Predicting Traffic Congestion on City Road Networks

Andrew Deng
andrewde@stanford.edu

Jennie Chen
jenniechen@stanford.edu

Wenli Looi
wlooi@stanford.edu

1 Introduction

Traffic congestion in metropolitan areas is a problem that plagues almost every highly populated city worldwide, leading to heavy delays in travelling short distances and disruptions in productivity for people working in such cities. An effective model for predicting when traffic will occur and at what intensity would be a great asset in both understanding how traffic evolves and designing traffic resistant road networks.

Prior works have focused on modelling traffic flow throughout the day as a spatio-temporal problem, relying on data about traffic throughout each day to predict how traffic will evolve. In this project, we intend to develop a system to predict traffic congestion based solely on the road network, relying less on the temporal aspect of traffic and focusing instead on the topology and attributes. Given the road network and limited information about each road segment, we will predict how congested each road segment will be. With this approach we hope our model will be able to generalize well to new street networks that may have differing temporal characteristics.

2 Related Work

2.1 Analyzing Traffic Congestion

Wen et al. use both the street network topology as well as partial data about real traffic volumes to predict which segments in the street network of a city are likely to encounter heavy traffic congestion [1]. The authors develop a model that accounts for human movement preferences through an 'attractiveness' rating for each street segment and use a dual-graph representation of the street network to account for network structure, with street segments represented as vertices and intersections represented as edges. A PageRank-like algorithm was used to determine the congestion level of each street segment while considering the attractiveness ratings.

2.2 Using Community-Based Approaches

Duan and Lu approach the problem of understanding the robustness of city road networks through a community-focused lens, following the theory that incidents on specific roads will impact roads that are topologically nearby [2]. The authors use a label propagation-based detection algorithm to find communities in the dual graph, creating a second graph with communities as nodes and community connections as edges.

Duan and Lu find that the community graph shows more node degree diversity than the dual graph, allowing for a more nuanced investigation into the importance of different nodes. The authors also find that network fragmentation is most vulnerable to betweenness-based attacks where connections are severed based on betweenness centrality. Duan and Lu conclude that city road networks are likely highly clustered, making community-based approaches valuable when considering city road robustness.

2.3 Relational Fusion Networks

Relational Fusion Networks (RFNs) [3] are a very recent (2019) alternative to Graph Convolutional Networks (GCNs) that are specifically designed for road networks. Compared to previous GCNs, a key difference of RFNs is that they learn embeddings for vertices, edges, and "between-edges", which are edges between vertices of the dual graph. The authors claim that RFNs perform better on road networks because these networks contain important information in edges and between-edges.

2.4 Influence On Our Work

Wen et al. produce a model providing traffic congestion predictions which is developed with very little actual data about streets and street traffic beyond just the network structure. This shows that prediction of congestion based mostly on graph structure and some minimal traffic data is at least somewhat feasible.

In this project, we expanded the work done in the paper into a machine learning model trained on an existing street network with full data. This model would then be able to make predictions by using the data to learn an association between the topological structure and traffic. We incorporate Duan and Lu's findings about the efficacy of community-based approaches and extend their approach to community detection by including other attributes such as a road's length or type in addition to purely structural features.

We did not implement an RFN for our project; however, Jepsen et al. provide insight into how attributes from road networks, like road types, can be effectively used with a graph convolutional neural network. Based on this, we obtain attributes from OpenStreetMap to aid in training our model.

3 Dataset

We use speed data provided for various cities through Uber Movement [4], combined with map data from OpenStreetMap [5].

For each road segment, the Uber Movement data provides mean speed, standard deviation of speed, and 85th percentile speed. Entries are provided for various hours of the day, aggregated across all days in the second quarter of 2019. Segments are additionally linked to the corresponding segments in OpenStreetMap, allowing us to incorporate additional data about each road segment. From the OpenStreetMap data, we will be using the road type (there are 13 road types, e.g. primary, secondary, residential) and road length.

Since the OpenStreetMap data is contributed by users, the data is incomplete and may be inaccurate.

3.1 Preprocessing

For our project, we preprocessed the Uber Movement data to compute a single traffic value for each road segment. We ignore all of the Uber Movement data except the entries from 7AM - 10AM; by focusing on these hours, we will develop a model that predicts likely congestion during morning rush hours. The traffic value of a road segment is defined as the percent slowdown in average speed during this time period compared to free-flow speed, defined by Uber as the 85th percentile of all speed values observed on the segment. (Note that the 85th percentile is a relatively standard metric for the maximum safe speed of a road [6].) This allows us to generalize for potentially different speed limits, which should allow our model to be effective on different road networks in various regions.

