Dynamic Data-driven Microscopic Traffic Simulation using Jointly Trained Physics-guided Long Short-term Memory

HTET NAING, Nanyang Technological University, Singapore
WENTONG CAI, Nanyang Technological University, Singapore
HU NAN, Alibaba Cloud, Singapore
WU TIANTIAN, Alibaba Cloud, China
YU LIANG, Alibaba Cloud, China

Symbiotic simulation systems that incorporate data-driven methods (such as machine/deep learning) are effective and efficient tools for just-in-time (JIT) operational decision making. With the growing interest on Digital Twin City, such systems are ideal for real-time microscopic traffic simulation. However, learning-based models are heavily biased towards the training data and could produce physically inconsistent outputs. In terms of microscopic traffic simulation, this could lead to unsafe driving behaviours causing vehicle collisions in the simulation. As for symbiotic simulation, this could severely affect the performance of real-time base simulation model resulting in inaccurate or unrealistic forecasts, which could, in turn, mislead JIT what-if analysis. To overcome this issue, a physics-guided data-driven modelling paradigm should be adopted so that the resulting model could capture both accurate and safe driving behaviours. However, very few works exist in the development of such a car-following model that can balance between simulation accuracy and physical consistency. Therefore, in this paper, a new "jointly-trained physics-guided Long Short-term Memory (JTPG-LSTM)" neural network, is proposed and integrated to a dynamic data-driven simulation system to capture dynamic car-following behaviours. An extensive set of experiments was conducted to demonstrate the advantages of the proposed model from both modelling and simulation perspectives.

CCS Concepts: • **Computing methodologies** → **Modeling methodologies**; **Real-time simulation**; **Agent / discrete models**; **Neural networks**.

Additional Key Words and Phrases: Data-driven modelling; car-following; physics-guided machine learning; online learning; just-in-time simulation; real-time simulation; symbiotic simulation; digital twin

ACM Reference Format:
Htet Naing, Wentong Cai, Hu Nan, Wu Tiantian, and Yu Liang. 2021. Dynamic Data-driven Microscopic Traffic Simulation using Jointly Trained Physics-guided Long Short-term Memory. *J. ACM* 1, 1 (July 2021), 27 pages.

1 INTRODUCTION

Symbiotic simulation systems are effective and efficient tools for just-in-time (JIT) short-term forecasting unlike conventional simulation approaches which are only suitable for medium-term or long-term forecasting. A typical symbiotic simulation system comprises a "base" or "reference" simulation [3, 23], which is essentially a Digital Twin, running in synchronization with a physical system. Furthermore, it incorporates real-time (or near real-time) sensor measurement data for updating

Authors' addresses: Htet Naing, htetnain001@e.ntu.edu.sg, Nanyang Technological University, Singapore; Wentong Cai, aswtcai@ntu.edu.sg, Nanyang Technological University, Singapore; Hu Nan, nan.h@alibaba-inc.com, Alibaba Cloud, Singapore; Wu Tiantian, qiutian.wtt@alibaba-inc.com, Alibaba Cloud, China; Yu Liang, liangyu.yl@alibaba-inc.com, Alibaba Cloud, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2021 Association for Computing Machinery.
0004-5411/2021/7-ART $15.00
https://doi.org/

important model parameters regularly. This allows its users to readily instantiate multiple what-if simulation (WIS) branches from the base simulation so that alternative scenarios could be analyzed quickly to assist JIT operational decision making. They have been successfully adopted in many different real-world applications of various industries ranging from manufacturing, transportation, information technology and healthcare [31].

Recent symbiotic simulation systems have made use of machine learning to further improve their accuracy for better prescriptive analytics [31]. In this context, machine learning (ML) is usually used to model complex but important components of a simulation system. Unlike mathematical or physics-based models, these data-driven ML models are not only capable of learning from real-world data, but also more flexible to incorporate additional input features. Despite its success in other domains, there are very few works that utilize such a simulation system in microscopic traffic simulation. Moreover, the application of most existing data-driven microscopic simulation models – [5, 14, 15, 32, 33, 46, 47, 49, 53, 54] – is confined to either offline predictive or prescriptive analysis. In other words, their usage of real-world data is limited to offline model training and evaluation or scenario initialization. If a learning-based model is to be used as part of symbiotic simulation, it should be capable of learning from online data so that the base simulation can accurately represent the physical system. To address the above limitations, our previous work [28, 30] have provided a dynamic data-driven microscopic traffic simulation framework that involves a learning-based car-following model (Long Short-Term Memory, LSTM) whose parameters are updated dynamically with online traffic data. Our proposed framework has significantly improved the JIT short-term forecasting accuracy when compared against other baseline models.

One major shortcoming of learning-based models is that they are not robust enough to produce physically consistent outputs [48]. Moreover, their performance degrades considerably when encountering unseen scenarios that are not covered in the training dataset [16]. In terms of microscopic traffic simulation, these problems could lead to unsafe driving behaviours causing vehicle collisions in the simulation. As for symbiotic simulation, this could severely affect the performance of base simulation producing inaccurate or unrealistic forecasts, which could in turn degrade the performance of JIT simulation. To overcome this issue, a physics-guided data-driven modelling paradigm has to be adopted in such a way that the resulting model could capture both accurate and safe driving behaviours. The car-following models proposed in [27] is an initiative towards this research direction. Nevertheless, their proposed models do not account for safe driving yet since their final model outputs are still solely determined by learning-based models.

Inspired by [27], we have extended their work in this paper to develop a new car-following model that combines a physics-based model (Intelligent Driver Model, IDM) and a learning-based model (LSTM). The parameters of these two models are jointly learnt from real-world driving data while IDM provides guidance towards the LSTM's learning process so that the latter could be physically consistent in capturing accurate driving behaviours. Moreover, this jointly trained physics-guided model (JTPG-LSTM) is also integrated to our previously proposed data-driven simulation framework [30]. As a result, our base simulation model could dynamically learn from online data while being aware of the desirable balance between model accuracy and safe driving rather than naively learning all driving patterns (which might be realistic or unrealistic or both) from the data. Our main contributions are as follows:

(1) a new car-following model, JTPG-LSTM, namely "jointly trained physics-guided LSTM", is developed by using offline traffic data.
(2) the proposed model is integrated to a data-driven microscopic traffic simulation system to dynamically update its parameters with online traffic data for JIT forecasting.

