CSE 575 Project Report: Traffic Flow Prediction

Name (Group 10)	**ASURITE ID**	**Role**
Rath, Prabin	prath4	Lead, Code, Research, Documentation
Cao, Richard	rscao	Code, Research, Documentation
Gade, Shreyash	sgade13	Code, Research
Tokala, Jaswanth Reddy	jtokala	Code, Research
Vasireddy, Vikas	vvasire2	Code, Research

Introduction

In this project, we present a deep learning-based regression model to predict traffic flow in cities and urban areas. Traffic flow forecasting is crucial as it can help traffic management and crowd control. The prediction task is complex because of various factors, such as inter-region traffic, weather conditions, and local events. Spatiotemporal data analysis is an emerging research area due to the development and application of novel computational techniques allowing for the analysis of large vehicle trajectory databases. Deep Learning has proven to be an efficient technique for solving spatiotemporal data problems. Hence, this project presents a deep-learning model based on the ST-Resnet[1] architecture. We fine-tune the model parameters and introduce minute changes in the original architecture to showcase the results on BikeNYC[2] and TaxiBJ[1] datasets. Additionally, we show the performance of our model on multi-step lookahead predictions and analyze the potential improvement opportunities for the future.
The source code of our implementation for this project can be found at:
https://github.com/prabinrath/Traffic-Flow-Prediction

Previous work

Traffic flow prediction dates back to the 1970s when the flow was predicted using the Autoregressive Integrated Moving Average (ARIMA) technique for short horizon flow prediction. Numerous researchers have also employed statistical and Non-parametric methods such as Kalman Filters, K-NN, Fuzzy Inference, and ANN as prediction tools [3]. It has been observed that Neural Networks (NN) have good prediction power and robustness compared to other approaches in the literature [3]. However, shallow networks and other statistical or non-parametric methods are limited in identifying long temporal relationships within the flow data. To address this issue, Lv, Yisheng, et al. employ the first-ever deep learning model based on Stacked Auto Encoders (SAE) for traffic flow prediction [3]. They use autoencoders with KL Divergence loss as sparsity constraints to pre-train a series of layers, one after another, that can learn to predict the input signal. These stacked autoencoders are followed by a logistic regression layer to predict the traffic flow over a finite time horizon. They show that deep learning approaches can capture long-term temporal relationships in data.

Zhang, Junbo et al. introduced convolutional NNs for capturing spatial relationships within 2D grid flow data for the first time in their paper on Deep-ST [7]. In the following paper, they introduced ST-ResNet (Spatio Temporal Residual Networks) [1], a convolution-based residual network that can incorporate arbitrary deep architectures for capturing temporal relationships from several months of traffic flow data. Additionally, they fused external influences such as occasional storms or festive events, which might significantly affect the traffic flow. Liang, Yuxuan, et al. have proposed UrbanFM [4], a feature extractor CNN followed by a distributional upsampling block to impose structural constraints for improving the traffic flow inference. Lin, Ziqian, et al. have proposed DeepSTN+ [5], a CNN-based approach with a modified ResNet block for capturing long-range spatial relationships within the data. They also introduce a prior Point of Interest (PoI) distribution based on geographical location features which might influence the traffic flow at different times of the day. Saxena, Divya, et al. have used deep adversarial GAN networks to identify Spatiotemporal relationships in an unsupervised manner [6].

Problem Description

Predicting the traffic flow in a particular region in a given city is essential because the prediction would help in controlling the traffic as well as help in controlling the crowd in that specific area. In the past, there have been various incidents where massive crowd gatherings have created fatal incidents. For instance, a massive crowd gathering during New Year's eve in Shanghai in 2015 resulted in a stampede, due to which 36 people were killed. Last month (Oct 31, 2022), at least 151 people were killed during the Halloween crowd stampede in South Korea. In order to avoid such situations, it is imperative to predict the traffic flow for a specific area beforehand, such that various safety measures can be taken to avoid unintended consequences.

Traffic flow consists of traffic inflow and outflow. Inflow is the traffic entering a region from other regions in a particular time interval. Outflow is the total traffic leaving the given region to other regions at a particular time interval. To understand the overall traffic flow, we need to predict both the inflow and outflow of traffic.