We form a graph of the road network from OpenStreetMap using the osmnx library [7]. This graph is initially a directed multigraph, as roads in OpenStreetMap are directed and parallel roads may exist. For simplicity, we convert this to a directed simple graph where all parallel edges are combined into a single edge. The Uber Movement and OpenStreetMap datasets are joined using the OSM way ID, and all edges (road segments) that are not present in both datasets are dropped. We also drop data about roads outside of the city limits. From OSM, we pull in attributes for each road such as road type, road length, and number of lanes; roads without a specified number of lanes are defaulted to one lane.

We also construct the dual graph for each city, where we represent the edges (street segments) as nodes and the nodes (road intersections) as directed edges. More formally, an edge (i, j) in the dual graph means that it is possible to navigate from road i to road j. This is not the same as being able to navigate from road j to road i since roads are directed; however, we find that since most roads are bidirectional, the dual graph usually will have both edges (i, j) and (j, i) for most pairs of road segments i and j.

As an additional step, we run community detection on our original graph using Clauset-Newman-Moore greedy modularity maximization [8]. This algorithm works by starting with a set of isolated nodes (each in its own community) and then iteratively adding back the edges between nodes to merge communities and greedily maximize modularity. Modularity, which measures that there are many edges within communities and few between them, is defined as the following:

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w)$$

where m is the number of edges in the graph, A_{vw} is an element of the adjacency matrix A of the graph (where A_{vw} is 1 if nodes v and w are connected and 0 otherwise), k_v is the degree of a node v, and $\delta(c_v, c_w)$ is 1 if the inputs are equal and 0 otherwise.

We incorporate the community detection into our models by adding a 0/1 edge attribute describing whether or not the road connects two different communities. By incorporating communities this way, our model can work on various graphs that may inherently have different numbers of communities instead of forcing all input graphs to have the same number of communities.

3.2 Summary Statistics

Data is available for various cities; we will be using the data for San Francisco, New York City, and Seattle. We will perform 3-fold cross validation by training on two graphs to develop a model and then applying the model to the third graph to test generalization of the trained model.

Here are some summary statistics about our three graphs:

City	# Nodes (Junctions)	# Edges (Road Segments)
San Francisco	15758	33784
Seattle	23799	48841
New York City	69462	138586

We also have summary statistics on the dual graphs:

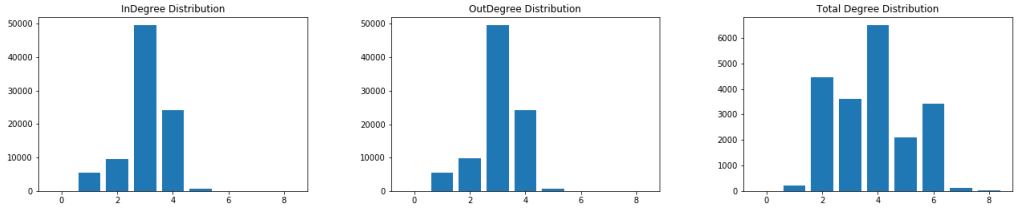
City	# Nodes (Road Segments)	# Edges (Junctions)
San Francisco	33784	86131
Seattle	48841	111659
New York City	138586	322096

As we can see, the number of nodes in the dual graph is the same as the number of road segments in the original graph. We also note that the number of edges has increased significantly. This is because the same junction in the dual graph may be represented by multiple edges. For example, at a 4-way intersection, each road segment can reach every other road segment in the intersection. Then we will have 3 edges representing the single intersection. If all of the roads are bidirectional, we may even have up to 6 edges representing the single intersection, to show that we can go through the intersection in the opposite direction.

4 Initial Analysis

In order to understand our network data before we build our traffic congestion model, we conducted a preliminary analysis of the San Francisco city road network, which we intend to use for training. Analysis on the other city road networks (New York City and Seattle) showed similar trends as with our analysis for San Francisco.

4.1 Node Degree Distribution



Note that since almost all roads are bidirectional, the in-degree and out-degrees of most nodes are the same. We will provide a general analysis over both the in-degree and out-degree distribution.

We can see that the highest in-degree/out-degree for a node is degree 3. As much of the city is in a grid structure, this makes sense; a road segment entering a 4-way intersection often is able to access the three other streets in the intersection.

