

Graph Convolution Networks for Probabilistic Modeling of Driving Acceleration

Jianyu Su¹, Peter A. Beling², Rui Guo³, and Kyungtae Han⁴

Abstract—The ability to model and predict ego-vehicle’s surrounding traffic is crucial for autonomous pilots and intelligent driver-assistance systems. Acceleration prediction is important as one of the major components of traffic prediction. This paper proposes novel approaches to the acceleration prediction problem. By representing spatial relationships between vehicles with a graph model, we build a generalized acceleration prediction framework. This paper studies the effectiveness of proposed Graph Convolution Networks, which operate on graphs predicting the acceleration distribution for vehicles driving on highways. We further investigate prediction improvement through integrating of Recurrent Neural Networks to disentangle the temporal complexity inherent in the traffic data. Results from simulation with comprehensive performance metrics support that our proposed networks outperform state-of-the-art methods in generating realistic trajectories over a prediction horizon.

I. INTRODUCTION

Autonomous pilots or intelligent driving assistants predict the future state of traffic in order to warn human drivers about collision risks. The autonomous system in the ego-vehicle should consider not only the ego-vehicle’s interactions with its immediate neighbors, but also hierarchical and chains of interactions that might affect the ego-vehicle’s future state.

Many approaches have been proposed to predict the behavior of vehicles, with most methods falling into the broad categories of regression formulations or classification formulations. While formulating the problem of predicting vehicle behaviors as a classification problem makes it easier to train the model and compare its performance, this classification approach fails to provide detailed future traffic information for planning the future trajectory. Regression methods, however, are able to infer the future state of traffic, such as vehicle position, velocity and acceleration. In the literature, many of the methods for the regression formulation of traffic prediction employ Recurrent Neural Networks (RNNs). RNNs are widely used to study time-series data. In particular, researchers have been successfully applying Long-Short Term Memory (LSTM) network to various applications such as speech generation, machine translation, and speech

¹Jianyu Su is a doctoral student in the Department of Engineering Systems and Environment, University of Virginia, 151 Engineer’s Way, Charlottesville, VA, 22904, U.S.A js9wv@virginia.edu

²Peter A. Beling is a professor in the Department of Engineering Systems and Environment, University of Virginia, 151 Engineer’s Way, Charlottesville, VA, 22904, U.S.A pb3a@virginia.edu

³Rui Guo is a principal researcher in Toyota InfoTech Labs, 465 N Bernardo Ave, Mountain View, CA, U.S.A rui.guo@toyota.com

⁴Kyungtae Han is a principal researcher in Toyota InfoTech Labs, 465 N Bernardo Ave, Mountain View, CA, U.S.A kyungtae.han@toyota.com

recognition [1]. In this work, we also use an RNN structure as part of our proposed framework.

A principal weakness of existing driving behavior prediction methods is that they use models that require inputs of fixed size and fixed spatial organization, making it difficult to generalize from training sets into practice. In [2], for instance, the proposed method uses a leader-follower model that focuses only on the interactions between the ego-vehicle and its leading vehicle. More recently, neighbor models that capture more interactions between ego-vehicle and its surrounding vehicles have been proposed [3], [4]. Though these neighbor methods show some success in predicting the ego-vehicle’s future acceleration, they only consider a fixed number of neighbor vehicles. In addition, they need to deal with information padding if one of the pre-defined neighbors is absent.

Graph neural networks (GNNs) are a type of neural network designed for the analysis of graphs [5]. Recently, GNNs have been drawing increasing attention from both academia and industry for the flexibility that the graph data structure provides and for their convincing performance on various tasks in different domains, such as social science [6], [7], neural science [8], and knowledge graphs [9]. For instance, motivated by a first-order approximation of spectral convolution on a graph, Graph Convolution Networks (GCNs) are a computationally efficient variant of GNNs that have shown success in achieving fast and scalable classification of nodes in a graph [7]. Another class of GNNs is the Graph Attention Network (GAT), which utilizes *self-attention* [10] to allow for inductive reasoning among nodes, thereby providing additional interpretability while matching other GNNs on benchmark evaluation.

In this paper, we propose a flexible driving behavior prediction framework that we call the *Traffic Graph Framework*. Combining GCNs and LSTMs, our proposed method is able to capture not only spatial features of various sizes but also temporal features. This framework consists of undirected graphs that represent the interactions between vehicles, a multi-layer graph convolution neural network used to directly encode the graph structure, and a fully-connected or LSTM mixture density network used to predict future acceleration distributions.