In order to use deep learning models for predicting the traffic flow, we need to consider three major factors:

1. Spatial dependencies:
    a. *Nearby*: The inflow and outflow of nearby regions affect each other. For example, consider two neighboring regions, R1 and R2; the outflow of R1 affects the inflow of R2 and vice-versa. A pictorial representation is shown in Figure 1.

   b. *Distant*: The inflow and outflow of a region can also be affected by distant regions. Multiple vehicles might go towards the expressway, causing traffic at the city intersection.
2. Temporal dependencies:
   a. *Closeness*: The current traffic flow within the last few hours significantly influences the current traffic flow at a region.
   b. *Period*: The traffic flow may be similar on consecutive weekdays at the same time; for example, traffic flow on Tuesday at 9 am may be similar to traffic flow on Wednesday at 9 am.
   c. *Trend*: Trends occur in traffic flow in longer time intervals; for example, the traffic congestion in the morning hours may gradually come later as winter comes because people tend to wake up later in the morning as the weather gets colder.
3. External influences: Other factors, such as weather conditions, may influence traffic flow.

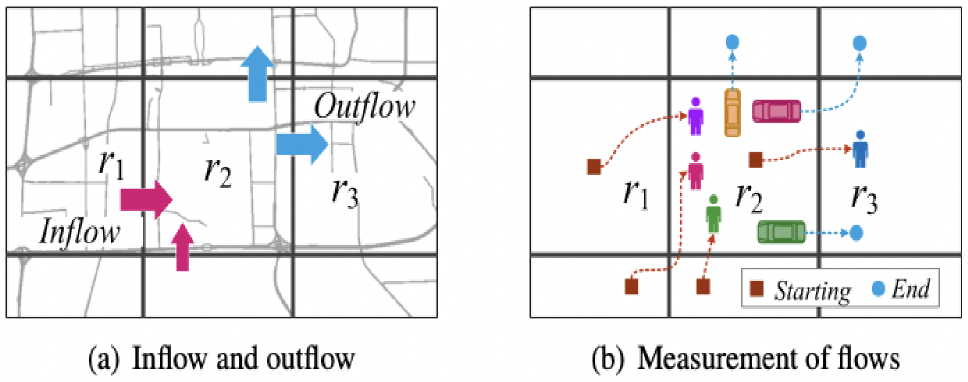

Fig 1: Nearby spatial dependency in traffic flow

Let X(i,t) denote the flow at time t and location i. Given the flow data for i=1,2,3…n and t=1,2,3…m, the problem is to predict the flow X(j,t+Δ) for the location j over a time horizon Δ. Traffic flow is known to have a stochastic and nonlinear nature making it a challenge for accurate prediction.

Methodology

Our methodology divides the city into grids based on longitude and latitude. Each grid represents a region within the city where flow data is recorded. Spatial and temporal features can be used to train a deep learning model, ST-ResNet [1], that collectively predicts the inflow and outflow of traffic in every city region. We implement deep convolutional residual networks, which take the temporal closeness, period, and trend properties as the input to predict traffic flow for a particular timestamp. We implemented a separate residual stack for each property mentioned above (closeness, period, and trend); each unit captures the traffic flow's spatial properties. The model then aggregates the residual networks' output by assigning appropriate weights to each output. This aggregated output is then combined with external factors, such as weather or events happening in the city, to predict the traffic flow for every region.

In order to model the entire citywide dependencies, up to 15 layers of deep convolutional units are required. Generally, such deep neural networks tend to have the problem of vanishing gradients. This can be tackled using residual learning, which has been demonstrated to be very effective for training super-deep neural networks. In our model, we have stacked residual units. A unit comprises two convolutional layers, each preceded by a ReLU activation. Within each residual unit, batch normalization is applied to the data before feeding it to the convolution layer. BN has shown significant improvements in terms of training time reduction. These measures ensure that the model's prediction accuracy is not compromised by the deep structure of the neural network.

Architecture

Firstly, we prepare the inflow and outflow throughout the city as 2-channel grid maps. Temporal data are sampled as recent time(closeness), near history(period), and distant history(trend). Sampled matrices are passed through their respective sub-networks to model the temporal dependencies. The three sub-networks share the same architecture, i.e., convolutional layers with residual connections. A fully connected layer embeds external features such as weather, temperature, wind speed, day of the week, holidays, etc. Finally, all the components' outputs are merged to predict the traffic flow. The architecture of our model is shown in Figure 2.

