2023 IEEE 26th International Conference on
Intelligent Transportation Systems (ITSC)
24-28 September 2023. Bilbao, Bizkaia, Spain

Physics-guided Graph Convolutional Network for Modeling Car-following Behaviors under Distributional Shift

Htet Naing[1], Wentong Cai[1], Jinqiang Yu[2], Tiantian Wu[2] and Liang Yu[2]

***Abstract*— Recent works on data-driven car-following modeling have shown that a Graph Convolutional Network (GCN)-based car-following model (CFM) can outperform other data-driven models such as Long Short-term Memory (LSTM) networks. Inspired by this result, a new physics-guided GCN-based CFM is proposed in this paper. The model has been extended from the previous GCN-based CFM by integrating a vehicle platoon graph along with other refinements. Furthermore, the first attempt to develop physics-guided graph-learning-based CFM is made in this paper by adopting physics-guided machine learning (PGML) in different modeling aspects. The effectiveness of the proposed model was demonstrated in both in-distribution and out-of-distribution scenarios, yielding safe and accurate car-following behaviors. The experiment results showed that under the most severe distributional shift, the proposed model outperformed the best pure data-driven model by approximately 26.25% and a well-calibrated physics-based model by about 11.45% in terms of collision-penalized gap root mean squared error. Future research can focus on improving the robustness of our proposed model under various distributional shifts, leveraging the insights and suggestions given in the paper.**

I. INTRODUCTION

Car-following (CF) behavior modeling has been an active area of research for nearly six decades [1]. It is used to model the longitudinal dynamics of a vehicle and has found many real-world applications from microscopic traffic flow simulation [2] to longitudinal motion planning in autonomous vehicles [3].

Traditional CFMs are physics-based in nature and formulated based on traffic flow theory with their model parameters having meaningful physical interpretation [4]–[6]. With recent advances in machine learning (ML), researchers have adopted learning-based approaches for modeling CF behaviors under the umbrella term, *data-driven models* [7]–[9]. They can be classified into two categories – 1) *behavior cloning* where a CFM is learned via human-driven trajectories in a supervised learning manner, and 2) *reinforcement learning* (RL) where an agent attempts to learn CF behaviors by interacting with a simulation environment in a trial-and-error manner. The review of these models can be found in a recent survey paper [3]. At the core of both types of learning exist various deep learning (DL) models used for accurate approximation of CF behaviors based on certain input features. Hence, this paper focuses on the development of a new supervised DL-based CFM trained in a behavior cloning manner, and its extension with RL is left for future works.

[1]Htet Naing and Wentong Cai are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore.
[2]Jinqiang Yu, Tiantian Wu, and Liang Yu are with City Brain Lab, Alibaba Cloud, China.

In the field of DL, graph neural networks (GNN) are gaining popularity since they can effectively capture relationships and interactions among entities modeled as a graph structure. Accordingly, recent works have utilized GNNs to capture traffic participant interaction [10], and car-following modeling [11]. The key idea is to model the surrounding vehicles that are traveling nearby an ego vehicle as a graph (including the ego at the center) with the vehicle states as nodes, and their relative information as edges. It has been demonstrated in [11] that a CFM which is developed based on a type of GNN, namely *Graph Convolutional Network (GCN)*, can actually outperform other standard data-driven models such as *Long Short-term Memory (LSTM)* networks. Inspired by their initial results, our first objective is to improve their GCN-based CFM by addressing the shortcomings discussed in Section II-A.

One major limitation shared by all data-driven CFMs trained in a behavior cloning manner is that they are vulnerable to *distributional shift* [12], [13], a phenomenon in which an ML model is unable to generalize as expected when there is a mismatch between training and test data distributions. Generally, three approaches are viable to tackle this issue – 1) *physics-guided machine learning (PGML)* [14], [15]; 2) *reinforcement learning (RL)* [16], [17]; and 3) *data augmentation (DA)* [16], [18]. To the best of our knowledge, none of the studies has investigated the performance of data-driven CFMs explicitly under distributional shift. Therefore, in this paper, by conducting controlled experiments, we take the initiative to study their performance under both in-distribution and out-of-distribution scenarios.

As our second objective, we aim to enhance our improved model with PGML in order to tackle distributional shift. This extension is motivated by the PGML's potential to compensate for the inadequacy of training data (in terms of coverage, pre-processing, biases, etc.) by incorporating physics into the data-driven model. Further study on the usages of the other two approaches (*RL* and *DA*) for tackling distributional shift is left for future works. Our main contributions are:

1) An improved CFM based on GCN is developed by considering a vehicle platoon graph (VPG) as the model input along with other refinements.
2) A new physics-guided data-driven CFM is proposed by extending the above model for robustness under distributional shift.
3) An extensive set of experiments were conducted to show the effectiveness of the proposed model under both in-distribution and out-of-distribution scenarios.

The rest of the paper is organized as follows: Section II describes related works and their research gaps. Our proposed CFM is presented in Section III. The experiment results are analyzed in Section IV. Finally, the paper concludes with a summary and future research recommendations in Section V.

II. RELATED WORKS

A. Data-driven Car-following Models

The development of data-driven CFMs began with classic ML techniques such as *Locally Weighted Regression* [19], *Random Forest* and *Artificial Neural Network* [20]. They have been shown to outperform physics-based models such as Gipps' [5] due to their flexibility to easily incorporate more relevant input features. With the rapid advances made in deep learning (DL), more sophisticated DL models, such as a family of recurrent neural networks (RNN) – Gated Recurrent Unit (GRU) [7], and Long Short-term Memory (LSTM) [9], have been adopted to model CF behaviors. These models surpass the performance of classic ML models because their recurrent architecture can incorporate driver memory effects by modeling temporal dependency among past vehicle states.