In series of empirical tests, we investigate the the performance of our proposed models relative to baselines, including GAT and other GCN variants. The test environment for our methods is a simulation designed to mimic real-world traffic. The simulation is built using the NGSIM I-80 dataset, which contains vehicle trajectories of more than

2000 individual drivers [11]. In the simulation, ego-vehicles' traffic states are propagated based on models' predictions. Models are evaluated by comprehensive metrics to measure the discrepancy between the generated trajectories and the ground truth. Furthermore, ablation studies were performed to analyze the effectiveness of the proposed GCNs and RNN architectures. Results show that including the proposed GCNs and RNN structure improves model's prediction quality.

Our principal contributions are three-fold:

- We propose a graph structure to denote vehicle's spatial relationships in a dynamic traffic environment. Our structure supports modeling at fine time scales and can be scaled to include an arbitrary number of neighbors for the ego vehicle.
- We introduce new variants of GCN layer-wise propagation rule in the context of traffic modeling and we propose a new acceleration prediction framework combining GCNs and LSTM. We successfully applied our framework to a simulation built from real-world data. The resulting systems outperform others from the literature on the task of acceleration prediction.
- We demonstrate that GAT models fail to make accurate acceleration predictions. This result is significant because GATs have been successful in other traffic modeling settings, notably the work by Diehl et al. [12]. From an investigation of the attention weights generated by the *self-attention* mechanism, we identify the causes of GAT underperformance on our problem.

The rest of this paper is organized as follows: In the Related Work section, we summarize prior arts. In the Methodology section, we introduce our framework and our proposed GCN variants. In the Experiment section, we present the training procedure, baselines, and simulation results for all models. In the Discussion section, we elaborate on our findings about GCNs and LSTM in the experiment and demonstrate why GAT fails to generate realistic trajectories. The final section concludes the study.

II. RELATED WORK

The task of modeling driving behavior consists of modeling car-following behavior and lane-changing behavior. In our work, we focus on augmenting the car-following model.

Car-following models capture the interaction between the ego-vehicle and the vehicles directly adjacent on the microscopic level of the traffic. Based on the number of interactions captured, models can be categorized as being either a single-lane or multiple-lanes.

A single-lane model focuses on the interactions between vehicles in a single lane. This model considers up to two kinds of interactions: namely, the ego-vehicle with its leading vehicle, and the ego-vehicle with its following vehicle. Many traditional fixed-form models fall into this category, including the Gazis-Herman Rothery model [13], the collision avoidance model [14], linear models [15], psycho-physical models [16], and fuzzy logic-based models [17].

Some recent general driving models have moved away from making assumptions about drivers. Lefvre et al. compare the performance of feed-forward mixture density network against traditional baselines [18]. Their empirical tests suggest that the proposed method is able to achieve performance comparable to the baselines. Morton et al. study the effectiveness of LSTM in predicting driving behavior on highways. They reveal that the LSTM's ability to remember historic states of the ego-vehicle appears to be the key to achieving the state-of-art performance [2].

More recently, multiple-lane models that consider more interactions, coupled with neural networks, have been introduced in the literature. Kim et al. propose a framework based on LSTM to predict vehicle's future position over the occupancy grid [19]. Altche et al. use LSTM that predicts traffic using as input state information on the ego-vehicle states and up to 9 of its neighbors. The model is trained and evaluated on the NGSIM 101 dataset which has trajectories from more than 6000 individual drivers [3].

Diehl et al. [12] used GNNs for vehicle coordinates prediction and demonstrated that GAT models outperform other baselines. Note that our method differs from that work in three main ways. First, we are interested in generating vehicle trajectories. Hence, our models are structured to predict 0.1-second future acceleration, which can be propagated to vehicle trajectories of any length with velocity, acceleration, and coordinates information. Diehl et al. aim to predict the 5-second-later coordinates of a vehicle, which contains limited information for the construction of realistic vehicle trajectories. Second, our framework allows for including an arbitrary number of neighbors. The models of Diehl et al., by contrast, consider only up to 8 neighbors, which might result in ignoring important information about the state of traffic around ego-vehicles. Third, we believe that information of ego-vehicle's past states affects future actions. Diehl et al. does not consider RNN structure, whereas we include this structure because it acts to memorize the past states of a vehicle. Furthermore, we analyze the performance of GNNs with and without RNNs.

III. METHODOLOGY

This section describes the construction of traffic graphs and our proposed graph convolution network variants.