(3) an extensive set of experiments was conducted to demonstrate the advantages of the proposed model from both modelling and simulation perspectives.

The rest of the paper is organized as follows: Section 2 discusses related works including our previous dynamic data-driven simulation framework and existing data-driven car-following models. In Section 3, we present our proposed JTPG-LSTM and how it is integrated to the dynamic simulation system with online learning. Our experiment results are analyzed and discussed in Section 4. Finally, Section 5 concludes the paper with future research recommendation.

2 RELATED WORKS

2.1 Dynamic Data-driven Microscopic Traffic Simulation Framework

The dynamic data-driven microscopic traffic simulation framework which was proposed in our previous work [30] is briefly described in this section. The underlying concept of the framework is based on the combination between symbiotic simulation and online machine (deep) learning. The proposed framework was applied in a case study featuring a pseudo real-time high-way traffic simulation. Three main stages are involved in the framework. In the first stage, a LSTM-based car-following model was developed by using offline data. Since this was an offline model, it was not suitable to be used in the base simulation of a symbiotic simulation system. The goal of the base simulation is to incorporate real-time sensor measurement data to update its components so that it can closely reflect its physical counterpart. Thus, in the second stage of the framework, the offline LSTM model was integrated to the base simulation and equipped with online learning so that it could update its parameters dynamically by using real-time traffic data. Finally, multiple JIT simulation branches could be readily triggered to perform what-if analysis for alternative scenarios. Such efficiency could not be achieved without the base simulation since numerous factors – data organization, data pre-processing, model calibration, etc. – could slow down the JIT simulation process.

One existing limitation of our proposed framework is the ambiguity in terms of how reliable the LSTM model output is. Since the dynamics of a simulated vehicle is produced by the LSTM model, this dynamics should be sufficiently safe to avoid any potential collision with its leader. To ensure such safe driving behaviours, the output velocity of the LSTM model was only used as preferred velocity and then, it would be revised accordingly if it was deemed unsafe according to the default car-following model of the simulator, namely Simulation of Urban Mobility (SUMO) [20, 22]. Hence, it served as a black-box safe driving mechanism for our LSTM model. To address this limitation, we aim to extend our previous model in this paper by deeply coupling with a safe physics-based car-following model (IDM) during both training and deployment phases. More details about this extension will be described in Section 3.2.

2.2 Data-driven Car-following Models

In microscopic traffic simulation, car-following models play an important role since they are key aspect of modelling individual driving behaviours along with lane-changing models. Traditional physics-based car-following models, such as Gipps [11], Krauss [20], IDM [40], etc., are often restrictive and difficult to incorporate new variables of interest. Unlike these models, data-driven car-following models are more flexible because it can incorporate more input features to capture accurate and realistic driving behaviours. Furthermore, recent data-driven models are based on machine learning or deep learning which can exploit and learn complex patterns from real-world datasets. Hence, in the literature, the terms "data-driven model" and "learning-based model" are sometimes used interchangeably. The existing works on data-driven car-following models can be categorized into – (i) classic machine learning models [33, 34, 52]; (ii) deep learning models

Table 1. Summary of existing data-driven car-following models

Model Type	Ref	Year	Models Used	Hybrid Model	Main Contribution
Classic Machine Learning	[33]	2015	Loess	No	To capture more accurate car-following behaviours by using more input variables
	[34]	2018	Loess	No	To extend the model in [33] for traffics practicing weak-lane discipline
	[52]	2019	Random Forest, ANN, Gipps	Yes	To avoid vehicle collisions when using data-driven models
Deep Learning	[46]	2018	GRU, LSTM	No	To incorporate temporal dependency and driver memory effect into the model
	[24]	2019	Encoder-Decoder LSTM	No	To handle both variable input and output lengths and consider reaction delay when applying the model output
	[39]	2020	GRU, Markov Chain	Yes	To bound the model predicted value within the reasonable range
Deep Reinforcement Learning	[56]	2018	DDPG	No	To achieve better generalisation capability than deep learning models
	[57]	2019	DDPG	No	To extend the model in [56] to capture more realistic driving behaviour
Physics-Informed Deep Learning	[27]	2021	IDM, OVM, ANN, LSTM	Yes	To systematically integrate physics-based models and deep learning models

[24, 39, 46]; (iii) deep reinforcement learning models [56, 57], and (iv) physics-informed deep learning models [27]. The research works under each category are thoroughly discussed in more detail in our supplementary materials [29].

In literature, there seems to be a dilemma between physics-based models and data-driven models. On one hand, although data-driven models have superior accuracy over conventional methods, they could show unsafe driving behaviours under unseen scenarios. Furthermore, they lack model interpretability or explainability due to their black-box nature. On the other hand, most traditional physics-based models [20, 40] could guarantee longitudinal collision-free property despite their inferior performance. They are usually developed based on traffic flow theory and have strong analytical explainability. In the last few years, there is a growing interest in combining machine learning and physics-based models together for tackling complex science and engineering problems. In a recent survey [48], very detailed review of this emerging new research area is provided with useful insights, featuring its successful applications in many domains such as climate science, material discovery, biological sciences, hydrology and so on. However, as highlighted in [27], the traffic flow modelling and simulation appears to be lagged behind this advancement. Only one recent work [38] that adopts such a hybrid model is found in macroscopic traffic flow modelling and estimation.

To capitalize on the advantages of both modelling approaches, a novel paradigm has been proposed in [27] to model car-following behaviours by combining them in a deeper manner. During the training process, a learning-based model (either LSTM or Artificial Neural Network, ANN) is trained by using two types of losses – (i) data loss and (ii) physics loss. The former measures the discrepancy between the learning-based model output and the observed data while the latter accounts for the discrepancy between the learning-based model output and the physics-based model output (either from IDM or Optimal Velocity Model, OVM [4]). This encourages the learning-based model to produce physically consistent outputs. Through experiments, their proposed physics-informed deep learning models have been shown to surpass other baselines that do not couple with

physics-based models, especially when only sparse observation data are used in model training. However, their model formulation does not account for safe driving yet since the final model outputs are still solely determined by learning-based models.