The above ML/DL-based CFMs are only able to model fixed spatial input features (e.g., grid or sequence), but not ideal for modeling flexible spatial input representations such as graphs that can capture rich interactions and relationships among entities. Accordingly, deep neural networks that can operate on graphs, namely *Graph Neural Networks (GNN)*, have emerged. Many successful applications in Intelligent Transportation Systems (ITS) follow, ranging from traffic flow prediction [21], trajectory prediction [10], and car-following modeling [11].

In [11], a CFM developed based on GCN has outperformed baselines such as LSTM, thereby unlocking the potential of GNNs for modeling CF behaviors. However, there exist a number of limitations that need to be addressed. First, although the RNN structure was integrated into their GCN to capture the driver memory effect, it only considered the past 2-second trajectory. This is contrary to an existing work [18] that emphasizes the importance of long memory (i.e., past 10-second trajectory) for modeling the CF process. As a result, the performance of their GCN with RNN structure was worse in terms of simulated position error than the one without RNN structure. Second, their chosen step size was 0.1 second, which could be too short to account for driver reaction time [7]. Third, their CF simulation duration was only 10 seconds long, and hence, there still remains a question of whether the model can perform well under a long CF period which is typically considered as 15-60 seconds in literature [8], [16], [22]. Lastly, the model was not benchmarked against any well-calibrated physics-based model. Hence, it is unclear whether the model can maintain its performance for longer CF duration when compared to physics-based models.

To address the above limitations, an improved CFM is proposed in this paper by extending the initial GCN-based CFM developed in [11].

B. Physics-guided Data-driven Car-following Models

All the data-driven CFMs discussed above are vulnerable to *distributional shift* [12], [13], especially when they are trained via behavior cloning. This could lead to model performance degradation when evaluated using unseen scenarios/examples which are not covered in the training data. One viable approach to tackle this issue is *Physics-guided Machine Learning (PGML)* [23] where classic physics-based and modern learning-based approaches are integrated to develop models that surpass the performance of either one. Many successful applications of PGML have been highlighted in a recent survey [23], and its potential extensions with GNNs are covered in [24].

In the CFM literature, only very few works that leverage PGML exist despite its promising potential. The pioneering work in this direction has been done in [14] by developing a data-efficient CFM that could produce more accurate CF behaviors (with only a few training samples) than those without PGML. However, the potential of PGML for tackling distributional shift was not mentioned in [14]. In our previous work [15], PGML was adopted along with online machine learning for modeling dynamic CF behaviors in pseudo-real-time microscopic traffic simulation. Through our experiment results, we discovered the potential of PGML for tackling distributional shift in CF modeling, and briefly analyzed its implications on the results. However, it was not expanded any further. None of the existing (CFM) works has attempted to incorporate PGML into the development of GNNs yet. Furthermore, the performance of such a model under distributional shift has also not been studied.

To fill the above research gaps, our improved CFM variant is also enhanced with PGML so that it can achieve both safe and accurate CF behaviors not only for in-distribution but also for out-of-distribution scenarios.

III. PROPOSED CAR-FOLLOWING MODEL

A. Formulation of a Data-driven CFM

Building upon the definitions given in [7], [15], a discrete-time CFM represented with a function f that considers past vehicle states up to the k^{th} window length as its inputs can be defined as:

$$a_{t+1} = f(X_{t-k+1}, X_{t-k}, ..., X_t|\Theta) \tag{1}$$

$$a_{t+1} \in [a_{LB}, a_{UB}]$$

Given input features containing the past states of a vehicle at each time step, $X_{[t-k+1,t]} = \{X_{t-k+1}, X_{t-k}, ..., X_t\}$, and parameters Θ, a CFM produces longitudinal acceleration a to be taken in the next time step $t+1$ after accounting for reaction time. The goal of the data-driven CFM is to learn how to approximate the function f that maps $X_{[t-k+1,t]}$ given Θ to the output a_{t+1} subject to the lower bound, emergency braking acceleration a_{LB}, and the upper bound, maximum possible acceleration a_{UB}. In previous CFMs, X_t is represented by a feature vector of fixed size limiting the models' capability to learn richer spatial information.

To overcome this limitation, GNNs were used in [11] to enable the model to process inputs with a graph structure that consists of a variable number of vehicles at each time step. Accordingly, Equation 1 can be revised as follows to include *spatial information processing (SIP)* function f_S for X_i.

$$a_{t+1} = f_T\Big(f_S(X_{t-k+1}|\theta_S), ..., f_S(X_t|\theta_S)\Big|\theta_T\Big) \quad (2)$$

where f_T is the *temporal information processing (TIP)* function that aggregates information across different time steps to model temporal dependency among past vehicle states. Finally, it produces CF acceleration based on this aggregated information. θ_S and θ_T are parameters for f_S and f_T, respectively; $\Theta = \{\theta_S, \theta_T\}$ defines the set of all learnable model parameters. Our goal here is to model f_S with GCN for *SIP* while GRU is used to model f_T for *TIP*. Future works may consider other alternative architectures if they can improve the model performance.