A. Traffic Graph and Features

To leverage the spatial relationships and interactions between vehicles on the highway, we use an undirected graph $G = (E, V)$ with N nodes $v_i \in V$, edges $(v_i, v_j) \in E$, an adjacency matrix $A \in \mathbb{R}^{N \times N}$, a degree matrix with $D_{ii} = \sum_j A_{ij}$, and a nodes feature information matrix $X \in \mathbb{R}^{N \times F}$ to model the interactions between vehicles. As shown in Figure 1, for a vehicle pair (v_i, v_j) where $v_i \in V$ and $v_j \in V$, the edge (v_i, v_j) is connected if and only if:

- vehicle v_j and v_i appear at the same frame; and
- vehicle v_j is less than one lane away from vehicle v_i at the current frame (vehicle v_j should be on the same lane with vehicle v_i or on vehicle i 's left, right lanes); and

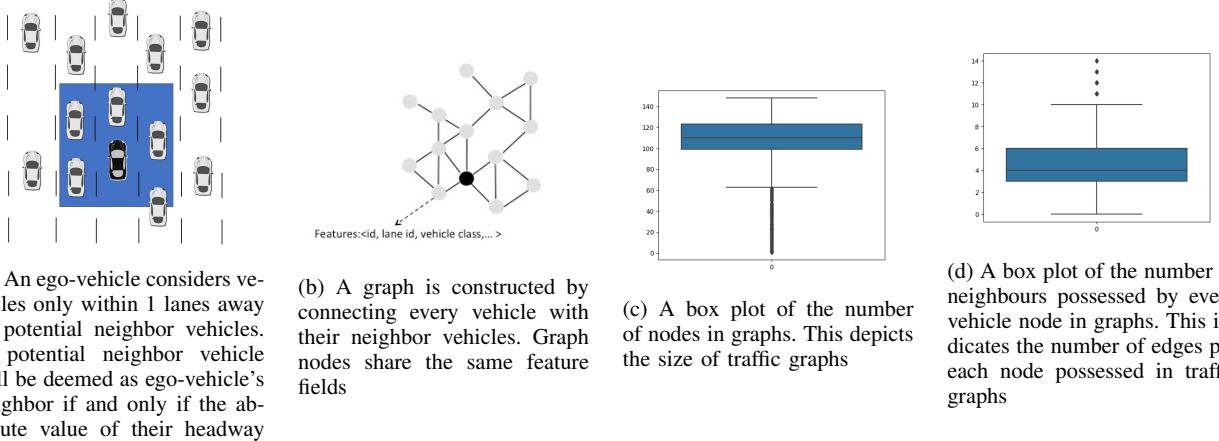


Fig. 1: Mapping from real world traffic to traffic graph

- the absolute value difference of vehicle v_j 's y-coordinate and vehicle v_i 's y-coordinate is less than the designated value τ at the current frame.

Note that there is no fixed limit on the number of neighbors; all vehicles within an ego-vehicle's designated distance τ are its neighbor vehicles. In NGSIM I-80 dataset, the traffic of the study area changes frequently. The traffic, hence, is updated at the same frequency as data was collected in the original dataset. Figure 1c, and Figure 1d present statistics regarding graphs.

In this work, we adopt the features used in [4]. For a vehicle node in the graph at frame t , its feature vector includes the following elements: vehicle lane id l_t , vehicle class id c , vehicle velocity v_t , vehicle acceleration a_t , relative distance from 3 nearest front neighbor vehicles $\{d_{f_1}, d_{f_2}, d_{f_3}\}$ (pad τ if the number of front neighbors is smaller than 3), and negative relative distance from 3 nearest rear neighbor vehicles $\{-d_{r_1}, -d_{r_2}, -d_{r_3}\}$ (pad $-\tau$ if the number of rear neighbors is less than 3).

B. Graph Convolution Network

GCN takes input as a graph G and output nodes encodings. We consider the propagation rule originally introduced in [7] as our base model:

$$H^{l+1} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^lW^l), \quad (1)$$

where $\hat{A} = A + I_N$ is the summation of the undirected graph G 's adjacency matrix with binary entries A and self-connection $I_N \in \mathbb{R}^N$, $I_N \in \mathbb{R}^N$ is a identity matrix, D is a degree matrix with $D_{ii} = \sum_j A_{ij}$, $W^l \in \mathbb{R}^{N \times C^l}$ is a matrix of trainable weights at depth l , σ is an activation function, and H^l is the encoding of all nodes in the graph at depth l ($H^0 = X$).

This layer-wise propagation rule can be rewritten in the

following vector form:

$$h_{v_i}^{l+1} = \sigma\left(\sum_j \frac{h_{v_j}^l}{c_{ij}} W^l + \frac{h_{v_i}^l}{c_{ii}} W^l\right). \quad (2)$$

Here, j indexes neighboring nodes of v_i , normalization factor $\frac{1}{c_{ij}}$ is an entry located at the i th row, j th column of $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$.

