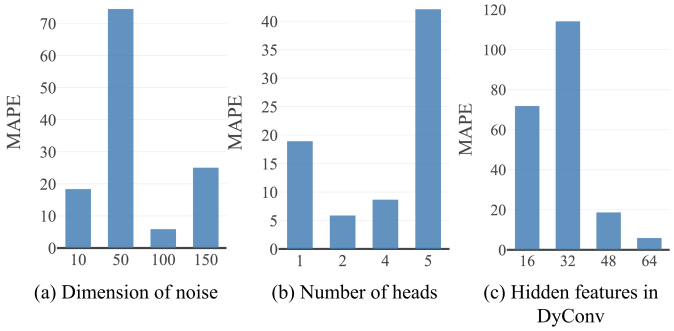



**Figure 11: Impact of parameters in taxi inflow estimation.**

*4.5.4 Evaluations on Curb-GAN parameters.* Curb-GAN has many settings including the number of stacked layers of Building Block 2 and Building Block2, initial dimension of noise, the number of heads in self-attention mechanism, *etc.* To investigate the robustness of Curb-GAN, we present the results under various parameter settings in both tasks.

Since the number of stacked Building Block 1 (DyConv layers) and Building Block 2 (self-attention layers) inside generator and discriminator could influence the final estimation results, we evaluate the Curb-GAN with 2,3,4 stacked layers of Building Block 1 and 1,2,3 stacked layers of Building Block 2. The evaluations results are shown in Table 3. In both tasks, the more layers of Building Block 2 (self-attention) we use, the lower error we get in both metrics. It is because more layers of self-attention can better learned the temporal dependencies of traffic. When the number of DyConv layers increases from 2 to 4, the errors significantly decrease, which indicates too few of DyConv layers are not enough to capture the spatial auto-correlations, the structure of Curb-GAN should be adjusted to get the best estimation results for different datasets.

Next we test the impact of dimension of noise (See Figure 10(a) and Figure 11(a)), where the errors are both high when the noise dimension is too low or too high. With low dimension of noise, the fewer number of weights in DyConv is not enough to learn the spatial patterns, and it would need more time and training epochs to get good estimation results if the noise dimension is too high.

Then we test the impact of the head numbers in self-attention layers, Figure 10(b) and Figure 11(b) show the model performance with different number of heads and the same training epochs. The model with 2-head self-attention has better performance than the model with 1-head does, but with the number of heads increasing, the errors keep increasing. This indicates that more heads would record different temporal dependencies including local and long-term dependencies, but too many heads would weaken the capability of capturing the effective information and lead to higher errors.

We also change the number of hidden features in DyConv. The results of two tasks are shown in Figure 10(c) and Figure 11(c). Since we apply four layers of DyConv (four stacked Building Block 1) in traffic speed estimation, here we change the number of hidden feature of the first DyConv layer. We find that the model performance is sensitive to the hidden features, more hidden features in DyConv lead to better performance, which indicates more weights in DyConv can better capture the spatial patterns of traffic.

## 5 RELATED WORK

**Urban Traffic Prediction.** Previous works focused on urban traffic prediction from different perspectives. There are some previously published works focusing on predicting an individual's movement based on their location history such as [7, 22]. They mainly forecast millions of individuals' mobility trajectories rather than the aggregated traffic conditions in a region. Some other researchers aimed to predict travel speed and traffic volume on roads. For example, [35] proposed a hybrid framework that integrated both state-of-art machine learning techniques and well-established traffic flow theory to estimate citywide traffic volume. In [16] and [25], the authors developed models to predict the road traffic volume and crowd flows in subway stations. These work assumed unchanged urban settings and predict the traffic volume over time and locations. Traditionally in civil engineering, agent-based simulation models [13] or physical models [23] were used to estimate the projected traffic volume after constructions. However, these models rely heavily on model choice and parameter settings, which are not transferable across urban regions. In our work, we focus on studying the spatio-temporal auto-correlations of traffic and predict regional traffic based on different local travel demands.