Finally, an appropriate choice of numerical integration schemes [25] can be used to derive the next vehicle position x_{t+1} and velocity v_{t+1}. Similar to [11], the Euler update is adopted in this paper for its simplicity and efficient computation, and the step size Δt is set to 1 second.

$$v_{t+1} = v_t + a_{t+1}\Delta t; \;\; v_{t+1} \geq 0 \quad (3)$$

$$x_{t+1} = x_t + v_{t+1}\Delta t; \;\; \Delta t = 1 \quad (4)$$

B. Spatial Information Processing (SIP)

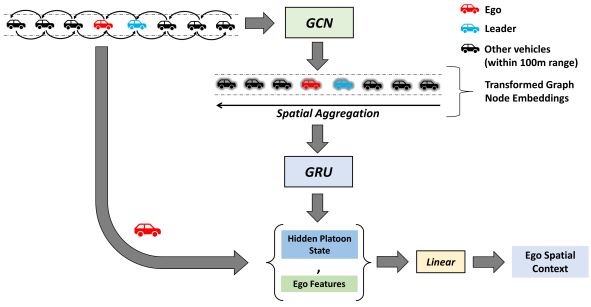

Fig. 1. Spatial information processing using GCN and GRU

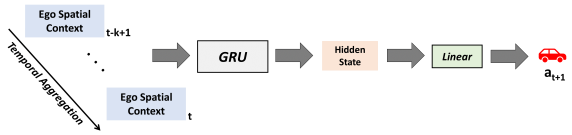

Fig. 2. Temporal information processing using GRU

In this paper, $f_S(X_t|\theta_S)$ is designed as shown in Figure 1. Each of the components is explained below.

1) Input Graph: Unlike the input graph used in [11] which considers surrounding vehicles in all directions, we specifically consider a *vehicle platoon graph (VPG)* consisting of vehicles traveling along the same lane as the ego vehicle (with the red color in the figure) within a certain sensing range (defined as 100m in both front and rear directions). We hypothesize that when considering driving situations solely for longitudinal acceleration without any external influences coming from either cut-in, overtaking, or lane-changing scenarios, more accurate CF behaviors can be captured by only considering a vehicle platoon involving the ego vehicle at each time step. Each of the vehicles in VPG is represented by a feature vector F as follows:

$$F = [v, s, \Delta v] \quad (5)$$

where $v, s, \Delta v$ are velocity, gap distance and relative velocity (w.r.t corresponding leaders), respectively. Thus, our input feature matrix X_t contains all the feature vectors of vehicles (nodes) in the VPG at time t. As for edge connections, a given vehicle is connected to its immediate following and preceding vehicles only. This connection information is encoded in the adjacency matrix A.

2) Graph Learning: As for graph learning to extract interrelated patterns among vehicles, GCN [26] is adopted since its modified version has achieved the best performance among CFMs compared in [11]. The GCN layer used in this paper is defined as follows.

$$H^{(l+1)} = \sigma\left(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \quad (6)$$

where $H^{(l+1)}$ is node features at the $(l+1)$-th layer of the GCN; σ is activation function; A is adjacency matrix without self-connections; $D_{ii} = \sum_j A_{ij}$ is its diagonal degree matrix; $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the symmetric normalization of A (refer to [26]); $H^{(l)}$ is node features at the l-th layer of the GCN; $W^{(l)}$ is weight matrix of the l-th layer. Initial H^0 is equivalent to X. Self-connections are removed for better performance as suggested in existing works [10], [11].

After passing through the GCN layer, the nodes in VPG have been transformed into their respective embeddings encoded in H. Usually, the nodes need not be ordered. However, in this paper, inspired by *shock-wave-propagation theory* [27], the nodes are ordered with the first vehicle in the queried platoon taking the first index position, and the last taking the final index. Since traffic shock waves travel through a series of vehicles as a result of changing circumstances at the forefront of the platoon, a GRU layer is used to process this sequence of vehicle embeddings in their spatial order as illustrated in Figure 1. Since there can be different numbers of nodes in each VPG, a specific GRU aggregation mechanism is required to handle the variable sequence. Hence, we have adopted the adaptive GRU aggregation (a.k.a. readout function) proposed in a recent work [28] to achieve this purpose. Next, the *hidden platoon state* produced by the GRU layer is concatenated with the ego vehicle's original features in order to recover its own information signal that might be lost during the graph learning process. Eventually, the concatenated vector is processed through a linear layer to produce the final spatial context vector C_t of the ego vehicle.

Accordingly, the final output of our *SIP* function modeled with GCN can be defined as:

$$C_t = f_S(X_t|\theta_S) \quad (7)$$

Due to the space constraint, more complete descriptions of layer information are provided in our supplementary materials [29].

C. Temporal Information Processing (TIP)

The spatial context vectors C_t produced by $f_S(.|\theta_S)$ at each time step can be aggregated to combine all spatial information within the past window length of k. This is done by processing through another GRU layer. The hidden state vector produced by this GRU layer is further passed through a linear layer whose output is eventually activated by $tanh$ function to limit the output value within $[-1, 1]$. The output is then scaled based on the range $[a_{LB}, a_{UB}]$ to produce the final longitudinal acceleration a_{t+1}. This entire process can be defined as:

$$a_{t+1} = f_T(C_{t-k+1}, C_{t-k}, ..., C_t|\theta_T) \tag{8}$$

Due to the importance of long memory [18], k is set to 10 in this paper, hence modeling the driver memory effect that lasts 10 seconds.

D. Physics-guided Machine Learning (PGML)

In CFM literature [14], [15], one commonly adopted practice of PGML is the usage of a physics-guided loss function that combines: 1) *data loss* – the discrepancy between the model prediction and the target data, and 2) *physics loss* – the discrepancy between the model prediction and a reference physics-based model output. Thus, the model is not only trained by minimizing its prediction error w.r.t the data but also regularized to prevent its prediction from deviating too much from a physically consistent output. Unlike existing works, two additional venues where the knowledge of the physics-based models can be injected are also proposed in Sections III-D.3 and III-D.4.