The summary of the existing literature is provided in Table 1. As seen in the "Hybrid Model" column of the Table 1, many existing works attempt to develop these data-driven models purely relying on data without considering conventional approaches. The notion of "Hybrid Model" here is different from the definition of "Hybrid Simulation" used in existing literature [10, 21, 31]. The former combines different modelling methods such as physics-based and learning-based models whereas the latter integrates various simulation paradigms such as agent-based simulation, system dynamics, discrete-event simulation, and so on. Only few research works have been conducted to uncover the potential of a hybrid model that combines both data-driven and physics-based models. Physics-informed deep learning models such as [27] are an initiative towards this new paradigm for the systematic development of such a hybrid model. Instead of relying exclusively on either one of the methods, it would be more advantageous to draw strengths from both approaches so that a more powerful hybrid car-following model could be developed. This serves as our motivation in this paper to extend their work and propose a new car-following model that combines a physics-based model (IDM) and a data-driven model (LSTM) for capturing both accurate and safe driving behaviours.

3 DYNAMIC DATA-DRIVEN MICROSCOPIC TRAFFIC MODELLING & SIMULATION

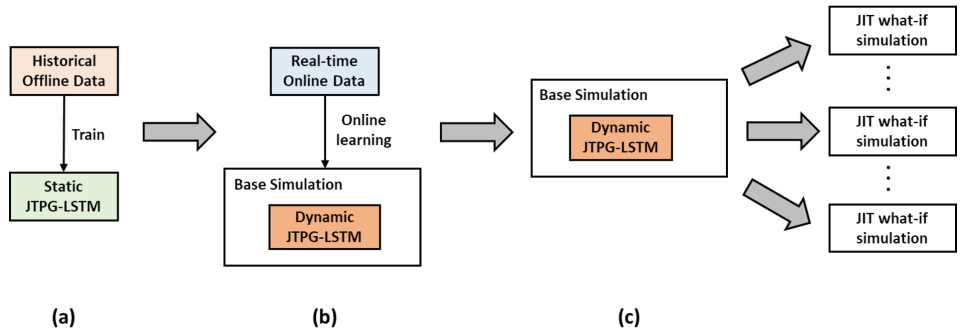

Fig. 1. Overview of dynamic data-driven microscopic traffic simulation framework – (a) Static JTPG-LSTM is trained using offline data, (b) Dynamic JTPG-LSTM is trained using online data in the base simulation, (c) Multiple just-in-time (JIT) what-if simulation branches are initialized from the base simulation for short-term forecasting

3.1 Overview

In this section, our dynamic data-driven simulation system (shown in Figure 1) involving symbiotic simulation and machine learning is briefly explained for better understanding of the overall system.

3.1.1 ***Static JTPG-LSTM.*** The "jointly trained physics-guided LSTM" is our new hybrid car-following model that consists of a physics-based car-following model (IDM) and a learning-based model (LSTM). This model is extended from [27] and enhanced to ensure safe car-following process since the original formulation does not guarantee the longitudinal collision-free property. The parameters of both IDM and LSTM (in the hybrid model) are jointly learnt from historical offline data. This pre-trained model is then embedded in the base simulation and used along with a selected lane-changing technique to form a complete microscopic traffic simulation model. During the

training process, the IDM provides guidance towards the LSTM model so that the LSTM predicted outputs are consistent with safe driving behaviours. More details about this newly extended model will be elaborated in Section 3.2.

3.1.2 ***Base Simulation.*** In our dynamic data-driven simulation system, the base simulation is a real-time simulation that runs in synchronization with a physical system. In a real-time traffic simulation, driving behaviours may change over time depending on the current traffic situation. Consequently, past driving patterns learnt from offline training data may not be adequate to capture dynamically changing driving behaviours. Hence, this component of the simulation system ensures that the parameters of the integrated traffic simulation model are always updated with the latest sensor measurement data. This dynamic parameter update process could be achieved by using online (machine) learning. In doing so, the static JTPG-LSTM becomes dynamic with online learning capability. The resulting model could capture dynamic car-following behaviours while being aware of safe driving patterns.

3.1.3 ***Just-in-time Simulation.*** From the base simulation, multiple just-in-time (JIT) simulation instances can be replicated for short-term forecasting. This could facilitate what-if analysis required for urgent decision making process in an efficient manner. It is because the parameters of the simulation model have already been updated with online data and thus, it is suitable for situations that demand urgent simulation runs. Accordingly, it is very important to build an accurate and realistic base simulation model so that the error propagation towards its JIT simulation branches could be alleviated. All the above components involved in our dynamic data-driven simulation system are explained in the following sections.

3.2 Jointly Trained Physics-guided LSTM (JTPG-LSTM)

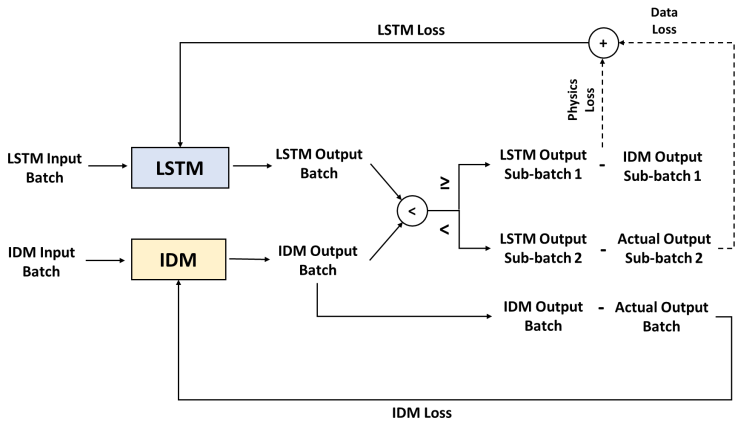

Fig. 2. Joint training process of JTPG-LSTM

In this section, we describe how IDM and LSTM are jointly trained and combined to produce safe and accurate car-following behaviours. The training process of the combined model, JTPG-LSTM, is also illustrated in Figure 2.

3.2.1 ***Intelligent Driver Model (IDM).*** In microscopic traffic simulation, IDM is a widely adopted physics-based car-following model. Since its first development about more than two decades ago [40], it has been still studied nowadays, and extended further in recent literature [2, 37, 43, 50, 58].

It was originally formulated to be collision-free in longitudinal direction [40]. This property is achieved by employing "intelligent braking strategy" which encourages soft braking behavior under normal situations, but causes emergency braking under a critical/near-collision scenario [42] (also see [40] for more details regarding the mathematical analysis of this property). Owing to this formulation, IDM is adopted in this paper so that our extended model, JTPG-LSTM, could inherit this collision-free property. Without loss of generality, it should also function well with other collision-free car-following models and further investigation with them is left as future works.