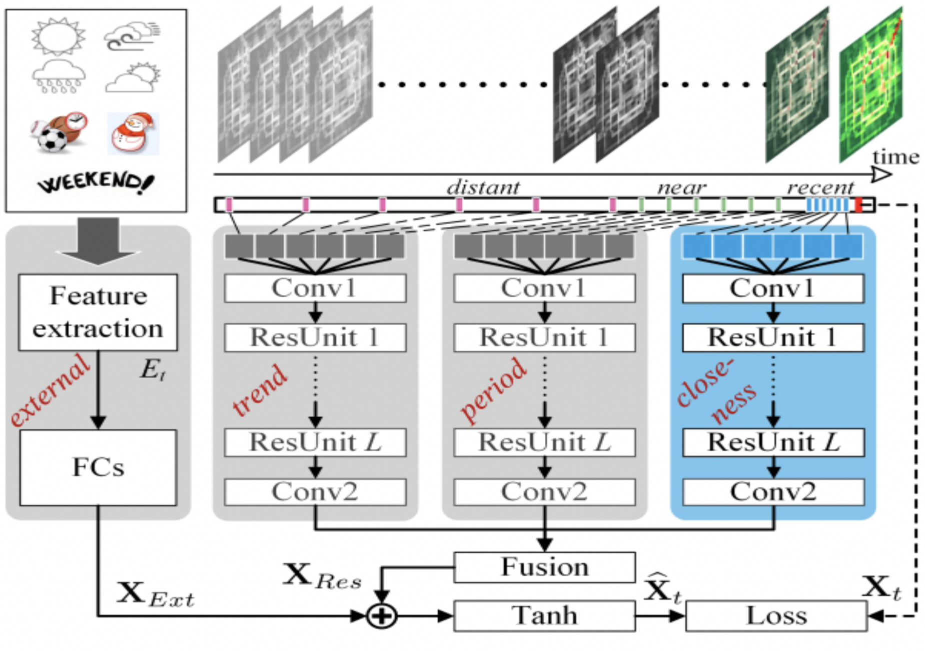

Fig 2. ST-ResNet Architecture

Structure of the sub-networks

Regions of a city have variations in population densities; therefore, the flow at different regions affects each other. Such data can be effectively handled using convolutional neural networks, which hierarchically capture spatial structure information. CNN captures the flow dynamics essential for predicting the flow ahead of time. Different kernel sizes represent the neighborhood radius considered for learning the flow dynamics. In a bigger metropolis, numerous motorways and subway systems typically connect many far areas. Thus, flow in a region can be influenced by distant places. Therefore, one can do subsampling and preserve the distant dependencies to capture this. In our implementation, we do not subsample the data; instead, we train multiple convolutional neural networks to capture the distant spatial dependencies. For kernel size 3, a node in the middle-level feature map depends on 3x3 nodes, and those nodes, in turn, depend on every node in the lower-level feature map (shown in Figure 3). It implies that a single convolution naturally captures spatially close connections and that a stack of convolutions can detect even farther-reaching, citywide dependencies.

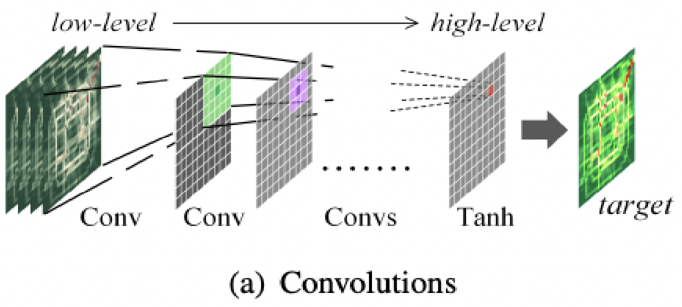

Fig 3. Convolution of flow data for capturing spatial dependencies

Our model employs the same convolution (shown in Figure 4), i.e., it does not use any pooling layers for feature extraction. So, the input and output dimensions of the grid maps remain unchanged throughout the model pipeline. Narrow convolution is helpful for classification and representation learning, but traffic flow prediction is a regression task. The same convolution avoids data loss which is essential for minimizing the MSE loss during the training process.

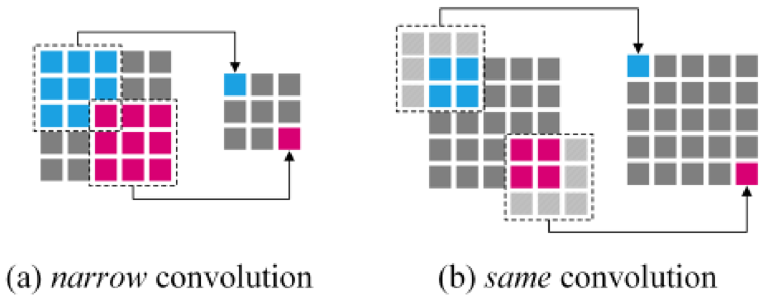

Fig 4. Convolution types

Residual Units

In our model, we stack L residual units as shown in the below equation. Here, Xc represents the flow grid map passing through the residual stack.

$$\mathbf{X}_c^{(l+1)} = \mathbf{X}_c^{(l)} + \mathcal{F}(\mathbf{X}_c^{(l)}; \theta_c^{(l)}), l = 1, \cdots, L$$

Here, F is the residual function (combination of ReLU + Convolution), $\theta^{(l)}$ includes all the learnable parameters in the l^{th} residual unit. We have attempted batch normalization, which is added before the ReLU. The architecture of a single residual unit is shown in Figure 4.