1) Physics-based CFM: With reference to previous works [14], [15], *Intelligent Driver Model (IDM)* is adopted in this paper as our ideal physics-based model. The IDM can be described by the following equations.

$$a_{t+1}^{phy} = a_{max}\left[1 - \left(\frac{v_t}{v_0}\right)^4 - \left(\frac{s^*(v_t, \Delta v_t)}{s_t}\right)^2\right], \tag{9}$$

$$\text{where} \quad s^*(.) = s_0 + \max\left(0, v_t\,T + \frac{v_t\,\Delta v_t}{2\sqrt{a_{max}\,b_{max}}}\right)$$

It produces the longitudinal acceleration a_{t+1}^{phy} of a follower for the next time step depending on the inputs $v_t, s_t, \Delta v_t$ previously defined in Equation 5. It also has constant parameters (each having its own physical interpretation) such as maximum desired velocity v_0, jam spacing distance s_0, desired time headway T, maximum desired acceleration and deceleration a_{max} and b_{max}, respectively. $s^*(.)$ is the optimal safety gap which depends on v_t and Δv_t.

2) Physics-guided Loss Function: The physics-guided loss function used for model training is defined as:

$$\mathcal{L}(\Theta) = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\left[a_i^{data} - a_i\right]^2 + \lambda\left[a_i^{phy} - a_i\right]^2 \tag{10}$$

where a_i and a_i^{phy} are defined in Equations 8 and 9, respectively. $\lambda \in [0, 1]$ is the configurable parameter to control the contribution of the physics loss towards the total loss. Given training samples $\mathcal{P}$, the loss is calculated as the mean of the sum of squared error losses involving – 1) *data loss* measuring the difference between ground-truth a_i^{data} and our model acceleration a_i, and 2) *physics loss* measuring the difference between ideal physics-based model acceleration a_i^{phy} and a_i.

This *physics loss* can be viewed as a regularization term to prevent the model from overfitting to observed data. This could especially be helpful in situations where observed data would have limited coverage, both safe and unsafe driving maneuvers, or any other biases or errors that could be overlooked during data pre-processing. On a deeper level, since this physics loss plays a part in the gradient of the loss function w.r.t Θ, this can also be viewed as injecting physical consistency into the model parameter space via the gradient.

3) Physics-guided Features: When using *physics loss*, the physical consistency is only enforced via the model parameter space as explained above. In order to inject physical consistency into the model input space as well, two more physics-based features are added to the initial feature vector as follows.

$$F_{phy} = [v, s, \Delta v, v_{phy}, a_{phy}] \tag{11}$$

where v_{phy} and a_{phy} are velocity and acceleration derived from IDM, respectively.

By incorporating these two features, the model is now guided by physics to enforce physical consistency via this space. It can also be viewed as our data-driven model taking *reference model dynamics* into consideration when trying to learn the mapping function f instead of neglecting these dynamics and learning on its own.

4) Physics-guided Edges: The above addition of physics-guided features only affects the initial node representation in VPG, but the adjacency matrix still takes the form of a (0, 1) binary matrix. Since both input feature matrix X and adjacency matrix A are required for the GCN learning process, we can also design a weighted adjacency matrix where the derivation of the weights is physics-guided so that physical consistency can be enforced once again via edge connections. It is derived as follows.

The IDM equation can be interpreted as the interpolation of two terms [4]: 1) *free-road term* $a_{max}\left[1 - (\frac{v_t}{v_0})^4\right]$ modeling the tendency to accelerate, and 2) *interaction term* $-a_{max}\left[\frac{s^*(.)}{s_t}\right]^2$ modeling the tendency to brake. The latter is borrowed to define physics-guided edge weights.

$$e_{i,j} = \frac{\min\left(a_{max}\left[\frac{s^*(.)}{s_t}\right]^2, a_{UB}\right)}{a_{UB}} \tag{12}$$

$$e_{i,j} = e_{j,i}\ ; \quad e_{i,j} \in [0, 1] \tag{13}$$

For a pair of follower i and leader j, the edge weight that models their interaction can be quantified based on how much braking acceleration follower i is required to apply to maintain optimal safe gap distance s^* from its leader j. When the current gap distance s_t is considerably lower than

s^* (meaning unsafe situation), the braking acceleration will be high, resulting in $e_{i,j}$ close to 1, and vice versa.

5) Learning with Physics-guided Edge Weights: To account for physics-guided $e_{i,j}$, the layer-wise propagation rule defined in Equation 6 needs to be revised as follows.

$$H^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{14}$$

where $\hat{A}$ is the weighted adjacency matrix without self-connections with $\hat{A}_{i,j} = e_{i,j}$; $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ is its (weighted) diagonal degree matrix; $\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$ is the symmetric normalization of $\hat{A}$ (refer to [26]).

E. Training Physics-guided Data-driven CFM

The training algorithm for our physics-guided GCN-based CFM is described in Algorithm 1. In the beginning, the training constants, model parameters, and hyperparameters are initialized as described. The model weight parameters (θ_S for GCN and θ_T for GRU respectively) are jointly trained in this algorithm. It will be interesting for future works to consider spatial graph representation learning and downstream prediction tasks separately by optimizing based on different objectives. Over a number of E training epochs, the model parameters are updated in a batch-by-batch manner during each epoch. Each batch of training data contains B number of trajectories, and from each trajectory, valid input-output sample pairs are retrieved. After processing the input data through the GCN and GRU layers, the final output acceleration for an ego vehicle is obtained. With these model outputs and ground-truth data, the model parameters are then updated using the gradient of the physics-guided loss. For more robust validation, CF simulation-based validation is adopted here since it can offer better model generalization than conventional one-step prediction validation. This is because the former can measure the model error accumulation throughout the entire simulation period whereas the latter can merely account for the model error for the transition between two consecutive steps (t and $t+1$). The line-by-line description and explanation of the algorithm are also given in our supplementary materials [29].