**Deep Learning for Spatio-Temporal Prediction.** Deep learning methods have inspired many spatio-temporal applications. For example, CNNs were widely used in grid data modeling like citywide flow prediction [36] and taxi demand inference [29], since it can capture the spatial auto-correlations and thus provide good traffic estimations. Besides, RNNs [4] were also widely applied in spatio-temporal prediction problems due to their success in sequence learning. For example, [28] and [27] applied RNN to tackle video prediction and travel time estimation problem, respectively. In addition, [40] used ConvLSTM to to predict crowd density which

captured temporal and spatial dependencies simultaneously and [31] used the combination of CNN and LSTM to predict the road traffic speed. Yuan et al. [34] proposed to use a variation of the ConvLSTM model to predict traffic accidents. Huang et al. [12] employed a deep attention model to predict crimes. Li et al. [15] employed a reinforcement learning method to dynamically reposition shared bikes. However, all these above studies are only trying to capture the spatial and temporal auto-correlations of traffic, they did not consider the impact of travel demand changes. In our study, we study the impact of travel demand changes and spatio-temporal auto-correlations simultaneously.

## 6 CONCLUSION

This paper proposed and investigated a novel conditional traffic estimation problem, namely, estimating the impact of travel demands on regional traffic status. Solving this problem is crucial to potentially avoid traffic issues caused by sudden greatly improved travel demands, *e.g.*, emergencies and new urban constructions. In this paper, a novel generative model - Curb-GAN was proposed to estimate urban traffic based on various travel demands. Curb-GAN is capable of modeling both spatial and temporal auto-correlations and producing a sequence of estimated traffic in consecutive time slots. Using real-world spatio-temporal traffic data, we evaluated our Curb-GAN on two datasets. The well-trained generator is capable of generating realistic traffic distribution sequences in a region given a not-yet-observed travel demand sequence, and our proposed Curb-GAN significantly outperforms all baseline models.

## ACKNOWLEDGMENTS

Yingxue Zhang and Yanhua Li were partly supported by NSF grants IIS-1942680 (CAREER), CNS-1657350 (CRII), and CMMI-1831140. Xun Zhou was partially supported by a grant from the SAFER-SIM University Transportation Center.