The IDM can be described by Equations 1 and 2. It produces a follower's acceleration in the next time step (a_{t+1}^{IDM}) as its output based on a set of inputs – the follower's current velocity (v_t); front gap distance to the leader (s_t); the relative velocity with respect to the leader (v_t^{rel}). It also uses constant parameters that are maximum desired velocity (v_0), jam distance (s_0), maximum desired acceleration and deceleration (a_{max}) and (b_{max}) respectively, desired time headway (T) and configurable exponent parameter (δ) often set to 4 in existing studies. Finally, $s^*(.)$ is the minimum desired gap distance which depends on v_t and v_t^{rel}. Since human drivers are unable to perform too many actions during a very short period of time, the time increment is set to 1 second with reference to [46]. This 1-second resolution also applies to all the models used in this paper.

$$a_{t+1}^{IDM} = a_{max}\left[1 - (\frac{v_t}{v_0})^{\delta} - (\frac{s^*(.)}{s_t})^2\right], \tag{1}$$

$$\text{where} \quad s^*(v_t, v_t^{rel}) = s_0 + v_t\,T + \frac{v_t\,v_t^{rel}}{2\sqrt{a_{max}\,b_{max}}}; \quad \delta = 4 \tag{2}$$

3.2.2 ***Long Short-term Memory (LSTM).*** A type of recurrent neural network, LSTM, can be regarded as the state-of-the-art model among data-driven car-following models. It has been extensively studied in existing literature [24, 27, 46, 54] and shown to outperform conventional models in terms of simulation accuracy. It was also adopted in our previous work [30] for modelling car-following behaviours. Our previous LSTM model is modified accordingly in this paper so that it could be integrated better with IDM. Therefore, the rationale behind the choice of input features, input trajectory length, optimization method, and other hyperparameters can be found in our previous work. The implementation of LSTM layer follows the original paper [12]. The internal mechanism of LSTM is controlled by gates and states and the final output can be calculated based on the equations below.

$$f_t = \sigma(W_f s_t + U_f h_{t-1} + b_f) \tag{3}$$

$$i_t = \sigma(W_i s_t + U_i h_{t-1} + b_i) \tag{4}$$

$$o_t = \sigma(W_o s_t + U_o h_{t-1} + b_o) \tag{5}$$

$$c_t = (f_t \circ c_{t-1}) + (i_t \circ tanh(W_c s_t + U_c h_{t-1} + b_c)) \tag{6}$$

$$h_t = o_t \circ tanh(c_t) \tag{7}$$

$$y_t = V h_t + b \tag{8}$$

y_t is the final output after going through the last linear layer; *t* and *t-1* denote current time step and previous time step respectively; operator $\circ$ is the Hadamard product; σ is the logistic sigmoid function; *tanh* is the hyperbolic tangent function; *s* is the vehicle state vector; *f, i* and *o* are forget, input, output gates respectively; *h* and *c* are LSTM hidden states and cell states respectively; *W, U* and *V* are weight parameters in the network; *b* is the bias term. $s \in \mathbb{R}^d$; $f, i, o, h, c \in \mathbb{R}^m$; $W_f, W_i, W_o, W_c \in \mathbb{R}^{m \times d}$; $U_f, U_i, U_o, U_c \in \mathbb{R}^{m \times m}$; $b_f, b_i, b_o, b_c \in \mathbb{R}^m$; $V \in \mathbb{R}^{1 \times m}$; $b \in \mathbb{R}$; *d* and *m* are dimensions of the vehicle state vector and the hidden units respectively.

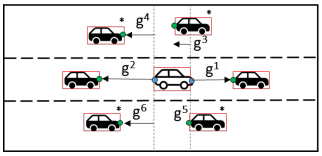

Fig. 3. Surrounding gap distances (adapted from [30])

As for LSTM, the state of a vehicle at a given time step can be represented as follows.

$$s = \{x, lane, v, v_{rel}, a, length, g^1, g^2, g^3, g^4, g^5, g^6\} \tag{9}$$

$$g^{\mathrm{i}} = \begin{cases} x_{veh_l} - length_{veh_l} - x_{veh}, & \text{if } veh_l \text{ is present} \\ x_{veh} - length_{veh} - x_{veh_f}, & \text{if } veh_f \text{ is present ; } i \in \{1, ..., 6\} \\ g_{default}, & \text{otherwise} \end{cases}$$

Let *veh, veh_l, veh_f* denote a vehicle of our interest, a leading vehicle in front of *veh*, a following vehicle behind *veh* respectively. Leading and following vehicles can be travelling either on the same lane or adjacent lanes. x is longitudinal position; *lane* is the current lane on which *veh* is driving; v is its velocity; v_{rel} is the relative velocity w.r.t its leader; a is its acceleration/deceleration; *length* is a vehicle's length; g^i denotes the longitudinal gap distance between *veh* and another vehicle (see Figure 3); when i∈{1,3,5}, the gap is associated with leading vehicles from the same lane, left adjacent lane and right adjacent lane respectively; likewise, when i∈{2,4,6}, the gap is associated with following vehicles on respective lanes. $g_{default}$ is set to 100 m (an arbitrarily chosen sensing range).

To account for temporal dependency and driver memory effect, the LSTM uses a follower's state trajectory of pre-defined length as its inputs and produces the next acceleration as its output. The goal of LSTM is to approximate the function f in Equation 10 by learning the optimal values of parameters θ (i.e., weights and biases of the network) from training data.

$$a_{t+1}^{LSTM} = f(s_{t-n+1}, s_{t-n}, ..., s_t|\theta) \tag{10}$$

3.2.3 ***Model Output***. In order to ensure collision-free car-following behaviours, a safe acceleration upper bound should be defined for a_{t+1}^{LSTM}. Since IDM can guarantee longitudinal collision-free driving, a_{t+1}^{IDM} can serve as the upper bound for a_{t+1}^{LSTM}. Then, the final output for JTPG-LSTM can be defined as follows.

$$a_{t+1}^{JTPG-LSTM} = \min(a_{t+1}^{LSTM}, a_{t+1}^{IDM}) \tag{11}$$

In fact, it is possible to separately train or calibrate the parameters of the two models and then, only during the deployment stage, the final output acceleration could be revised according to Equation 11. This can be considered as a naive combination of the two methods since the underlying data-driven model does not receive any guidance from the physics-based model. Although such an approach is likely to work if the representative quality of the training data is high, its generalisation performance can degrade severely when the data lacks a diverse set of training samples. Therefore, it is necessary to prevent the data-driven model from overfitting to the training data by providing guidance from the physics-based model during the training process. One popular way of doing so (in physics-informed ML literature) is to regularize the loss function by adding "physics loss" that measures the deviation between the data-driven model output and the physics-based model output. Consequently, the data-driven model is able to learn not only from training data but also

from guidance provided by the physics-based model so that its output values can be consistent with physical laws.

