Forecasting Traffic with a Convolutional GRU Decoder Conditioned on Adapted Historical Data

Alexander Roitenberg [1] **Lior Wolf** [1 2]

Abstract

We propose a method for performing multi-step temporal prediction of spatial traffic demand. The method is based on encoding the recent history with a convolutional GRU, obtaining a hidden state vector that is used to initialize a second convolutional GRU. The second GRU is conditioned on time-matching historical data, after being adapted to match the scale of the recent history. Our fully convolutional method is much more efficient than the recent methods in the field of traffic prediction and does not need additional data, such as weather or public holiday data. The method outperforms the literature methods by a sizable margin, especially when predicting further into the future.

1. Introduction

The prediction of future spatiotemporal data is a ubiquitous task with very diverse applications, including weather forecasting, agricultural planning, social activity anticipation, and brain-machine interfaces.

In this work, we focus on the problem of predicting demand for public transportation, as well as future drop-off events. Due to the growing popularity of ride-hailing services such as Uber and Didi Chuxing, where accurately anticipating demand can lead to better service at a reduced cost, this problem has attracted considerable attention.

The passengers' traffic is known to be influenced by factors such as public holidays and the weather. However, in order to make our method as generic as possible, we focus on the problem of predicting the future demand based solely on the history of this demand. Two forms of historical data are considered by us as most relevant: (i) the very recent history, and (ii) the historical average for the same time of the day at the same day of the week. To account for the variation between the weeks, we scale the second component in accordance with the ratio between the total activity in recent history and the total activity, for the matching day and time, on an average week.

Our method makes use of dilated convolutions (Yu & Koltun, 2015) and of convolutional gated recurrent units (GRUs) (Nicolas et al., 2015), which are a specific type of a Recurrent Neural Network (RNN). An RNN encoder first represents the recent history as the hidden state of a first GRU. Then, a second GRU is initialized by this representation and runs on the historical data of the time frames that are predicted, up to the end of the prediction window.

We compare our method to the recent deep learning methods, as well as to classical time prediction methods, using all relevant datasets we could obtain. Our results indicate a clear advantage in performance for predicting the next time step, which is the main focus of the current literature. When predicting further into the future, the performance gap becomes significantly larger.

2. Related Work

Traffic prediction has been the topic of many studies and a wide array of methods have been employed over the years. Classical approaches used statistical methods, such as autoregressive integrated moving average (ARIMA) (M. Van Der Voort & Watson, 1996; Williams & Hoel, 2003; Q. T. Tran & Trinh, 2015), Kalman filter (Okutani & Stephanedes, 1984; Y.-J. Wu & Yang, 2016) as well as non-parametric approaches such as K-NN (Davis & Nihan, 1991), historical Average (HA), vector autoregression (VAR) and Gaussian process based (Kamarianakis & Prastacos, 2005; S. Thajchayapong & Garcia-Trevino, Sep. 2010). These methods usually focus on predictions in a narrow region, such as a road segment, or several such segments, and usually cannot take into account complex non-linear spatiotemporal dependencies, and are, therefore, not very well suited for the task of citywide prediction that is the focus of this work.

In recent years, the go-to tool for problems involving non-linear relations has been Artificial Neural Networks (ANNs), that have a proven track record in dealing with non-linear

[1]Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel [2]Facebook AI Research. Correspondence to: Alex Roitenberg <Roitenberg.Alex@gmail.com>.

temporal or sequence predictions (van den Oord et al., 2016), as well as complex non-linear spatial dependencies (He et al., 2016). In light of this success, there have been many attempts to use ANNs for various traffic prediction tasks, such as speed (X. Ma & Wang, 2017), congestion condition (X. Cheng & Xu, Jul. 2018), traffic flow in particular roads or areas (Y. Lv & Wang, Apr. 2015; Manoharan, 2016; R. Yu & Liu, 2017; S. Du & Horng, 2018), and closely related to our work, citywide traffic flow prediction (J. Ke & Chen, Dec. 2017; Zhang et al., 2016; Wang et al., 2017; Zhang et al., 2017; Yao et al., 2018; 2019; Li et al., 2018; Zhou et al., 2018; Chen et al., 2018; Du et al., 2019).