## REFERENCES

[1] 2020. Curb-GAN. https://github.com/Curb-GAN/Curb-GAN. [Online].
[2] Dina Al-Shibeeb. 2019. *Vaughan council rejects Sports Village expansion after 4-year debate.* YorkRegion.
[3] Pablo Samuel Castro, Daqing Zhang, and Shijian Li. 2012. Urban Traffic Modelling and Prediction Using Large Scale Taxi GPS Traces. In *Pervasive*.
[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* (2014).
[5] Zhiyong Cui, Ruimin Ke, and YinhaiWang. 2017. Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction. In *6th International Workshop on Urban Computing*.
[6] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2014. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *CoRR* (2014).
[7] Zipei Fan, Xuan Song, Ryosuke Shibasaki, and Ryutaro Adachi. 2015. CityMomentum: An Online Approach for Crowd Behavior Prediction at a Citywide Level. In *ACM UbiComp*.
[8] Arthur Getis. 2008. A History of the Concept of Spatial Autocorrelation: A Geographer's Perspective. *Geographical Analysis* 40 (2008).
[9] Eric J. Gonzales, Ci (Jesse) Yang, Ender Faruk Morgul, and Kaan Ozbay. 2014. *Modeling Taxi Demand with GPS Data from Taxis and Transit.* Technical Report. Mineta National Transit Research Consortium.
[10] Jan Hauke and Tomasz Kossowski. 2011. Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data. *Quaestiones Geographicae* (2011).
[11] Ryan Herring, Aude Hofleitner, Pieter Abbeel, and Alexandre M. Bayen. 2010. Estimating arterial traffic conditions using sparse probe data. In *ITSC*.
[12] Chao Huang, Junbo Zhang, Yu Zheng, and Nitesh V Chawla. 2018. DeepCrime: Attentive Hierarchical Recurrent Networks for Crime Prediction. In *CIKM*.
[13] Benhamza Karima, Salah Ellagoune, Hamid Seridi, and Herman Akdag. 2019. Agent-based modeling for traffic simulation. (June 2019).
[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.
[15] Yexin Li, Yu Zheng, and Qiang Yang. 2018. Dynamic Bike Reposition: A Spatio-Temporal Reinforcement Learning Approach. In *KDD*.
[16] Xinyue Liu, Xiangnan Kong, and Yanhua Li. 2016. Collective traffic prediction with partially observed traffic history using location-base social media. In *CIKM*.
[17] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, Fei-Yue Wang, et al. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* (2015).
[18] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. In *CoRR abs/1411.1784*.
[19] Olof Mogren. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *CoRR* (2016).
[20] Naoto Mukai and Naoto Yoden. 2012. Taxi Demand Forecasting Based on Taxi Probe Data by Neural Network. In *IIMSS*.
[21] 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. *Advances in neural information processing systems* (2015).
[22] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, and Ryosuke Shibasaki. 2014. Prediction of Human Emergency Behavior and Their Mobility Following Large-Scale Disaster. In *SIGKDD*.
[23] John Taplin. 2008. Simulation models of traffic flow. (2008).
[24] Waldo R Tobler. 1970. A computer movie simulating urban growth in the Detroit region. *Economic geography* (1970).
[25] Ermal Toto, Elke A. Rundensteiner, Yanhua Li, Richard Jordan, Mariya Ishutkina, Kajal Claypool, Jun Luo, and Fan Zhang. 2016. PULSE: A Real Time System for Crowd Flow Prediction at Metropolitan Subway Stations. In *ECMLPKDD*.
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR* (2017).
[27] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When Will You Arrive? Estimating Travel Time Based on Deep Neural Networks. In *AAAI*.
[28] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S. Yu. 2017. PredRNN: Recurrent Neural Networks for Predictive Learning using Spatiotemporal LSTMs. In *NIPS*.
[29] Huaxiu Yao, Fei Wu, Jintao ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, and Jieping Ye. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. (2018).
[30] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *CoRR* (2018).
[31] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. 2017. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors* (2017).
[32] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. 2011. Driving with knowledge from the physical world. In *KDD*.
[33] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. 2010. T-drive: driving directions based on taxi trajectories. In *GIS*.
[34] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-ConvLSTM: A Deep Learning Approach to Traffic Accident Prediction on Heterogeneous Spatio-Temporal Data.
[35] Xianyuan Zhan, Yu Zheng, Xiuwen Yi, and Satish Ukkusuri. 2017. Citywide Traffic Volume Estimation Using Trajectory Data. *TKDE* (2017).
[36] Junbo Zhang, Yu Zheng, and Dekang Qi. 2016. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. *CoRR* (2016).
[37] Yingxue Zhang, Yanhua Li, Xun Zhou, Xiangnan Kong, and Jun Luo. 2019. TrafficGAN: Off-Deployment Traffic Estimation with Traffic Generative Adversarial Networks. In *ICDM*.
[38] Jiachen Zhao, Fang Deng, Yeyun Cai, and Jie Chen. 2019. Long short-term memory - Fully connected (LSTM-FC) neural network for PM2.5 concentration prediction. *Chemosphere* (2019).
[39] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *TIST* (2014).
[40] Ali Zonoozi, Jung jae Kim, Xiao-Li Li, and Gao Cong. 2018. Convolutional Recurrent Model for Crowd Density Prediction with Recurring Periodic Patterns. In *IJCAI*.

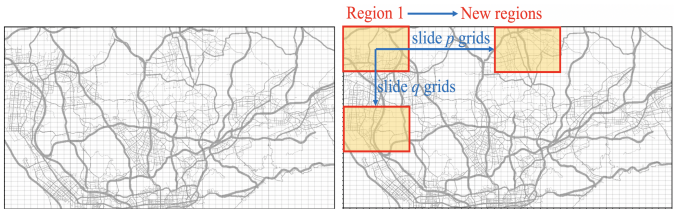
# A APPENDIX FOR REPRODUCIBILITY

To support the reproducibility of the results in this paper, we have released our codes and data[2]. Here, we detail the datasets and the baseline settings.

