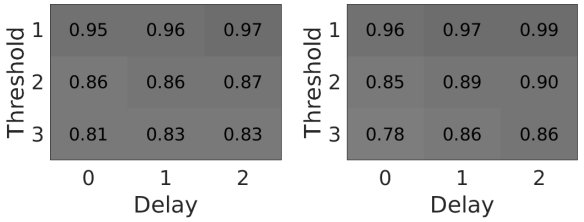
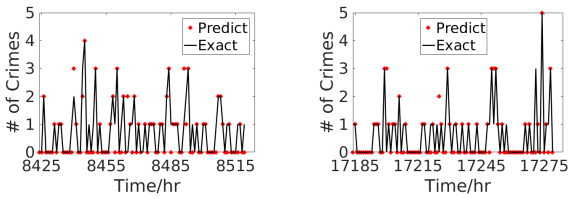
**Table 4: Average RMSE on for CHI Crime Data. Left: RMSE on CDF, Right: RMSE on PDF. Unit: number of crimes.**

| | Single Node | Joint Training | GSRNN |
|---|---|---|---|
| Group 1 | 0.174/0.166 | 0.204/0.143 | 0.102/0.108 |
| Group 2 | 0.381/0.348 | 0.153/0.133 | 0.181/0.197 |
| Group 3 | 0.699/0.697 | 0.413/0.495 | 0.382/0.412 |
| Average | 0.482/0.471 | 0.286/0.317 | 0.258/0.278 |




(a) Los Angeles (b) Chicago

**Figure 14: Precision matrix of GSRNN for Los Angeles and Chicago averaged across top all zipcodes with at least one hourly time slot containing more than 3 crimes.**




(a)CHI zip code 60171 (b)LA zip code 90003

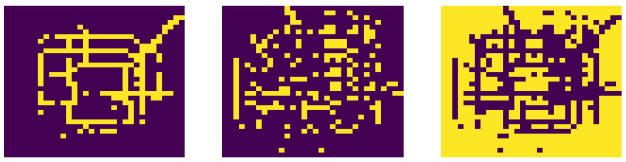
**Figure 15: Panels a), b) plot snapshots of the predicted vs exact hourly crime rate for CHI and LA data.**

graph leads to a further performance increase. Theses conclusions are consistent across the stratified groups as well. The precision matrices (thresholded to three in both number of crimes and delay) averaged over all the nodes for LA and CHI are plotted in Fig. 14, respectively. Figure 15 shows the predicted and exact crime time series over two graph nodes.

## 5 ST TRAFFIC FORECASTING RESULTS

We benchmark our methodology on two public datasets for traffic forecasting [19]. The BikeNYC and TaxiBJ datasets are both embedded in rectangular bounding boxes, forming a rectangular grid of size 32 x 32 and 16 x 8 respectively.

To cast the problem into the graph representation framework used in this paper, we consider each pixel in the spatial grid as a graph node, and connect each node with its four immediate neighbors. The graph weights are set to 1/4 for all edges, the same as in an unweighted graph. More sophisticated methods could be used for graph construction, but we found the 4-regular graph already yields good performance. The nodes are then sorted into three



(a) (b) (c)

**Figure 16: Panels a)-c) visualize the node class assignment from group 1 - 3 in the Beijing Traffic data respectively, where a yellow pixel indicates the assignment of the pixel node to its corresponding class. For example, the yellow pixels in panel a) are grouped to class 1.**

classes according to the overall cumulative traffic count. For the New York data, there are three equal size classes, whereas in the Beijing dataset, the classification is picked manually to reflect the geographical structure of the Beijing road system (see Fig. 16).

For the BikeNYC, we use a two layer LSTM with (32, 64) units and 0.2 dropout rate at each layer for the single-node model, and a two layer LSTM with (64, 128) units and 0.2 dropout rate at each layer for the joint and GSRNN model. For the TaxiBJ, we use a two layer LSTM with (64,128) units for the single-node, and a three layer LSTM model with (64, 128, 64) units for the joint model; for the GSRNN model, the edge RNN is a two layer LSTM with (64, 128) units, and the node RNN is a one layer LSTM with 128 units. All models are trained using the ADAM optimizer with a learning rate of 0.001 and other default parameters. The learning rate is halved every 50 epochs, and a total of 500 epochs is used for training.

For evaluation, we use the Root Mean Square Error (RMSE) across all nodes and all time intervals. The same train-test split is used in our experiments as in [19]. The results on RMSE (Table 5) are reported on the testing error based on the model parameters with the best validation loss. Comparisons between the predicted and exact traffic on two grids over a randomly selected time period is shown in Fig. 17. On a randomly selected time slot, we plot the predicted and exact spatial data and errors in Figs. 18 and 19.

**Table 5: RMSE on for Traffic Data.**

| | Single Node | Joint Training | GSRNN | STResNet [19] |
|---|---|---|---|---|
| Beijing | 23.50 | 19.5 | **16.59** | 16.69 |
| NY | 6.77 | 6.33 | **6.08** | 6.33 |

## 6 CONCLUSION

We develop a multiscale framework that contains two components: inference of the macroscale spatial temporal graph representation of the data, and a generalizeable graph-structured recurrent neural network (GSRNN) to approximate the time series on the STWG. Our GSRNN is arranged like a feed forward multilayer perceptron with each node and each edge associated with LSTM cascades instead of weights and activation functions. To reduce the model's complexity, we apply weight sharing among certain type of edges