Most of these works, except for (Zhou et al., 2018), also utilize external data, such as weather, holidays and major city events, to improve their accuracy, but our method manages to outperform these methods while relying solely on demand data. Another important difference is that most of these methods are designed for short-term prediction, and are not well-suited for multi-step prediction. To predict further into the future, they would need to feed their output back in, causing the prediction to drift and result in deteriorating performance. In this regard, (Zhou et al., 2018) were among the first to propose a method that is geared toward maintaining performance in a multi-step prediction setting. To this end, they have borrowed the concept of the sequence encoder-decoder from the different spatiotemporal task of video-frame prediction (Nitish Srivastava & Salakhudinov, 2015), where a separate LSTM (Hochreiter & Schmidhuber, 1997) decoder is employed for the purpose of predicting multiple future frames. In a similar manner, we use two convolutional GRU (Nicolas et al., 2015) to serve as the encoder and decoder. This still leaves open the question of the input to the decoder during the prediction. Using previous outputs as inputs to the decoder, or using no input at all would lead to similar performance degradation as with the other methods. A key point here is that unlike video prediction, traffic prediction can utilize inherent regularities that drive the periodic behavior of traffic demand. In (Zhou et al., 2018) attention over representative demands was used in an attempt to tap into those regularities. However, this approach does not take into account the periodic behavior of traffic demand. We take a different approach, using the adapted historical average as input, thereby explicitly capturing coarse spatiotemporal regularities, leaving the fine-tuning to the decoder.

3. Method

Following previous work (Zhang et al., 2016; Yao et al., 2018; Zhou et al., 2018), we define the problem of predicting pickups and drop-offs as a regression problem over two spatial grid maps. We partition an area of interest (e.g. a city) into regions using a rectangular $M \times N$ grid based on latitude and longitude coordinates, where $M = \frac{lat_{max} - lat_{min}}{D}$ and $N = \frac{lon_{max} - lon_{min}}{D}$, with D being the length of each grid cell. Smaller values of D lead to more fine-grained predictions, at the expense of a certain increase in computational cost, dependent on the chosen model. We denote the set of regions comprising the grid map as $G = \{g_{ij} \mid i \in [1, M],\ j \in [1, N]\}$.

Given a set of passenger trips starting from some point p_s at time t_s, and arriving at p_e, at time t_e, let us define the number of pickups and drop-off in region g_{ij} in the time interval $([t, t+1)$ respectively as:

$$x_{ij,t}^{\text{pickup}} = |\{(p_s, t_s) \mid p_s \in g_{ij},\ t_s \in [t, t+1)\}|$$

$$x_{ij,t}^{\text{dropoff}} = |\{(p_e, t_e) \mid p_e \in g_{ij},\ t_e \in [t, t+1)\}|$$

where $|\cdot|$ is the set's cardinality. Combining the pickup map $x_{ij,t}^{\text{pickup}}$ and the drop-off map $x_{ij,t}^{\text{dropoff}}$, we get $X_t \in \mathbb{R}^{2 \times M \times N}$, which we will refer to as the demand tensor.

Given a set of T recent demand tensors $X_{1...T}$, we would like to predict the demand for next R steps $X_{T+1...T+R}$.

Our model follows a general encoder-decoder framework. The encoder consists of several dilated convolutional layers, with the dilation exponentially growing with depth. Using increasingly dilated convolution increases the receptive field of consecutive convolutional layers, allowing the network to learn relationships between spatially distant regions. Increasing the receptive field can also be achieved using strided convolutions, as was done by Zhou et al. (2018), or using a pooling operation. However, these come at the expense of spatial resolution.