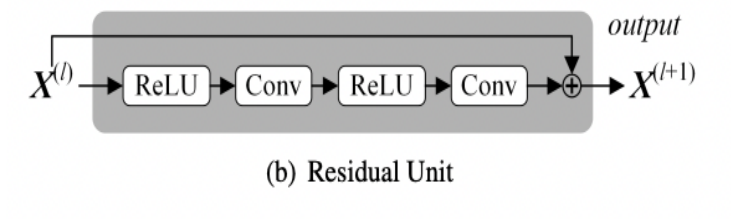

Fig 5. A single residual unit

Structure of External Component

Traffic flows can be affected by external factors such as weather and local events, and it even depends on the day of the week. In our implementation, we have mainly considered weather, holiday events, and metadata like DayOfWeek, Weekday/Weekend. To predict flows at time *t*, the weather and the metadata can be directly obtained. We stack two fully-connected layers to model the external components,

- First layer is an embedding layer for each subfactor, followed by ReLU activation
- Second layer is for upscaling to a high dimension in order to have the exact dimensions as the original grid map.

Fusion

The outputs of the three sub-networks are fused using a weighted sum learned during the training process. This fused output is directly added to the external output, and the result is passed through a TanH activation to generate the predicted flow. TanH function yields a faster convergence when compared with the standard logistic functions in back-propagation learning. The fusion equations are shown below:

$$\mathbf{X}_{Res} = \mathbf{W}_c \circ \mathbf{X}_c^{(L+2)} + \mathbf{W}_p \circ \mathbf{X}_p^{(L+2)} + \mathbf{W}_q \circ \mathbf{X}_q^{(L+2)}$$

$$\widehat{\mathbf{X}}_t = \tanh(\mathbf{X}_{Res} + \mathbf{X}_{Ext})$$

Important Implementation Changes

We added an additional ReLU activation to the end of the resnet stack, which reduced the overall training time. Figure 6 shows a pictorial representation of our model changes.

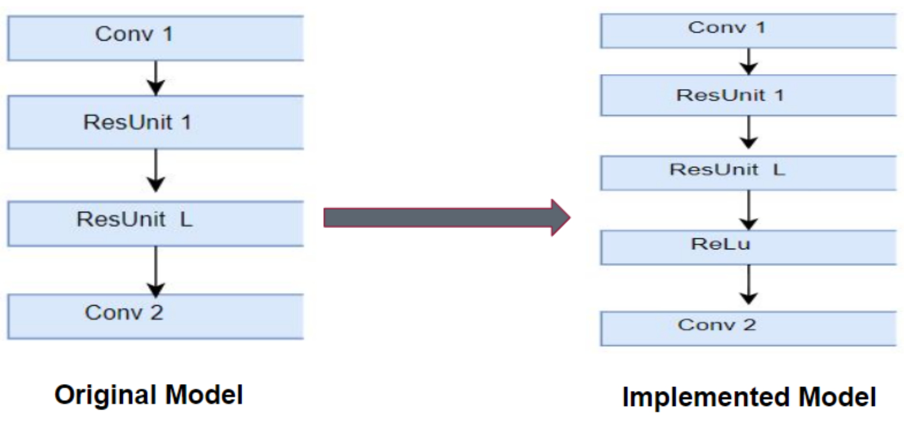

Fig 6. Changes in the resnet stack

Dataset Description

We have primarily focused on experimenting with two popular traffic flow datasets:

1. TaxiBJ: Trajectory data from taxicab GPS data and meteorological data in Beijing.
2. BikeNYC: It consists of Trajectory data from the NYC bike rental system.

Dataset	**TaxiBJ**	**BikeNYC**
DataType	GPS	GPS
Location	Beijing	New York
Time Periods	4 time periods ● 7/1/2013-10/30/2013 ● 3/1/2013-6/30/2014 ● 3/1/2015-6/30/2015 ● 11/1/2015-4/1/2016	1 time period ● 4/1/2014-9/30/2014
Time Interval	30 minutes	1hr
Grid Map Size	32x32	21x12

Model Training

We train our model, with multiple iterations of hyperparameter tuning and after many experiments, we found the following hyperparameters generate best results on the traffic flow prediction task:

- N_closeness = 4 #Samples to considered for closeness (30-mins time gaps)
- N_period = 1 #Samples considered for period (1-day time gap)
- N_trend = 1 #Samples considered for trend (1-week time gap)
- Batch_size = 32
- Num_of_epochs = 50
- Test_train_split = 9:1
- Adam_initial_learning_rate = 0.0001

Flow data, Temperature and Wind speed are normalized in the range of (0 to 1) during the training process. During training, we observed that batch normalization helps in attaining quick convergence. Note that the loss presented here is before denormalization. The loss function used for training was Mean Squared Error (MSE). Figure 7 shows the loss graph during the training process.