Considering the total degree (in-degree + out-degree), we found spikes in the number of nodes with degrees 2 and 4 on our dual graph. This is logical considering what this means in the road network; a road segment on a straight road would have degree 2 in the dual graph because it connects to the road segment in front and behind it. Similarly, a node leading to an 4-way intersection (which is common in grid networks) connects to the 3 other roads in the intersection as well as the road segment behind it. A minor spike was also visible at degree 6, which similarly corresponds to a road segment between two intersections of a grid.

4.2 Clustering Coefficient

The clustering coefficient C_i of a node i in a graph is defined as (twice) the number of edges between the neighbors of a node divided by the maximum possible number of edges between the neighbors. Conceptually it represents how well connected that node's neighbors are with each other.

$$C_i = \frac{2e_i}{k_i(k_i - 1)}$$

The average clustering coefficient of the San Francisco graph was 0.010028. Such a low clustering coefficient is to be expected given that in a city road network, only rarely will an intersection's neighboring junctions be directly connected to each other. Usually there will be another junction between any two neighbors of an intersection. After transforming each graph to its dual graph, we observe that the average clustering coefficient is now 0.3385. In the dual graph we see larger clustering coefficients because at most intersections, each street is able to reach all other streets; therefore, in the dual graph this becomes a clique where all road nodes are connected to each other.

4.3 Betweenness Centrality

The betweenness centrality of a node is the number of shortest paths in the graph that pass through that node. Considering the context of traffic congestion, it makes sense that a road with high betweenness centrality should be important in our road network and therefore be more likely to have congested traffic.

Figure 1 below shows betweenness centrality computed for edges (purple is low, green is high) in the original San Francisco graph. Edge centrality could be a useful indicator of traffic demands in the graph; for example, we see that the highways and some stretches of the road in the main city have relatively high centrality measures, which indicates that they are likely used more often than others.

Figure 1: Edge betweenness centrality for San Francisco

Comparing betweenness centrality to actual traffic speeds, we can see that although betweenness can identify some larger roads with slow traffic, overall it is not sufficient to accurately predict traffic congestion.

4.4 PageRank

The PageRank algorithm is an algorithm designed to compute and rank the importance of nodes within a directed graph. The importance is computed using the graph structure, and provides a relative measure of how much more likely that node is to be visited than other nodes in the graph. In the algorithm, importance is initialized uniformly, and iteratively distributed across the graph by the edges.

In our dataset, we compute the PageRanks of each node in the dual graph, representing edges in the original graph. This importance should give us another feature representing the likelihood that a road segment will receive high traffic flow; we would expect nodes with a higher PageRank score to be more congested.

Upon computing the PageRanks for our dataset, we find that it is not immediately obvious if the PageRank represents traffic flow well. The distribution of PageRank values on the city road network does not appear to match our empirical traffic data very well at all. See figure 2 for the PageRank of San Francisco.

Figure 2: PageRank on the dual graph

Figure 3: Communities in SF

4.5 Communities

As described earlier, we ran Clauset-Newman-Moore community detection on our graph as a preprocessing step. In Figure 3, we can see the communities found in San Francisco. Some of the communities seem to correspond well to San Francisco neighborhoods; for example, the light blue in the middle left seems to correspond to Golden Gate Park, while the green in the middle right roughly corresponds to Mission Bay. However, this is not true of most of the communities found, which in some cases subdivide larger neighborhoods and in other cases may combine several smaller neighborhoods.

Below, we include a table of the number of communities found in each city, as well as the percentage of edges that connect different communities:

City	# Communities	% Edges Crossing Communities
San Francisco	61	3.47%
Seattle	128	2.84%
New York City	159	2.16%

5 Evaluation

Our model performs 4-class classification on each road of our graph, with the classes being the following:

- 0-20% below free-flow speed
- 20-40% below free-flow speed
- 40-60% below free-flow speed
- 60-80% below free-flow speed

Ties go to the slower class - for example, 20% below free-flow speed counts as 20-40% below. For our three cities, we have no rows that are more than 80% below free-flow speed. These classes match the ones that Uber Movement uses, allowing us to easily compare to ground truth.

We perform qualitative evaluation on our model by plotting the actual and predicted traffic for each road on a map and checking to see if it looks similar and reasonable.

Our classes are relatively imbalanced; for example, around half of the roads are in the first class (0 - 20% below free-flow speed). Therefore, overall accuracy is not a very useful metric. Instead, we compute per-class accuracy for each of the four classes and also average the four accuracies as another metric.