3.2.4 ***Loss Functions.*** In this paper, both IDM and LSTM's parameters are jointly trained through backpropagation which is a standard learning method for neural networks. Hence, the IDM's parameters are calibrated via gradient-based optimization rather than conventional gradient-free calibration approaches such as Genetic Alogrithm (GA) adopted in most studies [7, 30, 35]. At each iteration, a batch of training samples (B) are randomly selected from training data and used to train the two models. The visual illustration of various losses involved in the JTPG-LSTM's training process is shown in Figure 2.

Our newly defined loss functions for LSTM in the joint training process are:

$$Loss_{LSTM} = Loss_{Data} + Loss_{Physics} \tag{12}$$

$$Loss_{Data} = \frac{\sum_{i=1}^{B} \mathbb{1}[C_1]\,(a_i^{LSTM} - a_i^{act})^2}{\sum_{i=1}^{B} \mathbb{1}[C_1]}\ ;\ where\ C_1 := a_i^{LSTM} < a_i^{IDM} \tag{13}$$

$$Loss_{Physics} = \frac{\sum_{i=1}^{B} \mathbb{1}[C_2]\,(a_i^{LSTM} - a_i^{IDM})^2}{\sum_{i=1}^{B} \mathbb{1}[C_2]}\ ;\ where\ C_2 := a_i^{LSTM} \geq a_i^{IDM} \tag{14}$$

$Loss_{LSTM}$ is the total loss used for updating the LSTM parameters. a_i^{LSTM} and a_i^{IDM} refer to the predicted acceleration by LSTM and IDM for an i^{th} training sample respectively. a_i^{act} is the actual ground-truth acceleration of an i^{th} training sample. $\mathbb{1}[.]$ is the indicator function to check if a certain constraint (either C_1 or C_2) is satisfied.

The LSTM predicted outputs can be divided further into two sub-batches based on safe driving criteria as follows. The acceleration given by IDM helps define the safe driving criteria to determine whether the LSTM outputs are safe or unsafe. The LSTM predicted acceleration is considered safe if it does not exceed that of IDM. Otherwise, it is considered unsafe and hence, it is likely to result in a collision with a leading vehicle. According to Figure 2, these unsafe LSTM outputs belong to *LSTM Output Sub-batch 1* whereas the safe outputs belong to *LSTM Output Sub-batch 2*.

On one hand, $Loss_{Physics}$ is derived from *LSTM Output Sub-batch 1* (where $a_i^{LSTM} \geq a_i^{IDM}$) meaning that the predicted acceleration by LSTM in these samples are unsafe and need to be regularized with guidance from IDM. Hence, it is defined as the mean squared error loss calculated between the LSTM and IDM predicted acceleration. On the other hand, $Loss_{Data}$ is based on *LSTM Output Sub-batch 2* (where $a_i^{LSTM} < a_i^{IDM}$) implying that the predicted acceleration by LSTM in these samples are safe and hence, no regularization is required. Thus, it is defined as the mean squared error loss calculated between the LSTM predicted and ground-truth acceleration.

As for IDM, it is only required to learn from training data to provide more accurate guidance towards LSTM. Thus, its loss function can be defined without any constraint as:

$$Loss_{IDM} = \frac{1}{B}\sum_{i=1}^{B}(a_i^{IDM} - a_i^{act})^2 \tag{15}$$

Algorithm 1: JTPG-LSTM Training

Result: LSTM and IDM with trained parameters θ^* and ϕ^* respectively

1 **Initialization**:

2 Initialize training hyperparameters for LSTM and IDM;

3 Transform LSTM training input values, into [-1, 1] using Min-Max scaling;

4 Transform ground-truth output values into [-1, 1] using Min-Max scaling;

5 Initialize IDM parameters ϕ using predefined values with lower and upper bounds for ϕ;

6 Initialize LSTM parameters θ using random values drawn from $\mathcal{U}(-\sqrt{\frac{1}{m}}, \sqrt{\frac{1}{m}})$;
where m := dimension of LSTM hidden state vector h;

7 **for** *epoch i = 1, ..., E* **do**

8 Randomly shuffle the training data instances;

9 Prepare N unique batches of training data with individual batch size B;

10 **for** *iteration j = 1, ..., N* **do**

11 Get a j^{th} training batch for model training; where $D^{in}_{LSTM}, D^{in}_{IDM}, D^{out}_{TRUTH} \in B_j$;

12 $Y^{out}_{LSTM} = LSTM(D^{in}_{LSTM}|\theta_j)$;

13 $Y^{out}_{IDM} = IDM(D^{in}_{IDM}|\phi_j)$;

14 Calculate $Loss_{LSTM}$ among Y^{out}_{LSTM}, Y^{out}_{IDM} and D^{out}_{TRUTH} using Equation 12, 13 and 14;

15 Calculate $Loss_{IDM}$ between Y^{out}_{IDM} and D^{out}_{TRUTH}using Equation 15;

16 θ_{j+1} = Update θ_j using $Loss_{LSTM}$ with RMSProp optimization;

17 ϕ_{j+1} = Update ϕ_j using $Loss_{IDM}$ with RMSProp optimization;

18 **end**

19 **end**

3.2.5 ***Hyperparameters.*** With reference to our previous work [30], the following hyperparameters are used in the JTPG-LSTM training process: dimension of the input layer=12; dimension of the output layer=1; batch size = 64; input trajectory length=10 (sec); number of LSTM layer=1; number of neurons in LSTM layer=10; training epochs=150; optimization method=RMSProp; learning rate=0.001. The IDM parameter bounds are inferred from the training datasets, the road speed limit of the study area, and most importantly, the acceleration/deceleration bounds are based on a systematic study conducted in [6]. The resulting lower and upper bounds are: v_0=[10, 33.3333] m/s; a_{max}=[0.28, 3.41] m/s^2; b_{max}=[0.47, 3.41] m/s^2; T=[1, 3] sec; s_0=[1, 5] m.