The propagation rule represented by Equation 1 is a first-order approximation of spectral convolution on a graph. It provides two advantages when used to analyze graphs: first, it enables to aggregate l^{th} order neighborhood of a central node during the encoding process; second, it prevents us from prohibitively expensive eigendecomposition of the graph Laplacian compared with spectral convolution models [7]. Those properties offer us a computational efficient approach to learn the interactions between vehicles that are not directly connected in the graph.

Ego-discriminated GCN (EGCN): During the implementation of the base model, we find that self-connection affects the performance of the system, an observation that leads to our adaptation of the base model. Self-connection was used to alleviate the problem of vanishing/exploding gradients in GCNs [7]. However, this method applies the same weight W^l to both the central node and its surrounding nodes. In our experiments, we find it is beneficial to remove the self-connection and apply different layer weights to discriminate the central node from its surrounding node. This leaves us with the ego-discriminated propagation rule, which can be represented as follows:

$$H^{l+1} = \sigma\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^lW^l + I_NH^lB^l\right), \quad (3)$$

where $I_N \in \mathbb{R}^N$ is an identity matrix, $B^l \in \mathbb{R}^{N \times C^l}$ is trainable weights at depth l for central nodes. The corresponding

vector form is given in the following expression:

$$h_{v_i}^{l+1} = \sigma \left(\sum_j \frac{h_{v_j}^l}{c_{ij}} W^l + \frac{h_{v_i}^l}{c_{ii}} B^l \right). \quad (4)$$

C. Distance-Aware Graph Convolution Network

For the models mentioned in the previous section, their adjacency matrices \hat{A} and A only denote whether a pair of vehicles is close or not, but they do not describe the degree of closeness. Based on our empirical driving experience—the closer our neighbor vehicle is, the more attention we will pay to it—we use absolute inverse relative distances as entries for our adjacency matrix \tilde{A} to differentiate the degree of closeness between vehicles. Therefore, we introduce the following distance-aware layer-wise propagation rule in our multi-layer GCN (DGCN):

$$H^{l+1} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^l W^l + I_N H^l B^l \right). \quad (5)$$

Here, \tilde{A} is an adjacency matrix with $\tilde{A}_{ij} = \frac{1}{|y_{v_i} - y_{v_j}|}$ where y_i represents vehicle v_i 's y-coordinate. \tilde{D} is a degree matrix with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. In this propagation rule, \tilde{A} 's entries denote the degree of closeness between vehicles. To stabilize gradients during training, we discretize the degree of closeness into three levels: 1, 2, and 3, which represent far away, medium close and very close, respectively. Equation 5 can also be rewritten in the following vector form:

$$h_{v_i}^{l+1} = \sigma \left(\sum_j \frac{h_{v_j}^l}{\tilde{c}_{ij}} W^l + h_{v_i}^l B^l \right), \quad (6)$$

where \tilde{c}_{ij} is an entry located at i th row and j th column of $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$.

D. Gaussian Mixture Model

In this work, we aim to predict human driver's acceleration distribution given the current traffic state. Hence the output of our network model is Gaussian mixture model (GMM) parameters that characterize the future acceleration distribution. This *mixture density network* (MDN) is first proposed by Bishop [20] and been successfully applied in speech recognition and other fields [21]. For a K -component GMM, the probability of the predicted acceleration follows this equation:

$$p(a) = \sum_{i=1}^K w_i \mathcal{N}(a | \mu_i, \sigma_i^2), \quad (7)$$

where w_i , μ_i , and σ_i are the weight, mean, standard deviation of the i th mixture component respectively.

IV. EXPERIMENT

A. Dataset

The NGSIM I-80 dataset contains detailed vehicle trajectory data collected using synchronized digital video cameras on eastbound I-80 in Emeryville, CA. This dataset provides precise positions, velocities and other vehicle information over three 15-minute periods at 10 Hz. The study area

covers approximately 500 meters in length and consists of six freeway lanes, including a high-occupancy lane and an on-ramp lane. We use the NGSIM I-80 reconstructed dataset, which contains vehicles position, velocity, acceleration from 4:00 p.m. to 4:15 p.m., because it corrects errors such as extreme acceleration, and inconsistent vehicle IDs [22] [23]. We split the data into training sets and testing sets by a ratio of 4 to 1.