Following the convolutional layers, the resulting feature maps for each of the recent time step $1...T$, are fed into an RNN, consisting of one or more layers, in order to extract the temporal demand dynamics. To preserve spatial correspondence between regions, we use a convolutional RNN, that is similar to a regular RNN, but uses convolutional operations instead of multiplication. Specifically, we use L layers of convolutional GRUs (Nicolas et al., 2015), that have been shown to yield similar performance to LSTMs, while using less memory (Chung et al., 2014). A convolutional GRU is governed by the following equations:

$$\begin{aligned} z_t &= \sigma(W_z * X_t + U_z * h_{t-1}), \\ r_t &= \sigma(W_r * X_t + U_r * h_{t-1}), \\ \bar{h_t} &= \tanh(W * X_t + U * (r_t \odot h_{t-1})), \\ h_t &= (1 - z_t) h_{t-1} + z_t \bar{h_t}, \end{aligned}$$

where $*$ denotes a convolution operation, $\odot$ denotes the Hadamard product, σ is the sigmoid non-linearity, W, W_z, W_r and U, U_z, U_r are learnable convolutional kernels. h_t is the hidden state, and has the form of a three-dimensional tensor in order to preserve spatial relationships.

r_t is the reset gate that controls how much the current hidden state will affect the candidate next hidden state, $\bar{h}_t$. z_t is the update gate that determines the relative importance of the previous hidden state and the candidate new hidden state. A multi-layered convolutional GRU uses the hidden state of layer l, h_t^l, as the input to layer $l+1$.

To predict a single step forward, the encoder output is often used directly, foregoing a decoder RNN (Yao et al., 2018; 2019; Du et al., 2019). While these methods work well for a single step, when trying to predict multiple steps, they rely on their previous outputs, leading to predictions quickly drifting from the ground truth and to deteriorating performance. Therefore, to predict farther into the future, we use a decoder RNN. The decoder RNN is fed not with predictions, which will cause a gradually increasing drift, but with modified historical averages.

The historical averages $H_t \in \mathbb{R}^{2\times M\times N}$ are computed per each day of week and time of day, based on the entire training data. Before being fed into the network, the historical averages for the next time steps are adapted, based on the the recent demand history. We use a simple adaptation, scaling the historical average by the recent global demand. Let the total demand in the recent history be defined as

$$S_g = \frac{1}{2MNT} \sum_{t\in[1,T],k\in[1,2],i\in[1,M],j\in[1,N]} X_t(k,i,j).$$

Similarly, we define the global historical demand S_h by averaging, across the four dimensions (t,k,i,j), the T historical averages H_t matching the times of the recent history window:

$$S_h = \frac{1}{2MNT} \sum_{t\in[1,T],k\in[1,2],i\in[1,M],j\in[1,N]} H_t(k,i,j).$$

The adapted historical average $\bar{H}_t$, which is a single tensor of size $2\times M\times N$ for every t, is then obtained by multiplying the historical average of times $T+1 \le t \le T+R$ by the global scaling $\bar{H}_t = \frac{S_g}{S_h} H_t$. Although this method of adaptation is very simple and computationally cheap, it captures the inherent demand regularity well, and the adapted historical average is a strong predictor in its own right, as can be seen in Sec. 4; furthermore, its error can remain stable for many steps forward. This provides the decoder with a strong baseline to work off of.

We, therefore, use the adapted historical average as input to the decoder at each prediction time step, after processing it using a similar CNN as the one used in the encoder. The decoder further mirrors the architecture of the encoder, using L layers of convolutional GRU, initialized using the last state of the encoder, after the latter has consumed the recent demands $t = 1..T$, see Fig. 1.