## A.1 Detailed Settings of Traffic Speed Estimation

**Preprocessing of Dataset**

As shown in Figure 12(a), we apply map gridding method in Shenzhen, China. The whole city is partitioned into $40 \times 50$ grid cells, the target region size is set to $\ell = 10$. Thus, there are in total $1,271$ possible target regions. As shown in Figure 12(b), the upper-left red box is the one region in Shenzhen City, we can slide it for $p$ grid cells in horizontal direction, $0 \le p \le 40$, and/or for $q$ grid cells in vertical direction to get new regions, $0 \le q \le 30, p, q \in \mathbb{N}$. However, since it is too costly to use all the $1,271$ regions as training data. Instead, we set $p = q = 5$ and get 63 regions covering entire Shenzhen city as training regions to train the models.




(a) Road map gridding (b) Region formalization

**Figure 12: Map gridding and regions illustration**

In traffic speed dataset, we extract the hourly average traffic speed for each grid cell in Shenzhen, China from Jul 1st to Dec 31st, 2016. The detailed information of the dataset is shown in Table 4.

**Table 4: Dataset descriptions.**

| Dataset | Traffic speed | Taxi inflow |
|---|---|---|
| **Timespan** | 07/01/2016-12/31/2016 | 07/01/2016-12/31/2016 |
| **Time slot** | 1 hour | 1 hour |
| **City size** | $40 \times 50$ | $40 \times 50$ |
| **Region size** | $10 \times 10$ | $10 \times 10$ |

**Quantifying Traffic Correlation**

Traffic correlation captures the inherent traffic dependencies between two grid cells. We calculate the Pearson correlation coefficient using time series traffic data of two grid cells over the entire study timespan to quantify their traffic correlation. As shown in Eq. 11, *Pearson correlation coefficient* [10] denoted with $a$ measures the linear correlation between $X$ and $Y$:

$$a_{XY} = \frac{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{\sqrt{\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2}\sqrt{\sum_{i=1}^{n}\left(Y_i - \overline{Y}\right)^2}}, \tag{11}$$

where $\overline{X}$ and $\overline{Y}$ are the mean of $X$ and $Y$ and $a \in [-1, 1]$. For an $\ell \times \ell$ region $R$, its traffic correlation matrix $\boldsymbol{A}^R$ is a symmetric $\ell^2 \times \ell^2$ matrix.

[2]https://github.com/Curb-GAN/Curb-GAN

In a road network, the traffic correlations are mostly positive, since if a location has traffic congestion, the nearby locations are very likely to have heavy traffic, such dependencies represent positive traffic correlations. However, the traffic correlations between distant locations are usually low, because they do not have direct road connections and the traffic flows will not influence each other. To remove the negative correlations and the low correlations, a threshold $\lambda \in (0, 1)$ is used, we set $a_{ij} = 0$, if $a_{ij} < \lambda$. In this paper, $\lambda = 0.47$.

Then we perform row normalization using Eq. 12, so that the traffic correlation matrix $\boldsymbol{A}^R$ will not affect the scale of the traffic status matrix in Eq. 1.