B. Data Preparation

Both training set and testing set are divided into 12-second segments (120 frames). The first 2-second segments (20 frames) are used to initialize the internal state of LSTM networks. Since the aim of the research is to predict driving acceleration using GCNs, we need to prepare traffic graphs from the raw data.

C. Baselines

Our proposed models are compared with the following non-GNN models and GAT models.

Fully-connected (FC): This model shares the same configuration and input features as the GCN without LSTM models.

LSTM: This model's configuration and input features are the same with GCN-LSTM models.

Our proposed models use heuristics to define normalization factors between nodes. For instance, DGCN uses inverse distance as entries for adjacency matrix A based on the heuristic that the ego-vehicles should pay more attention to closer neighbors compared with distant neighbors. In contrast, GAT, which applies *self-attention*, learns to generate the normalization factors for neighbouring nodes rather than resorting to weights in adjacency matrix A :

$$\alpha_{ij} = \frac{\exp \left(\text{LeakyReLU}(BW_a h_i + WW_a h_j) \right)}{\sum_{k \in N_i} \exp \left(\text{LeakyReLU}(BW_a h_i + WW_a h_k) \right)}, \quad (8)$$

where i indexes the central node, j indexes the surrounding nodes, W_a , B and W are weights in self-attention with B applied to central nodes and W applied to surrounding nodes, LeakyReLU is an activation function, and α_{ij} is equivalent to normalization factor $\frac{1}{c_{ij}}$ mentioned in previous equations. Following the practice in [12], we utilized different layer weights, B and W , to attend to central and neighbouring nodes respectively. *Self-attention* weights W_a , B , and W are updated such that GAT learns how to distribute α_{ij} .

GAT: This model shares the same configuration and input features as other models without LSTM.

GAT with LSTM: This model's configuration and input features are the same with other LSTM models.

D. Implementation

All models are trained to output predicted parameters for distributions over future acceleration values. Note that every model in this work shares the same hyperparameters because we aim to compare the effectiveness of GNN and LSTM on

improving model performance in the task of driver behavior prediction. We set $\tau = 20$ feet, empirically.

Model structures are shown in Table 7. Each model consists of 3 hidden layers and a 30-component MDN layer. Layer 1 applies Relu activation while other layers do not use any activation. Layer 1 and layer 2 are followed by batch normalization. Batch normalization is a mechanism to address the problem of *internal covariate shift*. It has been reported that adding batch normalization to state-of-the-art image classification networks yields higher classification accuracy compared with the original networks [24]. The performance of our models is also found to improve when batch normalization is applied. Layer 3 is either an FC layer or an LSTM layer. The final 30-component MDN layer follows layer 3 and has an output size of 90, which corresponds to a 30-component GMM's parameters.

All models are trained for 5 epochs. During training, the models are optimized by the Adam optimizer with a learning rate of 1×10^{-3} [25]. A dropout of 10 percent is applied to help prevent overfitting. Gradient norm clipping is also used to deal with gradient vanishing and gradient explosion [26]. All networks are implemented in TensorFlow [27] based on Kipf's GCN package [7] and Veličković's GAT package [28].

E. Evaluation

Once trained, each model is used to generate simulated trajectories. For every trajectory in the test set, the first 2-second segments (20 frames) of true data are used to initialize LSTM's internal state. In the following 10 seconds, ego-vehicle's velocity and position can be updated by assuming the following equations:

$$\begin{aligned} v(t + \delta t) &= v(t) + a(t + \delta t) \times \delta t \\ y(t + \delta t) &= y(t) + v(t + \delta t) \times \delta t, \end{aligned} \quad (9)$$

where v is ego-vehicle's velocity, y is ego-vehicle's Y -coordinate and a is vehicle's acceleration. The graph and node features are updated by propagating other vehicles' true trajectory data and ego-vehicle's simulated trajectory. Following the practice in [2], we evaluate the quality of simulated trajectories by the following metrics:

- *Root Mean Squared Error (RMSE)*: We use root mean squared error to evaluate the discrepancy of speed values between simulated trajectories and true trajectories at designated horizons for a given ego-vehicle:

$$RMSE_{velocity} = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (v_H^i - \hat{v}_H^{i,j})^2}, \quad (10)$$

where m is the number of true trajectories, $n = 20$ is the number of simulated trajectories per true trajectory, v_H^i is the velocity of i th true trajectory at horizon H , $\hat{v}_H^{i,j}$ is the value in j th simulated trajectory at time horizon H . Similarly, we also use root mean squared error to evaluate the displacement in Y -coordinate at 10 second horizon between simulated trajectories and true

trajectories:

$$RMSE_Y = \sqrt{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n (y_{10}^i - \hat{y}_{10}^{i,j})^2}, \quad (11)$$

where y_{10}^i is the Y -coordinate of i th true trajectory at 10 second, $\hat{y}_{10}^{i,j}$ is the simulated Y -coordinate value for sample j in the i th trajectory at 10 second horizon.