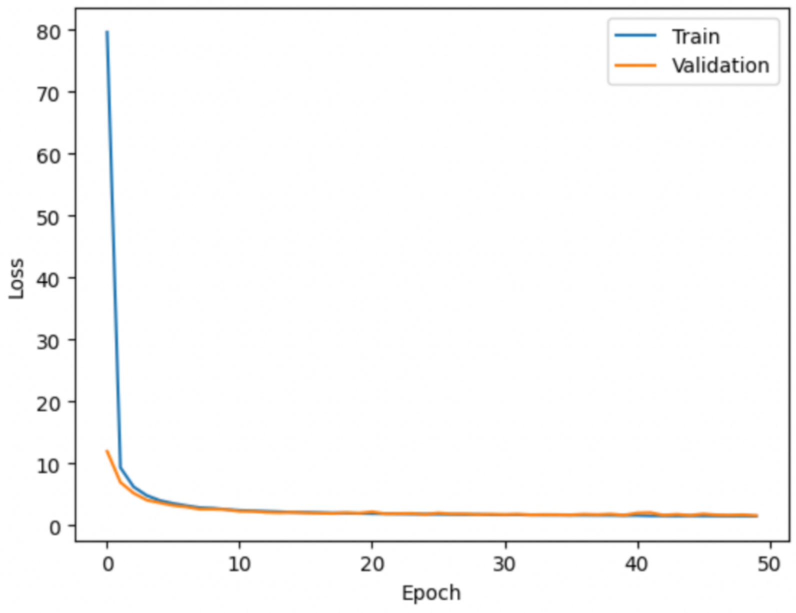

Fig 7. Loss graph during training

From Figure 7, we can see that as the epoch increases, there is a decrease in the loss for both train and validation data, and after around five epochs, both the losses are very similar, this indicates that our model is neither overfitting nor underfitting.

Results

Our results compared to the original paper are shown in the below table:

	Paper Implementation (RMSE)	Our Implementation (RMSE)
BikeNYC	6.33	9.517
Taxi BJ	16.89	17.904

In the case of BikeNYC dataset, the difference can be attributed to the fact that our implementation does not include an external unit. In the case of the Taxi BJ dataset, the RMSE loss is almost similar. The slight difference is that we have yet to consider holidays in our implementation. The loss presented here are after denormalization.

Analysis

The following are our observations with the trained model:

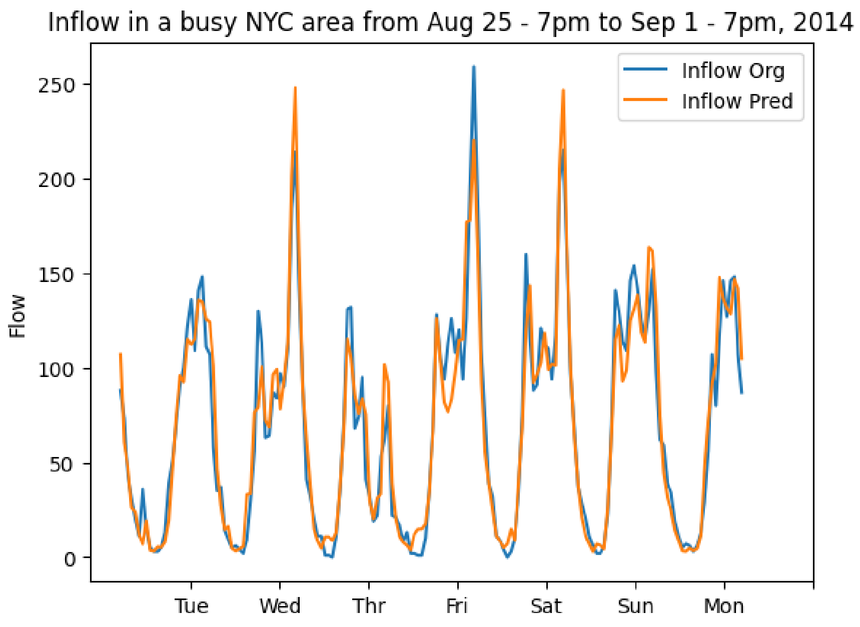

Fig 8. Prediction results with BikeNYC dataset

Figure 8 show our model's prediction (orange) for a typical week at an office area in NYC with the actual inflow (blue). Our predictions are close to the original inflow, and we successfully predicted the spikes and dips in traffic.