Algorithm 1: Model Training Algorithm

Result: CFM f with trained parameters Θ;
Initialization:
Initialize training and validation data D_{tr} and D_{val}, respectively, containing $X, \hat{A}, a_{data}, a_{phy}$;
Initialize hyperparameters for f;
Initialize network weights $\theta_S, \theta_T \in \Theta$ randomly;
Initialize a lower and upper bounds: $[a_{LB}, a_{UB}]$;
Initialize maximum batch count $B_{max} = \lfloor \frac{|D_{tr}|}{B} \rfloor$;
for *epoch = 1, ..., E* **do**
  Shuffle trajectories ζ in D_{tr};
  for *batch = 1, ..., B_{max}* **do**
    Get B trajectories from D_{tr};
    for ζ_1, ..., ζ_B **do**
      Get all valid training input and output sample pairs $\mathcal{P}$ from ζ_j;
      Get $X_{\mathcal{P}}, \hat{A}_{\mathcal{P}}, a_{\mathcal{P}}^{data}, a_{\mathcal{P}}^{phy}$ from $\mathcal{P}$;
      $C_{\mathcal{P}} \leftarrow GCN(X_{\mathcal{P}}, \hat{A}_{\mathcal{P}}|\theta_S)$ where $C_{\mathcal{P}} = [C_{t-k+1}, ..., C_t]_{\mathcal{P}}$;
      $a_{\mathcal{P}} \leftarrow GRU(C_{\mathcal{P}}|\theta_T)$;
      $\Theta \leftarrow \Theta - \alpha\nabla_{\Theta}\mathcal{L}(\Theta)$;
    end
  end
  Perform simulation-based validation using D_{val};
  Keep $f(\Theta)$ with the lowest validation error;
end

IV. Experiments

A. Experiment Setup

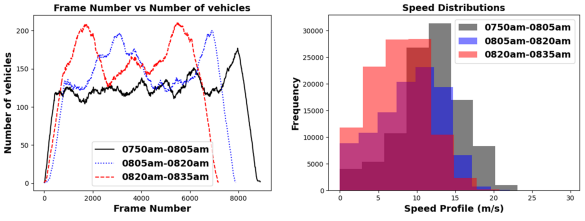

Fig. 3. Vehicle density (left), (figure extracted from [15]), and speed distributions (right) for each data collection period in the NGSIM dataset

A real-world trajectory dataset, namely Next Generation Simulation (NGSIM) US Highway 101 [30], was used to conduct our experiments. The vehicle density and speed profiles across three different periods in the collected data are displayed in Figure 3. The trajectory data from all five major lanes were used for experiments. Detailed data pre-processing steps involving data sampling/smoothing/filtering processes, can be found in our previous work [31] (and supplementary materials [29]). Some extra data pre-processing steps were also carried out for training various GCN-based models.

1) Simulation Setup: Only pure car-following (CF) scenario was considered for our experiments, i.e., no cut-in, over-taking, or lane-changing events for a given follower-leader pair. However, vehicles other than our follower-leader pair of interest were allowed to perform those actions. In our CF simulation experiments, all vehicles' movements other than the ego vehicle were determined by the actual ground-truth data while the ego vehicle's movement was controlled by different models. It should be noted that during simulation run-time, the relative features (such as relative velocity or gaps) of all the immediate neighbors of the ego vehicle should be properly updated based on its simulated information. The simulation step size was chosen as 1 second. During validation and testing, the first 10-second ground-truth trajectory was used as model inputs and vehicle state initialization. Afterward, the ego vehicle was simulated as mentioned above. In case of a collision event, the trajectory simulation was terminated.

2) Trajectory Data Information: To account for a minimum of 15-second CF period, follower-leader pairs with a trajectory length of at least 25 seconds were used. To prevent input data from having truncated graphs, the vehicle positions, which are at least 100m away from the beginning and the end of the study area (about 670m), were used.

Finally, a total of 719 trajectories were pre-processed from *Dataset 1* (07:50-08:05 AM); 703 from *Dataset 2* (08:05-08:20 AM), and 791 from *Dataset 3* (08:20-08:35 AM), respectively.

3) In-distribution Training & Testing: A total of 719 Trajectories from *Dataset 1* were further divided as follows – 518 (70%) for model training, 57 (10%) for validation, and 144 (20%) for testing. Using these training trajectories, the physics-based IDM was also calibrated with Genetic Algorithm [32] to be used for physics-guided learning of our proposed model. In this paper, all the test experiment results were obtained by taking the average of the results after training and testing five times with different random seeds.

4) Out-of-distribution Testing: All trajectories from *Dataset 2* and *3* were used to test model performance under distributional shift. Their density and speed profiles have already been shifted noticeably as seen in Figure 3.

5) Performance Evaluation Metrics: Two types of performance measures are considered in this paper – 1) front gap error to assess CF simulation accuracy, and 2) collision occurrences to assess safety.