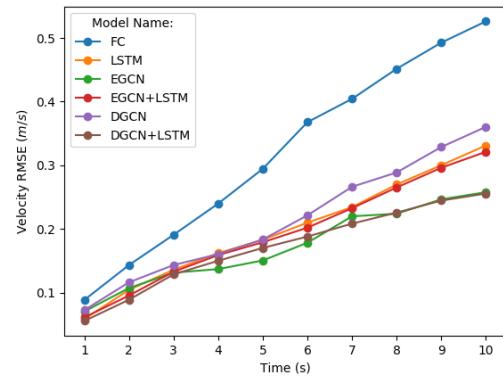


Fig. 2: RMSE results for all models

Figure 2 shows the velocity RMSE for the top 6 models over prediction horizons between 1 and 10 seconds. Models with original GCN [7] and GAT [28] are not included because of their bad performance in generating predicted trajectories. In general, the velocity RMSE accumulates over the time horizon. Our adapted GCN models outperform non-GCN models. For non-GCN models, LSTM outperforms the fully-connected model because LSTM is able to access past information. For GCN models, EGCN model and DGCN with LSTM outperform other GCN models.

The Y -coordinate RMSE column in Table II denotes the displacement in Y -coordinate between simulated trajectories and their corresponding true trajectories. EGCN model outperforms other models. Velocity RMSE at 10 second horizon reveals the discrepancy of speed between simulated trajectories and the ground truth. DGCN with LSTM outperforms other models in this metric.

- *Negative Headway Distance Occurrence*: This metric is used to evaluate models' robustness. It records the occurrences of unrealistic states led by models' poor decision making. Two types of negative headway distances are considered: (1) ego-vehicle's negative headway distance representing collisions with the front vehicle; and (2) following vehicle's negative headway distance denoting collisions between the ego- and its following vehicle. A robust model will have minimal negative headway distance occurrence.

Table III shows the number of negative headway occurrences over number of simulated trajectories for all models. Consistent with RMSE analysis, the results

TABLE I: Model Configuration

Model	layer 1	layer 2	layer 3	MDN layer	LSTM	clip norm	adjacency type
Fully-connected	128	256	128	90	no	5	/
	GCN base	128	256	128	90	no	5
	GAT	128	256	128	90	no	5
	EGCN	128	256	128	90	no	5
	DGCN	128	256	128	90	no	5
LSTM	128	256	128	90	yes	5	/
	GCN with LSTM	128	256	128	90	yes	5
	GAT with LSTM	128	256	128	90	yes	5
	EGCN with LSTM	128	256	128	90	yes	5
	DGCN with LSTM	128	256	128	90	yes	5

TABLE II: RMSE Analysis

Model	Y RMSE @ 10 s (m)	Velocity RMSE @ 10 s (m/s)
Fully-connected (FC)	2.89	0.526
	3.52	0.622
	4.13	0.688
	1.40	0.258
	1.91	0.360
LSTM	1.61	0.331
	3.40	0.653
	4.09	0.728
	1.86	0.321
	1.63	0.256

from Table III demonstrates that original GCN models often produce poor acceleration predictions which lead to unrealistic states. EGCN model and DGCN with LSTM model are robust because there are no unrealistic states occurring in their simulated trajectories.

- *Jerk Sign Inversions*: We use the number of jerk sign inversions per trajectory to evaluate the similarity between the smoothness of the true and simulated trajectories. This metric is used to quantify oscillations in model’s acceleration predictions.

Simulated trajectories of most of models have slightly higher jerk sign inversions than the true trajectories while the LSTM baseline model is not able to generate smooth trajectories. In addition, jerk sign inversions, combined with previous metrics, indicate that the trajectories generated by GAT with LSTM model fail to react against the changes of the ego-vehicle’s surrounding traffic.

Figure 3 shows the sample simulated trajectories by models, including adapted GCN models and non-GCN models. It can be seen that non-GCN models predict poorly if the ground truth trajectory includes a long period of acceleration values that are very close to zero while GCN models is able to generate smooth trajectories close to the ground truth. In addition, non-GCN models are prone to predict extreme acceleration values, which is compensated by oscillation of acceleration values.

V. DISCUSSION

Our experiments are designed to answer the following research questions:

- Does GCN improve model performance and are our adaptations to GCN beneficial?
- Does including LSTM increase prediction quality?