The actual traffic of each city is included below:

(a) San Francisco

(b) Seattle

(c) New York

Figure 4: Actual traffic class distributions for each city

6 Baselines

As baselines, we explored using simple classification models, such as logistic regression and a 1-hidden-layer neural network, to predict each street's traffic class from the attributes. As attributes, we included the number of lanes, the road length, the road type (as a one-hot-vector), whether or not the road connects two communities, and the dual graph node degree. At first we considered only each particular street's attributes in the classification, but we moved to using a slightly stronger network that considered the egonet of each street in the classification.

Each of our baselines was trained using an Adam optimizer with learning rate 0.01 for 3000 epochs. Since we have imbalanced classes, our training loss was negative log-likelihood where the weight for each sample was inversely proportional to the class size.

6.1 0-hop baselines

We had two models that classified the traffic level of a particular street just given that street's attributes. The first was a simple softmax network:

$$\hat{y}_i = S(Wx_i + b)$$

The second was a 1 hidden layer neural network with 512 hidden units and ReLU activations:

$$\hat{y}_i = S(W_2 \max(0, W_1 x_i + b_1) + b_2) = S(N(x))$$

where $N(x)$ represents the 1 layer network and $S(x)$ is the softmax function. In the 1 hidden layer model, we also included a dropout layer [9] with probability 0.5 after the hidden layer to reduce overfitting.

6.2 1-hop baseline

For our 1-hop network we take the means across all neighbor attributes and concatenate the mean vector with the street's attributes and pass it through a similar 1 layer neural network:

$$\hat{y}_i = S\left(N\left(x_i \parallel \left(\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} x_j\right)\right)\right)$$

where $\mathcal{N}(i)$ represents the neighbors of node i.

7 Method

7.1 GraphSAGE

Since we are predicting a traffic level for each node and we would like our model to consider the graph structure and the attributes of neighbors in its classification, we choose to use GraphSAGE as our classification model. GraphSAGE is a graph convolutional model for learning node embeddings in a graph. These embeddings can then be used for classification using any classification network. The embeddings are computed for each layer k as

$$h_v^k = \sigma\left(W_k \text{AGG}(h_u^{k-1}, \forall u \in \mathcal{N}(v)) \parallel B_k h_v^{k-1}\right)$$

where σ is a nonlinearity and AGG is some neighbor aggregation method that takes in all of the attributes of neighboring nodes and aggregates them into a single attribute vector. (Our specific choices are given below.) Both the aggregations and the node's own embedding are transformed by learned linear transformations W_k and B_k before concatenation as well.

7.2 Model

Following our preprocessing, we have a dual graph of roads as nodes and junctions as edges; for each node, we have node attributes corresponding to road type (as a one-hot-vector), road length, number of lanes, and whether or not the road connects different communities. We also explicitly include the degree of each node, although our models would probably be able to deduce this anyways.

This dual graph is fed into a GraphSAGE model, where we have chosen the ReLU function as our nonlinearity and the max function as our aggregation function. The GraphSAGE model included a dropout layer with probability 0.5 after each convolutional layer and has 128 hidden units. We tried variants of the model with 1, 2, 4, and 8 layers; each model was trained using an Adam Optimizer with a learning rate of 0.01 for 3000 epochs. Once again, since we have imbalanced classes, our training loss was negative log-likelihood where the weight for each sample was inversely proportional to the class size.

8 Results

8.1 Baseline Performance

The class accuracies for each of our baselines, for each possible cross-validation combination, are listed in Table 1.

Unsurprisingly, the 0-hop 1-layer model outperformed the 0-hop 0-layer model, but surprisingly the performance of the 0-hop 1-layer model was not much different from that of the slightly better 1-hop 1-layer model. We noticed that at the 0-hop level, the class accuracies for the 0-20% below free-flow speed class were relatively similar across cities; however, with the 1-hop model, this class accuracy was relatively low when tested on San Francisco, but very high when tested on New York. This difference might be because of differences between the New York graph structure and the San Francisco/Seattle graph structures, which would cause a difference in the 1-hop baseline but not the 0-hop ones. We explore this idea further in our model performance results.