- $CPGE$: The *"Collision-penalized Gap Root Mean Squared Error (CPGE)"* is defined as:

$$\sqrt{\frac{1}{V}\sum_{j=1}^{V}\Big[\Big(\frac{1}{L_j}\sum_{t=1}^{L_j}(s_{j,t}^{data}-s_{j,t})^2\Big)+\mathbb{1}_{[L_j<N_j]}\gamma P_j\Big]} \tag{15}$$

  where V is the total number of test vehicle trajectories. L_j and N_j are the simulated and actual trajectory lengths, respectively. s^{data} and s are ground-truth and simulated front gaps, correspondingly. P is the collision penalty and γ (≥ 0) is its factor (set to 2.5 in this paper; using a very large factor value is not recommended since it can overwhelm the other error term making the metric interpretation difficult). For a *j-th* trajectory, the penalty is defined as $P_j = x_{N_j}^{data} - x_{L_j}$, measuring the left-over distance from the point of collision x_{L_j} at terminated time L_j to the actual final position $x_{N_j}^{data}$ at the actual final time N_j. $\mathbb{1}$ is the indicator function evaluated to 1 if the constraint is satisfied, and 0 otherwise. Hence, no penalty will be added if $L_j = N_j$ meaning a complete simulation run.
- FC & RC: *Front* and *rear collisions* are calculated as:

$$FC = \sum_{j=1}^{V}\mathbb{1}_{[s_{i,l}\leq 0]_j};\ RC = \sum_{j=1}^{V}\mathbb{1}_{[s_{i,r}\leq 0]_j} \tag{16}$$

  where for a *j-th* trajectory, $s_{i,l}$ is the simulated front gap between an ego vehicle i and its leader l; $s_{i,r}$ is the simulated rear gap between an ego vehicle i and its rear vehicle r; the indices i, l, r belong to the set of all node indices in a given VPG at a particular time t. FC and RC calculate the total number of their respective collision events for all test trajectories V.

$CPGE$ is used to measure the model performance in terms of CF accuracy while accounting for collision penalty. This is more objective than simply measuring the usual RMSE because if the simulation has terminated due to collision events, it can result in fewer data points being used for the calculation of RMSE which could lead to the wrong interpretation of lower error.

FC is more important than RC based on our experiment setup because FC indeed reflects a model's ability to produce safe acceleration to avoid collisions with its leader. As for RC, it is included to follow the practice of our reference work [11]. RC cannot be interpreted literally since the rear vehicle behind an ego's vehicle is being controlled by the actual data neglecting any response made by the simulated ego. This will not happen in a proper simulation where all vehicles are controlled by the model. Nevertheless, it could reflect the number of cases in which the ego vehicle suddenly brakes harder than necessary (e.g., overreacting to its leader's responses) resulting in a collision with its rear vehicle that could not have stopped in time.

B. Car-following Models for Comparison

The following CFMs were considered for performance comparison. All the model outputs are longitudinal acceleration for an ego vehicle at each simulation step.

1) IDM: is uncalibrated and configured with default parameters reported in [33].

2) GA-IDM: is the IDM calibrated with Genetic Algorithm (GA) [32] using training trajectories. The fitness scores were calculated based on the CF simulation-based validation described in Section III-E.

3) FFN: is the simple feed-forward neural network using a total of 11 input features – taking 10 features used in [11] including ego vehicle information and gap distances towards its surrounding neighbors (vehicle length is used on behalf of vehicle class), and relative velocity (w.r.t to a leader) as an extra feature.

4) GCN: is the graph convolutional network that takes graph data at each time step as inputs. Each of the vehicles in the graph is represented by the same 11 features above. As in [11], it considers neighbors from adjacent lanes when building the graph with edge connections linked to immediate surrounding neighbors. Global mean pooling was used as an aggregation function to combine node embeddings. Ego's features are then concatenated to the aggregated context vector.

5) GCN-GRU: consists of a similar GCN layer as above but produces spatial context vector C_t at each time step (instead of acceleration) and is combined with GRU to perform aggregation of spatial context vectors from time $t-k+1$ to t. The model design is very similar to *EGCN-LSTM* used in [11].

6) GCN-GRU++: is our variation of GCN-GRU using VPG and shock-wave-propagation inspired aggregation as described in Sections III.B and III.C.

7) PG-GCN-GRU: is our newly proposed *"Physics-guided Graph Convolutional Network with Gated Recurrent Unit"* that enhances GCN-GRU++ with PGML as described in Section III-D.

TABLE I

MODEL PERFORMANCE COMPARISON USING BOTH IN-DISTRIBUTION AND OUT-OF-DISTRIBUTION DATASETS

	Dataset 1 (In-distribution)				*Dataset 2 (Out-of-distribution)*				*Dataset 3 (Out-of-distribution)*			
Model	***CPGE***	***FC***	***RC***	***Collisions (%)***	***CPGE***	***FC***	***RC***	***Collisions (%)***	***CPGE***	***FC***	***RC***	***Collisions (%)***
IDM	11.7626	**0**	24	16.67	10.8414	**0**	116	16.5	11.9689	**0**	192	24.27
GA-IDM	8.3535	**0**	8.2	5.69	7.2236	**0**	22.2	3.16	8.5135	**0**	72.4	9.15
FFN	8.5861	4.8	10.6	10.69	13.1275	229	15.4	34.77	15.0101	429.2	26.6	57.62
GCN	8.2685	5.8	8	9.58	13.542	199.8	64.2	37.55	15.5516	393.8	94	61.67
GCN-GRU	7.7299	5	**7.6**	8.75	13.535	205.8	68.2	38.98	15.399	362.2	133.6	62.68
GCN-GRU++	**6.5435**	**0**	**7.6**	**5.28**	9.5659	22.6	103.8	17.98	10.2215	39.2	178.6	27.53
PG-GCN-GRU	7.4034	**0**	8.6	5.97	**6.2095**	**0**	18.6	**2.65**	**7.5386**	**0**	64.4	**8.14**

C. In-distribution Performance Assessment

In this section, models were both trained and tested by using trajectories from *Dataset 1* where the traffic situation is the least congested among all three datasets (see Figure 3). The longest CF duration in this dataset is 55 seconds.