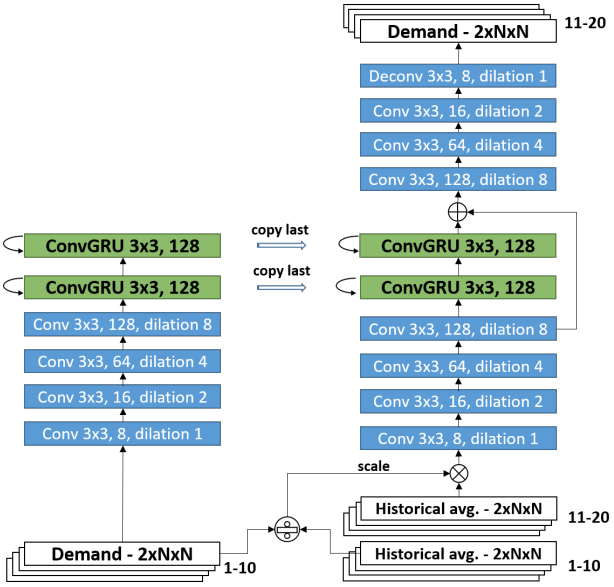

Figure 1. Our architecture for $T = R = 10$. The demand of the recent history is fed to an encoder (left column). The hidden states of the encoder's convolutional GRU layers are then used to initialize the hidden states of the matching layers of the convolutional GRU of the decoder (right column). The decoder's convolutional GRU receives as input the adapted historical averages.

It has been demonstrated repeatedly, following He et al. (2016), that neural networks are significantly better at learning residuals, rather than absolutes. The output of the decoder's convolutional GRU is, therefore, treated as a residual, by summing it with the output of the previous steps, followed by a ReLU non-linearity. Finally, the result is fed into a CNN that is the mirror image of the encoder CNN, performing dilated convolutions with decreasing dilations.

4. Experiments

We test our method using three real-life datasets: (i) BikeNYC[1] is a dataset compiled from 5.5M Citi Bike bicycle rentals and returns between Apr. 1st and Sep. 30th 2014, where the last ten days are used for testing and the rest for training. The data is accumulated in one-hour intervals into a 16 x 8 grid, with each cell being approximately 1km x 1km in size. (ii) YellowTaxiNYC[2] is a dataset of 1B NYC yellow taxi pickups and drop-offs between 2009 and 2015, where 2015 is used for testing and the rest for training. The data is accumulated in one-hour intervals into a 64 x 64 grid, with each cell being approximately 5km x 5km in size. (iii) BJ Taxi is a dataset of 240M taxi pickups and drop-offs in Beijing in the periods Jul. 1st – Oct. 30th 2013, Mar.

[1] www.citibikenyc.com/system-data

[2] www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

1st – Jun. 30th 2014, Mar. 1st – Jun. 30th 2015, and Nov. 1st 2015 – Apr. 10th 2016. The last four weeks are used for testing and the rest for training. The data is given in 30-minute intervals on a 32 x 32 grid.

In the encoder, we use four convolutions with 8, 16, 64, and 128 kernels, dilated respectively by 1, 2, 4, and 8, using padding to keep the spatial size constant, followed by two layers of convolutional GRU with 128 input and output channels. For the decoder, we use two layers of convolutional GRU that are the same as in the encoder, followed by four layers of transposed convolution with 128, 32, 8, and 2 kernels, dilated respectively by 8, 4, 2, and 1. We use a weekly historical average, meaning the average demand for each time of the week (e.g. Tuesday 8AM-9AM). We tie the weights of the convolutional layers of the encoder and the decoder, that precede the two convolutional-GRUs, i.e., the first four layers of the encoder and the decoder employ the same weights.

In all experiments, we used ten recent time steps ($T = 10$) to predict the next step or steps. The RMSE loss $\sqrt{\frac{1}{R}\sum_{t=T+1}^{T+R}(X_t - \hat{X}_t)^2}$ is minimized during training, without any regularization term. We train two models per dataset, one with $R = 1$ and one with $R = 10$. The model trained for $R = 1$ is slightly better that the other model for predicting the next time step and significantly worse, when evaluated on the $R = 10$ scenario (the two models drift apart around the sixth time step).

The models were trained on a single NVidia GeForce GTX 1080 Ti GPU. Adam optimization (Kingma & Ba, 2015) was used with a learning rate of 2e-4, and mini-batches of size 16. The implementation was written using pytorch.