8.2 GraphSAGE Model Performance

Likely due to the small amount of datapoints in the 60-80% below free-flow speed class, all of our models have very low accuracies for this class. This is slightly disappointing, as it means we cannot accurately predict the roads that are likely to have the slowest speeds. However, the 2-layer GraphSAGE model does perform

Model	City Tested On	Model Accuracy				
		0-20%	20-40%	40-60%	60-80%	Average
0-hop 0-layer	San Francisco	0.5214	0.2839	0.1621	0.1538	0.2803
	Seattle	0.6097	0.0973	0.4341	0.0909	0.3080
	New York	0.6337	0.0695	0.1339	0.3091	0.2866
0-hop 1-layer	San Francisco	0.6169	0.2669	0.3887	0.1538	**0.3566**
	Seattle	0.6675	0.1047	0.5220	0.1818	0.3690
	New York	0.7173	0.1208	0.2688	0.1091	0.3040
1-hop 1-layer	San Francisco	0.4840	0.3453	0.4251	0.1000	0.3386
	Seattle	0.6832	0.3288	0.4909	0.0172	**0.3800**
	New York	0.8668	0.1282	0.2063	0.0326	**0.3085**

Table 1: Accuracy of each baseline model for each of the four classes for each city

# Layers	City Tested On	Model Accuracy				
		0-20%	20-40%	40-60%	60-80%	Average
1	San Francisco	0.3584	0.4477	0.4886	0.0500	0.3362
	Seattle	0.6738	0.4326	0.4275	0.0000	0.3835
	New York	0.8546	0.0468	0.3944	0.0109	0.3267
2	San Francisco	0.7486	0.4085	0.4125	0.0000	**0.3924**
	Seattle	0.7416	0.1779	0.6918	0.0000	**0.4028**
	New York	0.8953	0.1218	0.2325	0.0000	**0.3124**
4	San Francisco	0.6223	0.5143	0.3341	0.0000	0.3677
	Seattle	0.7802	0.4154	0.0572	0.2414	0.3736
	New York	0.9169	0.1182	0.1136	0.0435	0.2980
8	San Francisco	0.9499	0.0000	0.0000	0.0000	0.2375
	Seattle	0.9185	0.2008	0.0134	0.0000	0.2832
	New York	0.8458	0.0002	0.0322	0.1957	0.2685

Table 2: Accuracy of each GraphSAGE model for each of the four classes for each city

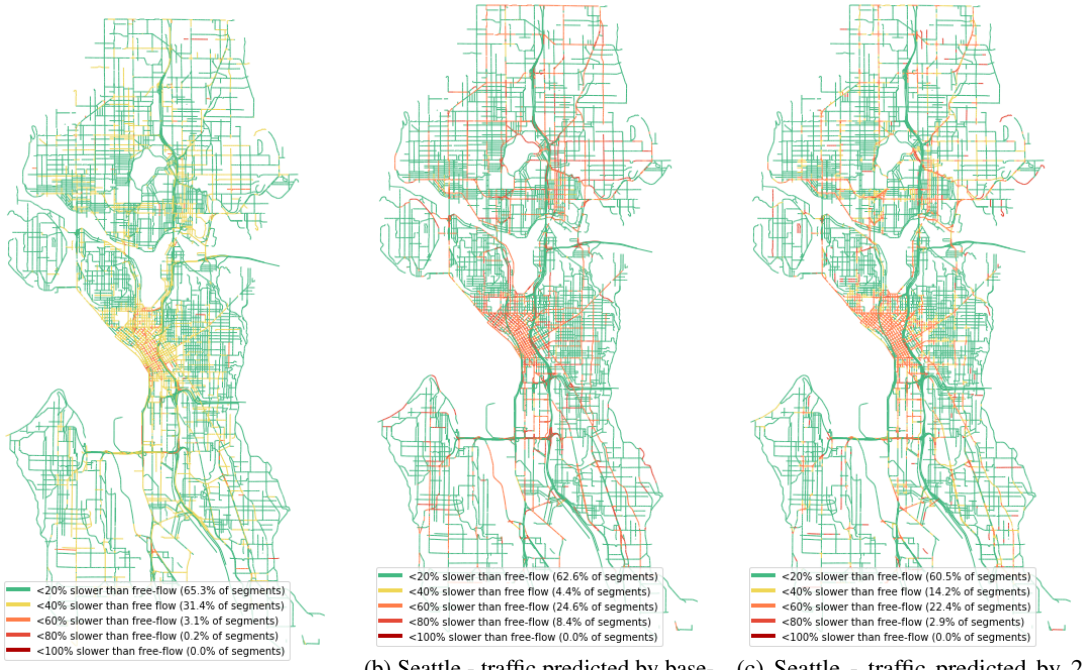
admirably on the class with the next slowest speeds (40-60% below free-flow speed), with 69.18% accuracy when tested on Seattle and 41.25% accuracy when tested on San Francisco.