3.2.6 ***Model Training.*** The joint training algorithm for our extended model, JTPG-LSTM, is described in Algorithm 1 and illustrated in Figure 2.

In *line 2*, the hyperparameters required in the training process are initialized. From *lines 3-4*, the training input and output values are normalized to the range [-1, 1] using Min-Max scaling. The feature scaling is a common practice in ML model training for achieving fast convergence and reducing bias that stems from the original ranges of different feature values. However, the training input data for IDM should not be scaled because the original value of each input variable has meaningful impact on the IDM output. It is also implicitly implied that the output values should be scaled or re-scaled back and forth between the original range and the scaled range whenever necessary. Hence, the scale transformations processes should be inferred based on the context although they are not explicitly described in the algorithm.

From *lines 5-6*, both IDM and LSTM parameters, ϕ and θ, are initialized with predefined values and random values drawn from a uniform distribution with the range $[-\sqrt{\frac{1}{m}}, \sqrt{\frac{1}{m}}]$, respectively. Since our model, JTPG-LSTM, is implemented in C++14 by using Libtorch which is a pytorch C++ API [1], this LSTM weight initialization follows their default implementation. In *line 5*, it is important to set lower and upper bounds for IDM parameters because during the optimization process, they could result in implausible or invalid values (for instance, a_{max} and b_{max} are in the denominator and hence, should never be zero).

In *line 7*, the model training process begins and runs for the predefined number of epochs (E). In *line 8*, the entire training data are randomly shuffled at each epoch. In *line 9*, a series of unique training batches (N) are prepared since mini-batch stochastic gradient descent optimization is adopted. In *line 10*, we iterate through each batch of data for training our model. At each iteration, a j^{th} batch of training input data are passed to LSTM and IDM for inference. Their predicted outputs (i.e., the next acceleration of a following vehicle) are obtained in *line 12* and *13* correspondingly. Then, from *lines 14-15*, both LSTM and IDM losses are calculated according to the equations defined in Section 3.2.4. Finally, the parameters of the two models are updated with their respective losses by using the RMSProp optimization technique in *line 16* and *17*. After each update, the IDM parameters are always adjusted to be within the defined lower and upper bounds.

3.2.7 ***Model Testing.*** During testing (deployment) stage, the final output acceleration for JTPG-LSTM can be calculated simply by using Equation 11. The output acceleration of IDM serves as the safe acceleration upper bound for that of LSTM, thus ensuring collision-free car-following process. Based on the predicted acceleration, we could only test the model accuracy. In order to assess the model performance of safe driving, the longitudinal position information needs to be derived from the acceleration.

In microscopic traffic simulation, the quality of trajectories simulated by time-continuous car-following models are subject to the choice of numerical integration schemes. A commonly adopted integration scheme is the first-order Euler's method which is often used for deriving the velocity and position of a vehicle based on the model predicted acceleration. Higher order integration schemes such as the forth-order Runge-Kutta method (RK4) may not offer the best results as investigated in [41] and even turned out to perform worse than the simple Euler's method in their investigation. Based on their analysis, we have chosen the ballistic method as our integration scheme for two reasons – (i) it allows to implicitly model the reaction time of a driver without introducing any explicit delay to the formulation [17], and (ii) it performs consistently better than Euler's method since it was only subject to 30% of the discretization errors as compared against the latter [41]. The ballistic update method can be defined as follows.

$$x_{t+1} = x_t + v_t \Delta t + \frac{1}{2} a_t (\Delta t)^2; \;\; x_{t+1} \geq x_t; \tag{16}$$

$$v_{t+1} = v_t + a_t \Delta t; \;\; v_{t+1} \geq 0; \;\; \Delta t = 1\, second \tag{17}$$

The above notations are already introduced when describing the state of a vehicle in Equation 9. The first constraint $x_{t+1} \geq x_t$ is used to ensure that the excessive deceleration does not lead to an updated position that is even lagged behind the current position. The second constraint $v_{t+1} \geq 0$ is used to avoid negative velocity after the update.

3.2.8 ***Model Novelty.*** Our JPTG-LSTM training algorithm is different from our reference work [27] in a number of ways. Firstly, their underlying LSTM model is simplified for two reasons – 1) it uses the same input feature set as IDM to represent the current vehicle state, 2) it considers only up to the input trajectory length of 3 seconds. This is in contrast to an existing study [47] which

has empirically verified that using long memory (10 second trajectory length) is important for a RNN-based car-following model in order to reproduce commonly observed traffic phenomenon. Secondly, in their joint training process, it seems that the physics-based model does not depend on the ground-truth output data for calculating the prediction error loss. Instead, it also relies on physics loss for training the constant parameters of the physics-based model. Ultimately, their loss function for the learning-based model is defined as the weighted average between physics loss and data loss whereas our LSTM loss function is defined as the sum of two losses where each loss is derived from selective ground-truth outputs and model predictions according to safe (or unsafe) acceleration criteria defined in Equations 13 and 14. In our formulation, it is also not necessary to weight the two losses for calculating the total loss because the combined size of *LSTM Output Sub-batch 1* and *2* is equivalent to the original training batch size. Hence, this implicitly accounts for weighting the two losses automatically through the learning process.

During both training and testing stages, their model output acceleration is solely determined by a learning-based model. As a result, the model is still prone to vehicle collisions since the collision-free car-following property is not guaranteed. On the contrary, in our case, the physics-based model (IDM) is not only involved in the joint training process for helping LSTM produce physically consistent outputs, but also in the deployment stage for ensuring the model output is collision-free. Nevertheless, our pre-trained JTPG-LSTM is an offline model whose parameters are static and may not be ready for capturing dynamic car-following behaviours. Since our goal is to integrate JTPG-LSTM to the real-time base simulation, it will be enhanced with online learning in the next section for updating its parameters dynamically.