In Table I, uncalibrated IDM has achieved the worst performance out of all models. Despite being left uncalibrated, it still maintains its front collision-free property. However, due to its strong braking nature [34], a relatively high number of RC is observed. When it is properly calibrated with GA, it has achieved performance as competitive as its data-driven counterparts such as FFN and GCN while still being collision-free in the front direction. In fact, it has actually outperformed FFN when evaluating with *CPGE*. The underperformance of FFN is due to the objective measure of *CPGE* that not only accounts for model accuracy but also safety by penalizing collision events. Although FFN might be more accurate in some cases, its CF behaviors were no longer safe when evaluated under long CF duration.

A *relative error decrease* (RED) of 3.7% is observed when comparing GCN against FFN. This shows the usefulness of graph input data for modeling CFM. However, when compared to GA-IDM, it has shown only slight performance improvement. When integrated with the GRU structure, a noticeable performance improvement of 7.46% can be observed when comparing the data-driven GCN-GRU against physics-based GA-IDM. This can be attributed to: 1) the higher model complexity of GCN-GRU; and 2) its incorporation of driver memory effect by aggregating temporal information across the past 10-second trajectory.

The best performance across all metrics is achieved by our variant GCN-GRU++ outperforming physics-based GA-IDM by about 21.67% and its predecessor GCN-GRU by about 15.35%. This demonstrates the efficacy of VPG built with simple but effective features in modeling CF behaviors and questions the impact of surrounding vehicle information in pure CF situations in alignment with an existing study [35]. Both GCN-GRU and GCN-GRU++ have achieved the lowest RC count which may reflect their less overreaction to a leader's sudden braking than IDM-based models.

One important observation is that none of the pure data-driven models except our variant GCN-GRU++ are actually collision-free with respect to leading vehicles. Lastly, our physics-guided PG-GCN-GRU achieves comparable performance as GCN-GRU++ and also retains the (front) collision-free property. In terms of accuracy, it still surpasses GA-IDM by a RED of about 11.37% while resulting in the second-highest accuracy among all data-driven models.

D. Out-of-distribution Performance Assessment

In this section, all models were tested with trajectories that were subject to distributional shift. They were not trained by any trajectories from *Dataset 2* or *3*. Compared to *Dataset 1* (model training data), more congested traffic can be observed in *Dataset 2*, and the most congested traffic in *Dataset 3*. The longest CF duration in *Dataset 2* and *3* are 83 and 78 seconds, respectively.

As evidenced in Table I, all pure data-driven models' performances have degraded severely under distributional shift. Their collision percentages range from 17.98% to 38.98% when testing with *Dataset 2* and from 27.53% to 62.68% with *Dataset 3*. Due to these very high collision rates, their results are no longer meaningful for analysis. Especially, high FC rate is undesirable in a CFM as its sole purpose is to model CF behaviors in order to avoid colliding with a leading vehicle.

Under distributional shift, our proposed model, PG-GCN-GRU, has surpassed all the models in terms of both accuracy and safety. Only physics-based models and PG-GCN-GRU did not encounter any FC under the shift. Our proposed model has outperformed GA-IDM by a RED of 14.04% with *Dataset 2* and 11.45% with *Dataset 3* while achieving the lowest collision rates in both datasets. Thus, these results clearly indicate the advantage of PGML when incorporated into a data-driven CFM. It is not only able to learn accurate behavioral patterns from data but also regulate its parameters to produce physically consistent outputs under both in-distribution and out-of-distribution scenarios.

We have also conducted an ablation study. Due to the space constraint, the analysis is given towards the results obtained based on *Dataset 3* only (which exhibits the most severe distributional shift). The physics-guided loss turns out to be the most critical component among the three PGML-based modeling aspects introduced in Section III-D. When this loss was excluded from the training process, the proposed model was no longer front-collision-free and experienced about 49 FC increasing the *CPGE* to 10.3523. More analysis of the ablation study and the results are provided in [29].

V. CONCLUSIONS

In this paper, a new physics-guided GCN-based CFM integrated with the GRU structure is proposed. The model is extended from our improved GCN-GRU-based CFM which incorporates a vehicle platoon graph along with other refinements. Under both in-distribution and out-of-distribution scenarios, the experiment results have shown the effectiveness of the proposed model in producing both safe and accurate car-following behaviors. Under the most severe distributional shift, the model has outperformed (in terms of CPGE) the best pure data-driven model by about 26.25%, and the well-calibrated physics-based model, by about 11.45%. Future works can consider improving the robustness of our proposed model under a variety of distributional shifts (e.g., data collected in different study areas) by building upon the insights and suggestions provided in the paper.

ACKNOWLEDGMENT

This work was supported by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI), Nanyang Technological University, Singapore.

REFERENCES

[1] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," in *Computer Graphics Forum*, vol. 39, no. 1. Wiley Online Library, 2020, pp. 287–308.

[2] L. Li, R. Jiang, Z. He, X. M. Chen, and X. Zhou, "Trajectory data-based traffic flow studies: A revisit," *Transportation Research Part C: Emerging Technologies*, vol. 114, pp. 225–240, 2020.