We compare our method to classic time series regression methods: (1) Historical Average (HA): each grid point is predicted based on the historical average for that point at the same time of the week (e.g. Tuesday 9AM-10AM), (2) Adaptive Historical Average (AHA): similar to HA, but each point is scaled by the average of the values in the recent time divided by the average of the historical average for the same period, and (3) AutoRegressive Integrated Moving Average (ARIMA). Additionally, we compare with recent neural network based methods: (4) DeepSD (Wang et al., 2017), (5) Deep spatiotemporal Residual Networks (ST-ResNet) (Zhang et al., 2017), (6) Deep Multi-View Spatial-Temporal Network (DMVST-Net) (Yao et al., 2018), (7) Attention Convolutional LSTM (AttConvLSTM) (Zhou et al., 2018), (8) Diffusion Convolutional RNN (DCRNN) (Li et al., 2018), (9) Deep Irregular Convolutional Residual LSTM (DST-ICRL) (Du et al., 2019), and (10) MST3D (Chen et al., 2018).

We note that training our method is much more efficient than training AttConvLSTM, due to the greatly reduced number of parameters. Among many architectural differences, this baseline method includes multiple fully connected layers, while our methods relies only on convolutional layers.

Table 1. Comparison of different methods for a single time step prediction, showing RMSE scores. [1]Chen et al. (2018) reported in-flow and out-flow separately, the statistics given are the average of the two. [2]Zhou et al. (2018) did not report results for one time step; the first two results are taken from (Du et al., 2019), the third is by our reimplementation. [3]US = uniform sampling. IS = importance sampling.

METHOD	BIKENYC	TAXIBJ	TAXINYC
HA	5.94	30.67	-
AHA	4.79	19.20	-
ARIMA	10.07	22.78	-
DEEPST	7.43	18.18	-
ST-RESNET	6.33	16.89	-
STDN	6.25	16.61	-
MST3D[1]	5.81	16.05	-
ATTCONVLSTM[2]	7.09	17.41	13.47
DST-ICRL (US)[3]	5.93	14.77	-
DST-ICRL (IS)[3]	5.77	14.07	-
OURS	**4.12**	**13.66**	**10.88**

Table 2. Comparison of RMSE for different methods for the prediction of ten time steps into the future. [1]Zhou et al. (2018) did not report results for TaxiBJ, and this result is by our reimplementation.

METHOD	BIKENYC	TAXIBJ	TAXINYC
HA	6.92	31.90	32.42
AHA	6.60	22.19	15.90
ST-RESNET	10.58	-	43.97
ATTCONVLSTM[1]	7.76	22.79	26.91
OURS	**5.81**	**18.45**	**15.78**

The results for predicting a single future time step are given in Tab. 1. As can be seen, our method significantly outperforms the existing literature methods and achieves state-of-the-art on all three datasets. This is achieved using only pure in-flow and out-flow, while almost all other neural network based models (except AttConvLSTM) utilize one or more forms of external data, such as local events and the historical weather.

In the second set of experiments, we make predictions for multiple future time steps. Specifically, following (Zhou et al., 2018), we predict the next 10 time steps. As can be seen in Tab. 2, our model's performance surpasses the other models, often with a larger margin than for the first time step. This strengthens our claim that using the historical average helps stabilize the future predictions, by providing a strong prior for the network to use as a baseline. This prior is independent of previous predictions, as well as on the number of prediction steps taken.

References

Chen, C., Li, K., Teo, S. G., Chen, G., Zou, X., Yang, X., Vijay, R. C., Feng, J., and Zeng, Z. Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 893–898, 2018.

Chung, J., Gulcehre, C., Cho, K. H., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 2014.

Davis, G. A. and Nihan, N. L. Nonparametric regression and short-term freeway traffic forecasting. *Journal of Transportation Engineering*, pp. 178–188, 1991.