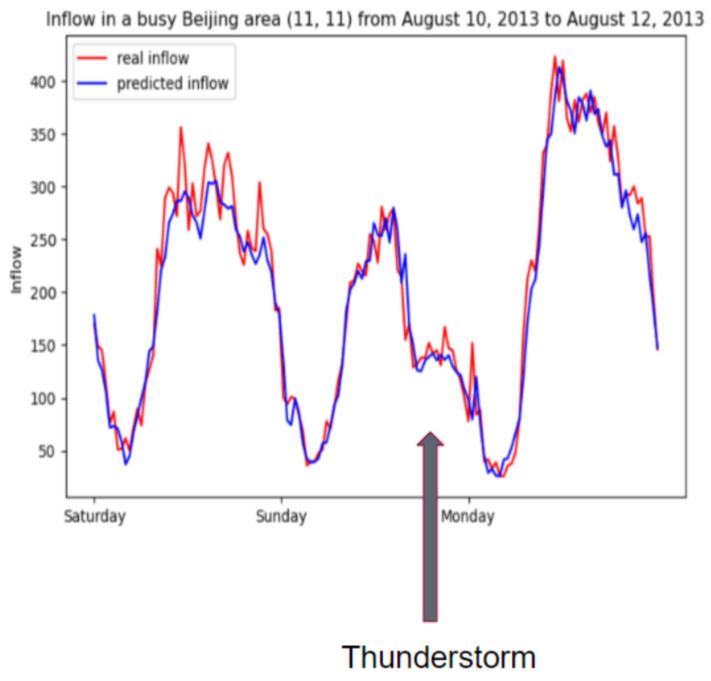

Fig 9. Prediction results with TaxiBJ dataset under external influences

Figure 9 shows the comparison between the actual inflow and the predicted inflow in an active region in the Beijing area. Moreover, the implemented model could predict the flow during a thunderstorm, evidenced by the steep drop in the flow during that time. The prediction aligns very well with the actual inflow, illustrating the model's predictive power.

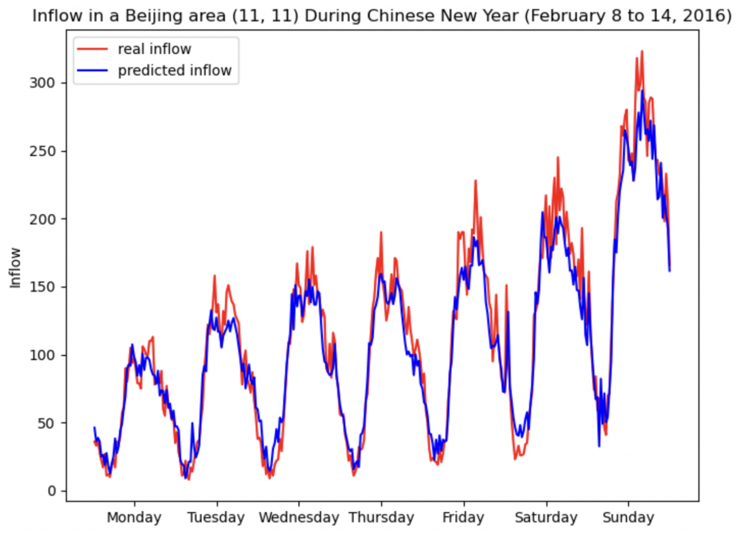

Fig 10. Prediction during a holiday week in Beijing

Figure 10 compares the actual and predicted inflow in the Beijing area during a holiday week. Our model predictions (blue) are in the above figure for a week during the Chinese New Year.

Predictions are accurate for the inflow despite the stark difference in the flow pattern compared to regular days.