3.3 Dynamic JTPG-LSTM

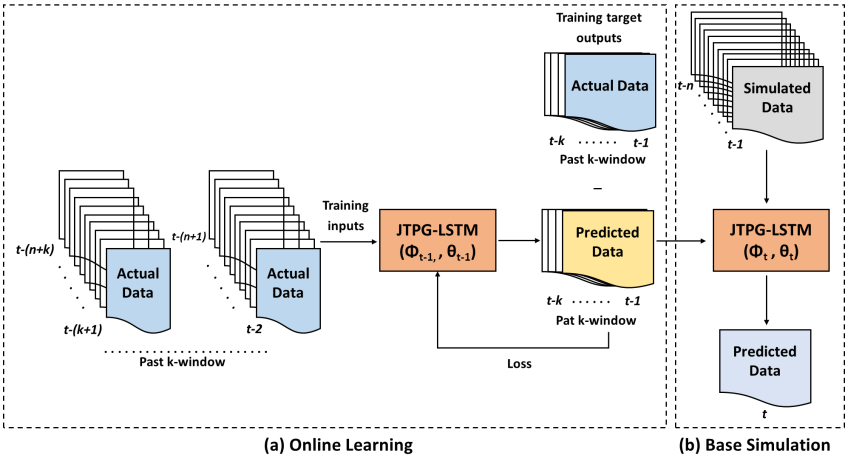

Fig. 4. (a) Online learning process of dynamic JTPG-LSTM; (b) Base simulation using dynamic JTPG-LSTM

To employ JTPG-LSTM in a real-time traffic simulation, it is desirable to update its parameters dynamically with online data so that the base simulation results could reflect closely to the ongoing traffic situation. The Algorithm 2 is adapted from our previous work [30] and modified accordingly so that it could be used to develop dynamic JTPG-LSTM with online learning capability. The online learning process is also shown in Figure 4(a) for visual illustration.

In *line 1* of the algorithm, the two models, LSTM and IDM, are initialized with their pre-trained parameters, θ^* and ϕ^* respectively. They are pre-trained by using offline traffic data via Algorithm

Algorithm 2: Dynamic JTPG-LSTM Online Learning

Result: JTPG-LSTM with updated parameters ($\hat{\theta}$ for LSTM and $\hat{\phi}$ for IDM)

1 **Initialization**: LSTM with pre-trained parameters θ^*; IDM with pre-trained parameters ϕ^*; window size k; trajectory length n;

2 **Constraints**: C_3 := input vehicle trajectory length $\geq$ n;

3 **for** *t = 1, ..., T* **do**

4 Ground-truth data D_{t-1}, arrived at t-1, where actual vehicle states $S_{t-1} \in D_{t-1}$;

5 **if** $t \geq k$ **then**

6 $D^{out}_{TRUTH} = D^{out}_{t-1} \cup D^{out}_{t-2} \cup ... \cup D^{out}_{t-k}$;

7 **else**

8 $D^{out}_{TRUTH} = D^{out}_{t-1} \cup ... \cup D^{out}_{1}$;

9 **end**

10 Get valid input data D^{in}_{TRUTH} w.r.t C_3, where $D^{in}_{TRUTH} = D^{in}_{t-2} \cup D^{in}_{t-3} \cup ... \cup D^{in}_{t-(k+1)}$;

11 **if** D^{in}_{TRUTH} *exist* **then**

12 $D^{in}_{LSTM}, D^{in}_{IDM} \in D^{in}_{TRUTH}$;

13 Transform D^{in}_{LSTM} values, into [-1, 1] using Min-Max scaling;

14 Transform D^{out}_{TRUTH} values into [-1, 1] using Min-Max scaling;

15 $Y^{out}_{LSTM} = LSTM(D^{in}_{LSTM}|\theta_{t-1})$;

16 $Y^{out}_{IDM} = IDM(D^{in}_{IDM}|\phi_{t-1})$;

17 Calculate $Loss_{LSTM}$ among Y^{out}_{LSTM}, Y^{out}_{IDM} and D^{out}_{TRUTH} using Equation 12, 13 and 14;

18 Calculate $Loss_{IDM}$ between Y^{out}_{IDM} and D^{out}_{TRUTH}using Equation 15;

19 θ_t = Update θ_{t-1} using $Loss_{LSTM}$ with RMSProp optimization;

20 ϕ_t = Update ϕ_{t-1} using $Loss_{IDM}$ with RMSProp optimization;

21 **end**

22 **end**

1. Furthermore, the window size *(k)*, which defines the past frame range to collect previously arrived traffic data, and the LSTM input trajectory length *(n)* are also configured. One constraint C_3 is defined to ensure the training input data have sufficient length for LSTM. Future works may eliminate this constraint by training the LSTM model in a specific setting such that it can function with variable trajectory length. As for IDM, it only uses the vehicle state from previous time step as its inputs.

In *line 3*, the online learning process begins and runs until the end of simulation model time T because this process is being executed as part of the base simulation. At every time step, ground-truth real-time traffic data that contain actual vehicle states are arriving at our disposal (*line 4*). From *lines 5-9*, k-most recent ground-truth output data are collected and stored in D^{out}_{TRUTH} which contain actual observed acceleration of the vehicles from k-most recent time steps. In line 10, all valid input data D^{in}_{TRUTH} from k-most recent time steps, that satisfy C_3 are also collected. The D^{in}_{TRUTH} contain the training input data required for both LSTM and IDM (*line 12*). D^{in}_{LSTM} contain vehicle state trajectories of length (n) while D^{in}_{IDM} only rely on vehicle states from the previous

time step. Finally, from *lines 13-20*, it follows the exact same parameter update process used in Algorithm 1. After executing the above steps, as illustrated in Figure 4(b), the JTPG-LSTM with updated parameters (θ_t, ϕ_t) is used to predict the next acceleration (a_t) for vehicles running in the base simulation. It should be noted that we rely on previously arrived ground-truth data for online learning whereas only simulated vehicle data are used as model inputs for acceleration prediction in the base simulation.

3.4 Base Simulation using Dynamic JTPG-LSTM

In our simulation system, the base simulation is a real-time traffic simulation that uses dynamic JTPG-LSTM as its car-following model. The benefits of having the base simulation are three-fold – 1) It can assess whether the current simulation model is adequate to represent the state of the physical system, 2) It is efficient since it can shorten the setup time required to carry out just-in-time (JIT) what-if simulation, and 3) It is effective since the model parameters are constantly updated with online data for better short-term forecasting.