[3] H. Zhou, J. Laval, A. Zhou, Y. Wang, W. Wu, Z. Qing, and S. Peeta, "Review of learning-based longitudinal motion planning for autonomous vehicles: research gaps between self-driving and traffic congestion," *Transportation research record*, vol. 2676, no. 1, pp. 324–341, 2022.

[4] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[5] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[6] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," 1998.

[7] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, "Capturing car-following behaviors by deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 910–920, 2017.

[8] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation research part C: emerging technologies*, vol. 97, pp. 348–368, 2018.

[9] H. Naing, W. Cai, N. Hu, T. Wu, and L. Yu, "Data-driven microscopic traffic modelling and simulation using dynamic lstm," in *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2021, pp. 1–12.

[10] F. Diehl, T. Brunner, M. T. Le, and A. Knoll, "Graph neural networks for modelling traffic participant interaction," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 695–701.

[11] J. Su, P. A. Beling, R. Guo, and K. Han, "Graph convolution networks for probabilistic modeling of driving acceleration," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–8.

[12] A. Malinin, N. Band, G. Chesnokov, Y. Gal, M. J. Gales, A. Noskov, A. Ploskonosov, L. Prokhorenkova, I. Provilkov, V. Raina *et al.*, "Shifts: A dataset of real distributional shift across multiple large-scale tasks," *arXiv preprint arXiv:2107.07455*, 2021.

[13] R. Huang, A. Geng, and Y. Li, "On the importance of gradients for detecting distributional shifts in the wild," *Advances in Neural Information Processing Systems*, vol. 34, pp. 677–689, 2021.

[14] Z. Mo, R. Shi, and X. Di, "A physics-informed deep learning paradigm for car-following models," *Transportation research part C: emerging technologies*, vol. 130, p. 103240, 2021.

[15] H. Naing, W. Cai, H. Nan, W. Tiantian, and Y. Liang, "Dynamic data-driven microscopic traffic simulation using jointly trained physics-guided long short-term memory," *ACM Transactions on Modeling and Computer Simulation*, vol. 32, no. 4, pp. 1–27, 2022.

[16] D. Li and O. Okhrin, "Modified ddpg car-following model with a real-world human driving experience with carla simulator," *Transportation Research Part C: Emerging Technologies*, vol. 147, p. 103987, 2023.

[17] F. Hart, O. Okhrin, and M. Treiber, "Formulation and validation of a car-following model based on deep reinforcement learning," *arXiv preprint arXiv:2109.14268*, 2021.

[18] X. Wang, R. Jiang, L. Li, Y.-L. Lin, and F.-Y. Wang, "Long memory is important: A test study on deep-learning based car-following model," *Physica A: Statistical Mechanics and its Applications*, vol. 514, pp. 786–795, 2019.

[19] V. Papathanasopoulou and C. Antoniou, "Towards data-driven car-following models," *Transportation Research Part C: Emerging Technologies*, vol. 55, pp. 496–509, 2015.

[20] D. Yang, L. Zhu, Y. Liu, D. Wu, and B. Ran, "A novel car-following control model combining machine learning and kinematics models for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 1991–2000, 2018.

[21] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[22] A. Sharma, Z. Zheng, and A. Bhaskar, "Is more always better? the impact of vehicular trajectory completeness on car-following model calibration and validation," *Transportation research part B: methodological*, vol. 120, pp. 49–75, 2019.

[23] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating physics-based modeling with machine learning: A survey," *arXiv preprint arXiv:2003.04919*, vol. 1, no. 1, pp. 1–34, 2020.

[24] C. Peng, F. Xia, V. Saikrishna, and H. Liu, "Physics-informed graph learning: A survey," *arXiv preprint arXiv:2202.10679*, 2022.

[25] M. Treiber and V. Kanagaraj, "Comparing numerical integration schemes for time-continuous car-following models," *Physica A: Statistical Mechanics and its Applications*, vol. 419, pp. 183–195, 2015.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[27] A. Ibrahim, M. Čičić, D. Goswami, T. Basten, and K. H. Johansson, "Control of platooned vehicles in presence of traffic shock waves," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1727–1734.

[28] D. Buterez, J. P. Janet, S. J. Kiddle, D. Oglic, and P. Liò, "Graph neural networks with adaptive readouts," *arXiv preprint arXiv:2211.04952*, 2022.

[29] H. Naing. (2023) Supplementary materials: Physics-guided graph convolutional network for modeling car-following behaviors under distributional shift. [Online]. Available: https://github.com/Javelin1991/PhyGuidedGraphConvCFM

[30] J. Colyar and J. Halkias, "Us highway 101 dataset," Federal Highway Admin. (FHWA), Washington, DC, USA, Tech. Rep. FHWA-HRT-07-030, 2007.

[31] H. Naing, W. Cai, T. Wu, and L. Yu, "Dynamic car-following model calibration with deep reinforcement learning," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 959–966.

[32] A. F. Gad, "Pygad: An intuitive genetic algorithm python library," 2021.

[33] V. Punzo, M. Montanino, and B. Ciuffo, "Do we really need to calibrate all the parameters? variance-based sensitivity analysis to simplify microscopic traffic flow models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 184–193, 2014.

[34] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4585–4605, 2010.

[35] S. Vasebi, Y. M. Hayeri, and P. J. Jin, "Surrounding vehicles' contribution to car-following models: Deep-learning-based analysis," *Transportation research record*, vol. 2675, no. 11, pp. 623–640, 2021.