The 2-layer GraphSAGE model overall performs the best in terms of average class accuracy; on the other hand, we can see that our 8-layer GraphSAGE model is clearly focusing too much on the 0-20% classes, which is closest to free-flow speed. The 8-layer model reaches over 90% accuracy on that class when tested on San Francisco and Seattle, as opposed to the near 0% accuracy in the other classes. This is reflected in the low average class accuracies.

We notice that the performance appears to decrease as the number of layers in our model goes up. This may be because increasing the number of layers increases the neighborhood that the network considers in classifying each street's traffic level, and traffic data is noisy enough that considering attributes beyond a small neighborhood can hurt performance by causing some kind of overfitting to noise. There may also be problems with numerical instability or local optima in larger models.

We also notice that our model performance is always weakest when we train on the San Francisco and Seattle graphs and then test on the New York graph. This may simply be due to the size of the graphs; the New York graph is much larger than the other two, having more nodes and edges than the San Francisco and Seattle graphs combined. The poor performance may be due simply to not having enough training data. In addition, the traffic distribution is rather different for New York compared than the other two graphs, with the majority of roads having speeds 20-40% below free-flow speed (as opposed to 0-20% below free-flow speed). This

may make training on SF and Seattle and then testing on New York a poor match; it may be better to train on Seattle and then test on SF.



(a) Seattle - actual traffic

(b) Seattle - traffic predicted by baseline

(c) Seattle - traffic predicted by 2 layer GraphSAGE

Figure 5: Model predictions for Seattle

We qualitatively compare the performance of our 1-hop 1-layer baseline and our 2-layer GraphSAGE model when tested on Seattle. In Figure 5, we can see that both models were able to predict general trends in Seattle speeds, such as slow traffic in downtown Seattle (in the center). The main problem seems to be that our models are a little over-enthusiastic in predicting very slow speeds (as opposed to speeds that are only a little slower than free-flow speeds). The baseline model went slightly overboard in predicting slow speeds, as we can see by the very red roads; the GraphSAGE model did a little better, with more yellow roads overall which more closely matches the actual traffic. We can also see that the GraphSAGE is better at predicting road speeds in other areas. Take the long road in the bottom middle of Seattle; it is green in the actual Seattle traffic. While the baseline predicts that traffic on that road is approximately 40-60% slower than free-flow speed, the GraphSAGE model accurately predicts that traffic there is close to free-flow speed.

9 Conclusion

Our results show that it is possible to predict city road traffic with only limited information about road segments and without any actual traffic information. The models achieve accuracies significantly higher than random chance while predicting traffic that qualitatively seem similar to the real traffic when plotted on a map, even if that is not reflected in quantitative evaluation.

One direction of future work is investigating alternative accuracy measures. The main issue with our current accuracy measure is that it may not actually indicate how "good" the traffic predictions are, e.g. in being useful for a particular task. Also, we treated this problem as a 4-class classification problem, but the traffic classes have an ordering that should probably be taken into account when considering prediction accuracy.

Additionally, we believe that extracting more useful features about the graph would greatly help our models. Some of our current attributes are somewhat related to how people would actually behave on those streets (street type, number of lanes), but more features like this are probably necessary for our model to accurately predict human behavior.

References

[1] Tzai-Hung Wen, Wei-Chien-Benny Chin, and Pei-Chun Lai. Understanding the topological characteristics and flow complexity of urban traffic congestion. *Physica A: Statistical Mechanics and its Applications*, 473:166 – 177, 2017. URL: http://www.sciencedirect.com/science/article/pii/S037843711730033X.

[2] Yingying Duan and Feng Lu. Structural robustness of city road networks based on community. *Computers, Environment and Urban Systems*, 41:75 – 87, 2013. URL: http://www.sciencedirect.com/science/article/pii/S0198971513000227.

[3] Tobias Skovgaard Jepsen, Christian S. Jensen, and Thomas Dyhre Nielsen. Graph convolutional networks for road networks, 2019. arXiv:1908.11567.

[4] Uber movement: Let's find smarter ways forward, together. https://movement.uber.com.

[5] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.

[6] Eric M Ossiander and Peter Cummings. Freeway speed limits and traffic fatalities in washington state. *Accident Analysis & Prevention*, 34(1):13–18, 2002.

[7] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers Environment and Urban Systems*, 65:126–139, 07 2017. doi:10.1016/j.compenvurbsys.2017.05.004.

[8] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), Dec 2004. URL: http://dx.doi.org/10.1103/PhysRevE.70.066111, doi:10.1103/physreve.70.066111.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.