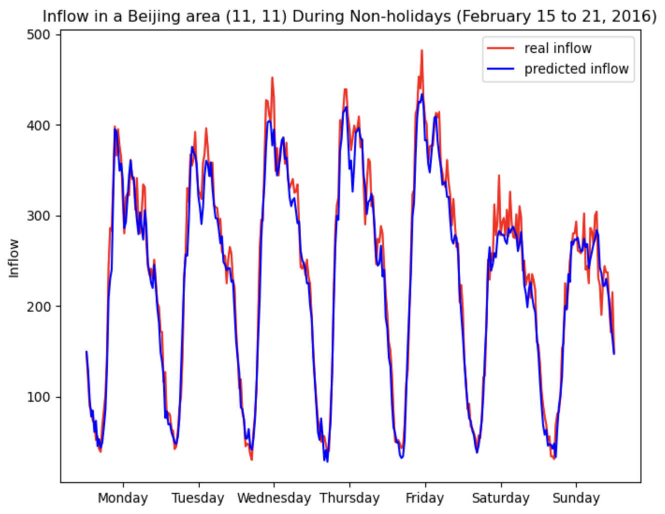

Fig 11. Predictions during a regular week in Beijing

Figure 11 compares the actual and predicted inflow in the Beijing area during a typical week. Even for the week after Chinese New Year (non-holidays/regular days), our model's predictions are accurate for the actual inflow. In the graph, the scale of the inflow during Chinese New Year (Figure 10) is ranged from 0 to 300, whereas during regular days (Figure 11), the inflow scales from 0 to 500, indicating that more people stay indoors during the holiday weeks.

T-step Look Ahead Predictions

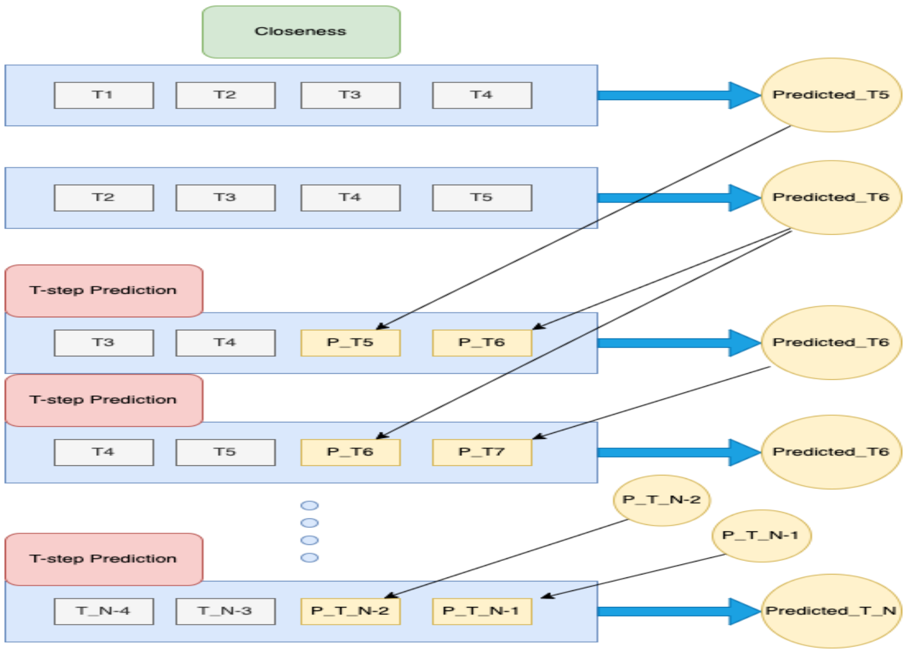

Fig 12. T-step Lookahead Prediction logic

We implemented lookahead prediction by augmenting the closeness component input to the network. The previous T inflow predictions are utilized to predict the next inflow value. A smaller value of T has better prediction power. A larger value of T is more prone to erroneous predictions.

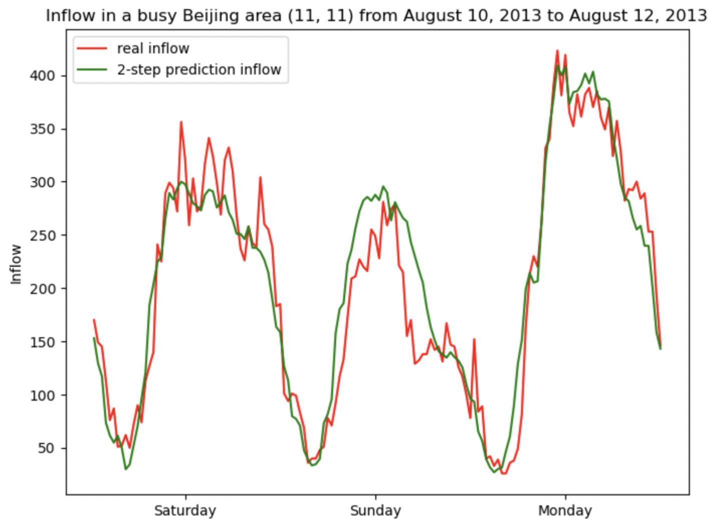

Fig 13. 2-step lookahead predictions

Figure 12 compares the actual inflow with the 2-step lookahead predicted inflow. The predicted inflow is not captured as well as in the previous case (T=1, Figure 9). However, this prediction is still meaningful as it captures the general trend of the traffic flow.