Du, B., Peng, H., Wang, S., Bhuiyan, Z. A., Wang, L., Gong, Q., Liu, L., and Li, J. Deep irregular convolutional residual lstm for urban traffic passenger flows prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

J. Ke, H. Zheng, H. Y. and Chen, X. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *Transportation Research Part C: Emerging Technologies, vol. 85*, pp. 591608, Dec. 2017.

Kamarianakis, Y. and Prastacos, P. Space-time modeling of traffic flow. *New York, NY, USA: Pergamon*, 2005.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *International Conference on Learning Representations (ICLR '18)*, 2018.

M. Van Der Voort, M. D. and Watson, S. Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, pp. 307–318, 1996.

Manoharan, S. Short term traffic flow prediction using deep learning approach. *Ph.D. dissertation, School Comput.*, 2016.

Nicolas, B., Li, Y., Chris, P., and Aaron, C. Delving deeper into convolutional networks for learning video representations. *ICLR 2016*, 2015.

Nitish Srivastava, E. M. and Salakhudinov, R. Unsupervised learning of video representations using lstms. *International Conference on Machine Learning*, pp. 843–852, 2015.

Okutani, I. and Stephanedes, Y. J. Dynamic prediction of traffic volume through kalman filtering theory. *Transp. Res. B, Methodol, vol. 18, no. 1*, pp. 111, 1984.

Q. T. Tran, Z. Ma, H. L. L. H. and Trinh, Q. K. A multiplicative seasonal arima/garch model in evn traffic prediction. *IJCNS*, pp. 43, 2015.

R. Yu, Y. Li, C. S. U. D. and Liu, Y. Deep learning: A generic approach for extreme condition traffic forecasting. *SDM*, 2017.

S. Du, T. Li, X. G. Z. Y. and Horng, S.-J. A hybrid method for traffic flow forecasting using multimodal deep learning. *arXiv preprint arXiv:1803.02099*, 2018.

S. Thajchayapong, J. A. B. and Garcia-Trevino, E. Lane-level traffic estimations using microscopic traffic variables. *Proc. 13thInt. IEEE Conf. Intell. Transp. Syst.*, pp. 1189–1194, Sep. 2010.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Wang, D., Cao, W., Li, J., and Ye, J. Deepsd: Supply-demand prediction for online car-hailing services using deep neural networks. *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 243–254, 2017.

Williams, B. M. and Hoel, L. A. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, pp. 664–672, 2003.

X. Cheng, R. Zhang, J. Z. and Xu, W. Deep transport: Learning spatial-temporal dependency for traffic condition forecasting. *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, pp. 1–8, Jul. 2018.

X. Ma, Z. Dai, Z. H. J. M. Y. W. and Wang, Y. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors, vol. 17, no. 4*, pp. 818, 2017.

Y.-J. Wu, F. Chen, C.-T. L. and Yang, S. Urban traffic flow prediction using a spatio-temporal random effects model. *J. Intell. Transp. Syst., vol. 20, no. 3*, pp. 282–293, 2016.

Y. Lv, Y. Duan, W. K.-Z. L. and Wang, F.-Y. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2*, pp. 865–873, Apr. 2015.

Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P., Ye, J., and Li, Z. Deep multi-view spatial-temporal network for taxi demand prediction. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pp. 2588–2595, 2018.

Yao, H., Tang, X., Wei, H., Zheng, G., and Li, Z. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. *Thirty-third AAAI Conference on Artificial Intelligence*, 2019.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Zhang, J., Zheng, Y., Qi, D., Li, R., and Yi, X. Dnn-based prediction model for spatio-temporal data. *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPACIAL '16)*, pp. 92:1–92:4, 2016.

Zhang, J., Zheng, Y., and Qi, D. Deep spatio-temporal residual networks for citywide crowd flows prediction. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Zhou, X., Shen, Y., Zhu, Y., and Huang, L. Predicting multi-step citywide passenger demands using attention-based neural networks. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*, pp. 736–744, 2018.