- Why do GAT models fail to generate realistic trajectories?

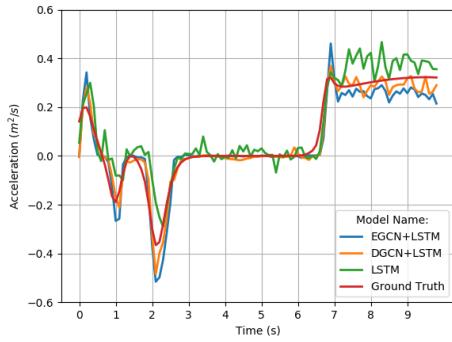
First, we discover that we improve GCN’s performance when we delete self-connections and apply different weights to the central nodes and their surrounding nodes. For GCN base model, we reduced velocity RMSE by 58.5% at 10 seconds horizon and negative headway occurrence by 17% during simulation. For GCN with LSTM model, we reduced its 10 seconds horizon velocity RMSE by 50.8% and negative headway occurrence by 15%.

Second, our experiments demonstrated that GCNs improve model performance. GCN models are able to generate smooth and robust trajectories close to the ground truth. For both LSTM and fully-connected models, the non-GCN baseline model is outperformed by its GCN counterparts, in general. Note that, in the experiments, our GCN models and non-GCN models share the same number of hidden layers and the same number of neurons in each hidden layer. Compared with non-GCN fully-connected model, our EGCN model reduced the negative headway occurrence rate from 0.08 to 0, 10 seconds horizon velocity RMSE by 59.6%. Compared with non-GCN LSTM, our DGCN with LSTM reduced the negative headway occurrence rate from 0.02 to 0, jerk sign inversions from 13.7 to 7.3 and 10 seconds horizon velocity RMSE by 22.7%. This trend can also be observed in Figure 3. The multi-layer GCN’s ability to capture multitudes of interactions between vehicles hierarchically improves model’s prediction quality in terms of our evaluation metrics.

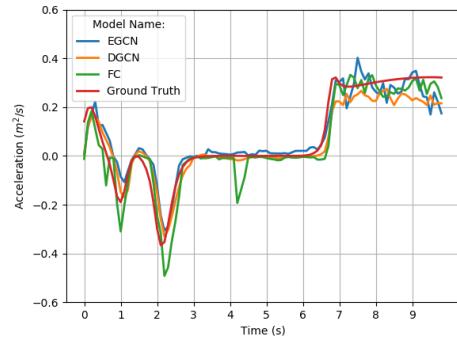
In general, we find that adding LSTM structure improves model prediction quality. Among all models, the best model is DGCN with LSTM. During simulation, this model is able to generate robust and smooth driving trajectories with 0 negative headway, 7.3 jerk sign inversions and 0.256 for 10-

TABLE III: Jerk Sign Inversions Per Trajectory

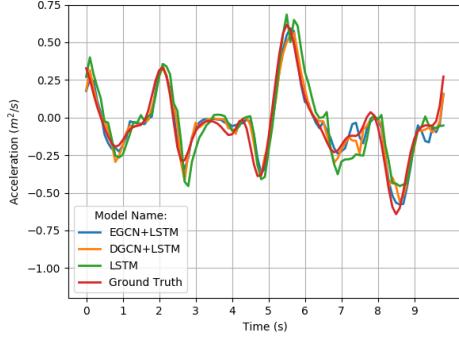
Model	Jerk Sign Inversions	Negative Headway Occurrence Rate
Fully-connected (FC)		
GCN base	7.5	0.08
GAT	5.9	0.17
EGCN	7.5	0.27
DGCN	7.3	0.03
LSTM	13.7	0.02
GCN with LSTM	6.7	0.17
GAT with LSTM	0.0	0.27
EGCN with LSTM	9.5	0.01
DGCN with LSTM	7.3	0
True trajectory	6.3	/



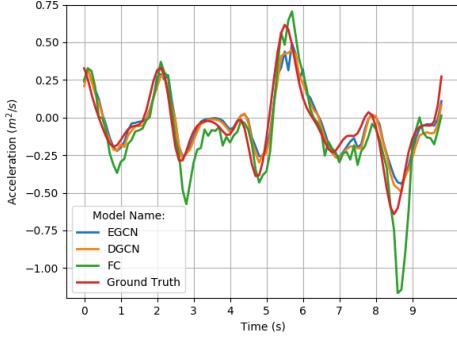
(a) LSTM models



(b) Fully-connected models



(c) LSTM models



(d) Fully-connected models

Fig. 3: Simulated Trajectories For All Models (Original GCN and GAT models are excluded for their bad performance)

second horizon velocity RMSE.