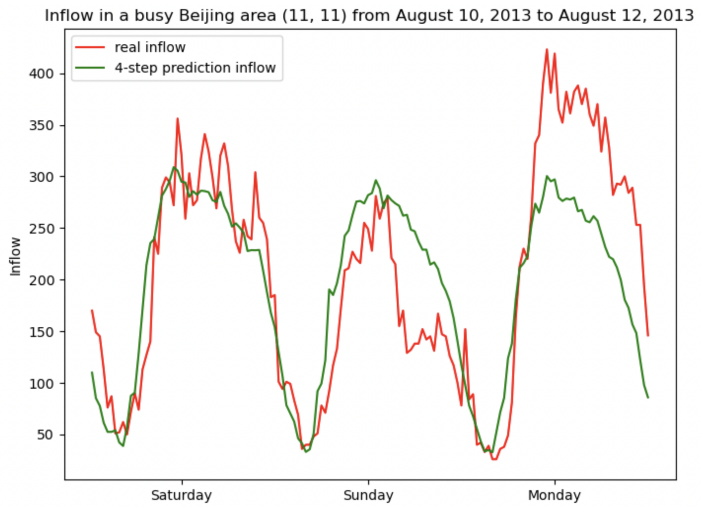

Fig 14. 4-step lookahead predictions

Figure 14 compares the actual inflow with the 4-step lookahead predicted inflow. Increasing the lookahead steps will increase the error or introduce more data loss. This is expected because each prediction relies on more previous predictions as T increases.

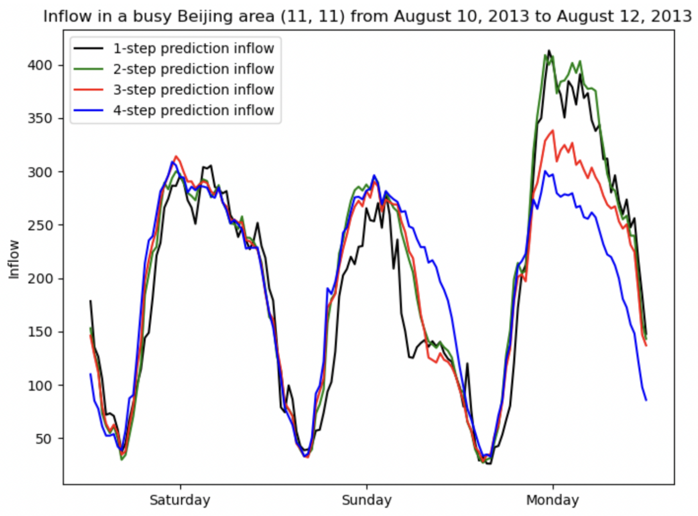

Fig 15. Superimposition of lookahead predictions

Figure 15 superimposes lookahead predictions from T = 1 to T = 4. It can be observed that the fit worsens from 1 to 4. Our current efforts are aligned on a model to improve these lookahead prediction accuracies.

Conclusion and Future Work

We implemented the original ST-ResNet model with few customizations and achieved very accurate results. Two traffic flow datasets were analyzed comprehensively, and the results were presented. Flaws with the lookahead predictions were highlighted, and efforts were made to understand how to improve the results. We plan to improve the T-Step look ahead using RNNs on top of ST-ResNet. Since the model is an ensemble neural network, its intricate architecture can always be modified to improve the results.

References

[1] Zhang, Junbo, Yu Zheng, and Dekang Qi. "Deep spatio-temporal residual networks for citywide crowd flows prediction." Thirty-first AAAI conference on artificial intelligence. 2017.
[2] Citi Bike, N. Y. C. "Citi Bike system data." (2020).
[3] Lv, Yisheng, et al. "Traffic flow prediction with big data: a deep learning approach." IEEE Transactions on Intelligent Transportation Systems 16.2 (2014): 865-873.
[4] Liang, Yuxuan, et al. "Urbanfm: Inferring fine-grained urban flows." Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.
[5] Lin, Ziqian, et al. "Deepstn +: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.
[6] Saxena, Divya, and Jiannong Cao. "D-GAN: Deep generative adversarial nets for spatio-temporal prediction." arXiv preprint arXiv:1907.08556 (2019).
[7] Zhang, Junbo, et al. "DNN-based prediction model for spatio-temporal data." Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems. 2016.