$$a_{ij} = \frac{a_{ij}}{\sum_{j=1}^{\ell^2} a_{ij}}. \tag{12}$$

**Settings of Baselines**

- **ConvLSTM.** This method uses conditional GAN structure, inside generator and discriminator, both ConvLSTM and a feed-forward network are followed by an addition operation and a layer normalization, the output of generator goes through a linear transformation and activated by Tanh function, the output of discriminator is activated by Sigmoid. The input channel of ConvLSTM is 4, the hidden channel is 32, the kernel size is $3 \times 3$. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 64.
- **FC-SA.** This method uses conditional GAN structure and applies 4 stacked fully-connected layers and 3 self-attention layers inside generator and discriminator. Each fully-connected layer is followed by batch normalization, the self-attention layer is followed by layer normalization. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-6}$ for 400 epochs with a batch size of 32.
- **CNN-SA.** This method uses conditional GAN structure and applies 4 stacked standard convolutional layers and 3 self-attention layers inside generator and discriminator. Each convolutional layer is followed by batch normalization, the self-attention layer is followed by layer normalization. The input channel of convolutional layer is 4, the output channel is 1, the kernel size is $3 \times 3$. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 64.
- **FC-LSTM.** This method uses conditional GAN structure and applies 4 stacked fully-connected layers and a 2-layer LSTM inside generator and discriminator. Each fully-connected layer is followed by batch normalization, the LSTM is followed by layer normalization. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 4.
- **CNN-LSTM.** This method uses conditional GAN structure and applies 4 stacked standard convolutional layers and a 2-layer LSTM inside generator and discriminator. Each convolutional layer is followed by batch normalization, the LSTM is followed by layer normalization. The input channel of convolutional layer is 4, the output channel is 1, the kernel size is $3 \times 3$. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 64.

- **DyConv-LSTM.** This method uses conditional GAN structure and applies 4 stacked dynamic convolutional layers followed by batch normalization and a 2-layer LSTM followed by layer normalization inside generator and discriminator. The output of generator goes through a linear transformation and activated by Tanh function, the output of discriminator is activated by Sigmoid. The initial input feature of DyConv in generator is 14. The initial input feature of discriminator is 5. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 1000 epochs with a batch size of 16.

## A.2 Detailed Settings of Taxi Inflow Estimation

**Preprocessing of Dataset**

The region selection and traffic correlation extraction are the same as traffic speed estimation. The detailed description of the dataset is shown in Table 4.

**Settings of Baselines**

- **ConvLSTM.** This method uses conditional GAN structure, inside generator and discriminator, both ConvLSTM and a feed-forward network are followed by an addition operation and a layer normalization, the output of generator goes through a linear transformation and activated by Tanh function, the output of discriminator is activated by Sigmoid. The input channel of ConvLSTM is 4, the hidden channel is 32, the kernel size is $3 \times 3$. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 64.
- **FC-SA.** This method uses conditional GAN structure and applies 4 stacked fully-connected layers and 3 self-attention layers inside generator and discriminator. Each fully-connected layer is followed by batch normalization, the self-attention layer is followed by layer normalization. The input dimension of fully-connected layer in generator is 104 and the input dimension in discriminator is 5. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-6}$ for 400 epochs with a batch size of 8.
- **CNN-SA.** This method uses conditional GAN structure and applies 4 stacked standard convolutional layers and 3 self-attention layers inside generator and discriminator. Each convolutional layer is followed by batch normalization, the self-attention layer is followed by layer normalization. The input channel of convolutional layer is 4, the output channel is 1, the kernel size is $3 \times 3$. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 64.
- **FC-LSTM.** This method uses conditional GAN structure and applies 4 stacked fully-connected layers and a 2-layer LSTM inside generator and discriminator. Each fully-connected layer is followed by batch normalization, the LSTM is followed by layer normalization. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 4.
- **CNN-LSTM.** This method uses conditional GAN structure and applies 4 stacked standard convolutional layers and a 2-layer LSTM inside generator and discriminator. Each convolutional layer is followed by batch normalization, the LSTM is followed by layer normalization. The input channel of convolutional layer is 4, the output channel is 1, the kernel size is $3 \times 3$. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 400 epochs with a batch size of 64.
- **DyConv-LSTM.** This method uses conditional GAN structure and applies 4 stacked dynamic convolutional layers followed by batch normalization and a 2-layer LSTM followed by layer normalization inside generator and discriminator. The output of generator goes through a linear transformation and activated by Tanh function, the output of discriminator is activated by Sigmoid. The initial input feature of DyConv in generator is 14. The initial input feature of discriminator is 5. This model is trained using Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and a learning rate of $2 \times 10^{-4}$ for 1000 epochs with a batch size of 16.