GATs utilize self-attention to assign attentional weights α_{ij} for neighbouring node j . The attentional weights α_{ij} indicate the relationship between the central node and its surrounding nodes. Following [10], we investigated α_{ij} to understand why GAT models fail in our experiment. The investigation shows that the relational kernel in the baseline GAT models fails to learn the relationships between central nodes and their surrounding nodes. From the sample in the vehicle 829 at the frame 2373, the relational kernel in the second GAT layer of the GAT model assigns the same weights to every node: vehicle 829 initially has two neighbours, 818 and 835. The

attentional weights for each node, including the central node 829, is 0.333. Later in the trajectory, another vehicle 795 approaches the ego-vehicle 829 and the attentional weights assigned to all 4 nodes are 0.25.

VI. CONCLUSION

In this paper, we propose the use of graphs defined by the spatial relationships between vehicles, to model traffic. We further build GCN models, operating on graphs, to predict future acceleration distributions. We propose two GCN models adapted from the state-of-art GCN and studied the effectiveness of LSTM architectures in our prediction

models. Our resulting frameworks outperform others on the task of acceleration prediction.

While our proposed methods have been shown to improve prediction performance, much work remains to be done. This work can be extended to prediction in two dimensions, which is an important problem in autonomous driving. At the same time, it will be interesting to evaluate different graph construction strategies, such as strategies that include multiple layers of relationships.

REFERENCES

- [1] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [2] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, “Analysis of recurrent neural networks for probabilistic modeling of driver behavior,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1289–1298, 2016.
- [3] F. Altché and A. de La Fortelle, “An lstm network for highway trajectory prediction,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 353–359.
- [4] D. Lenz, F. Diehl, M. T. Le, and A. Knoll, “Deep neural networks for markovian interactive scene prediction in highway scenarios,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 685–692.
- [5] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, “Graph neural networks: A review of methods and applications,” *arXiv preprint arXiv:1812.08434*, 2018.
- [6] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.
- [7] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [8] A. Fout, J. Byrd, B. Sharifi, and A. Ben-Hur, “Protein interface prediction using graph convolutional networks,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6530–6539.
- [9] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, “Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach,” *arXiv preprint arXiv:1706.05674*, 2017.
- [10] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.
- [11] J. Colyar and J. Halkias, “Us highway 80 dataset, federal highway administration (FHWA), vol,” *Tech. no. Rep.*, 2006.
- [12] F. Diehl, T. Brunner, M. T. Le, and A. Knoll, “Graph neural networks for modelling traffic participant interaction,” *arXiv preprint arXiv:1903.01254*, 2019.
- [13] R. E. Chandler, R. Herman, and E. W. Montroll, “Traffic dynamics: studies in car following,” *Operations research*, vol. 6, no. 2, pp. 165–184, 1958.
- [14] E. Kometani, “Dynamic behavior of traffic with a nonlinear spacing-speed relationship,” *Theory of Traffic Flow (Proc. of Sym. on TTF (GM))*, pp. 105–119, 1959.
- [15] W. Helly, “Simulation of bottlenecks in single-lane traffic flow,” in *International Symposium on the Theory of Traffic Flow, New York, 1959*.
- [16] R. Michaels, “Perceptual factors in car-following,” *Proc. of 2nd ISTTF (London)*, pp. 44–59, 1963.
- [17] S. Kikuchi and P. Chakraborty, “Car-following model based on fuzzy inference system,” *Transportation Research Record*, pp. 82–82, 1992.
- [18] S. Lefèvre, C. Sun, R. Bajcsy, and C. Laugier, “Comparison of parametric and non-parametric approaches for vehicle speed prediction,” in *2014 American Control Conference*. IEEE, 2014, pp. 3494–3499.
- [19] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, “Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 399–404.
- [20] C. M. Bishop, “Mixture density networks,” *Technical report*, 1994.
- [21] T. Robinson, M. Hochberg, and S. Renals, “The use of recurrent neural networks in continuous speech recognition,” in *Automatic speech and speaker recognition*. Springer, 1996, pp. 233–258.
- [22] M. Montanino and V. Punzo, “Making ngsim data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction,” *Transportation Research Record*, vol. 2390, no. 1, pp. 99–111, 2013.
- [23] ———, “Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns,” *Transportation Research Part B: Methodological*, vol. 80, pp. 82–106, 2015.
- [24] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.
- [25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [26] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *International conference on machine learning*, 2013, pp. 1310–1318.
- [27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., “Tensorflow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.
- [28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.