The base simulation is executed as follows. Upon the arrival of real-time traffic data, online learning (see Algorithm 2) is first carried out to dynamically update the parameters of JTPG-LSTM. This online learning process occurs at every simulation step. Then, based on the arrived data, new vehicles are initialized in the simulation as well as redundant vehicles are removed from the simulation. In a given time step, vehicles that are still running in the simulation but no longer present in the arrived traffic data are identified as redundant vehicles. After that, dynamic JTPG-LSTM with updated parameters is used to provide the next acceleration of simulated vehicles (see Figure 4(b)). During the execution of the base simulation, all these necessary computational works that are required for online parameter update, predicting vehicle acceleration, vehicle initialization and removal, other computational overheads, etc. have to be completed within the time duration of less than 1 second (i.e., our choice of simulation time resolution) so that it can keep up with the physical system consistently.

3.5 Just-in-time What-if Simulation

For JIT what-if simulation, the simulation model that has been already calibrated up to the latest time step can be used. Such model is more likely to provide better short-term forecasts than conventional offline models. On one hand, the base simulation which runs in pace with the physical system could be executed so that the virtual world could be synchronized with the real world while updating the underlying model's parameters dynamically. On the other hand, multiple offline JIT simulation could be branched out from the base simulation and run faster than the physical system (since they do not need to wait for real-time data arrival) to facilitate short-term forecasting of different performance measures. Therefore, the necessary operational management decision can be made just in time to address the ongoing situation of a physical system in an efficient manner.

4 EXPERIMENTS

4.1 Experiment Setup

The experiments were conducted on a computer operating on Ubuntu 20.04.3 LTS with an AMD EPYC 7742 64-Core processor, 256 GiB system memory and two NVIDIA GeForce RTX 2080 Ti graphics cards. A highway road traffic simulator was implemented in C++14 for simulation experiments unlike our previous work [30] which used SUMO [22] and relied on its internal safety check mechanism. This is because in this paper, we aim to assess the performance of our proposed model, JTPG-LSTM, in terms of both simulation accuracy and safe driving criteria. All the models involved in the experiments were also implemented in C++14 with the help of Libtorch which is

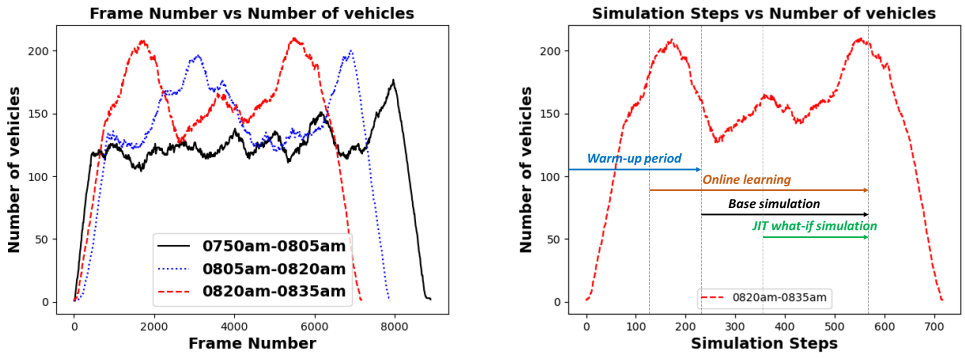

Fig. 5. (a) Vehicle density as captured through video frame sequences in each dataset (left), and (b) Simulation experiment setup (right)

a pytorch C++ API [1]. Model training, inference and online learning were carried out by using only one GPU card. No other component of the simulator utilized any GPU resource. Nevertheless, SUMO was still used for visual animation required in the qualitative performance assessment.

The popular Next Generation Simulation (NGSIM) US Highway 101 dataset [9] was used for our experiments. The trajectory dataset contains three sub-datasets that were recorded for 15 mins period each, *Dataset 1* - 07:50-08:05 AM, *Dataset 2* - 08:05-08:20 AM, and *Dataset 3* - 08:20-08:35 AM, correspondingly. The datasets consist of vehicle state trajectories that were pre-processed from recorded videos. The vehicle density profile of each recorded dataset can be seen in Figure 5(a).

The following assumptions were made in the implementation of real-time base simulation that consists of dynamic JTPG-LSTM: (i) *Real-time simulation* is only pseudo real-time simulation based on the NGSIM US Highway 101 dataset [9]; (ii) *Simulation coverage* is defined to only consider vehicles that are travelling on five major lanes from lane 1 to lane 5 (see Figure 6). The rightmost auxiliary lane, on-ramp and off-ramp (lanes 6, 7, 8 respectively) are not considered due to data inconsistency issues associated with these lanes as highlighted in [8]; (iii) *Data quality* is assumed to be high with minimal noise due to the recent advancement of big data collection and processing technology in Intelligent Transportation Systems [55]; (iv) *Network latency* is negligible since the location of the processing server is assumed to be within close vicinity of the study area. Hence, the already-processed real-time traffic data could arrive without any delay or loss at our disposal at every time step; (v) *Warm-up period* is usually required for the simulation to reach its stable state. However, in this section, it is merely pseudo-warm-up period since no vehicle is simulated but instead, all the movement of vehicles are based on actual observation data.

The study area was divided into two zones - 1) Initialization zone and 2) Simulation zone as illustrated in Figure 6. Initialization zone was used to add new vehicles to the simulation with their actual vehicle state information. Vehicles that were travelling in this zone used their actual vehicle position for vehicle movement. Conversely, if they had already reached the simulation zone, their movement would be based on acceleration predicted by a selected model. Simulation output statistics were also collected from this zone for the analysis of different performance measures.

4.2 Data Pre-processing

Each dataset has a resolution of 100 millisecond among all extracted data instances of 1,048,575. Since we are interested in trajectories with resolution of 1 second, data instances were sampled at every 1 second. The unique simulation time stamps were added based on the original frame

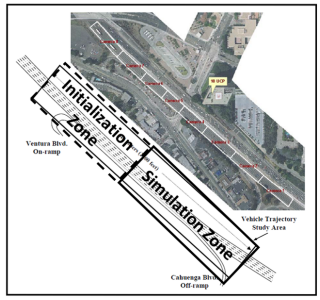

Fig. 6. Study area: US Highway 101 (adapted from [9])

sequence number for better referencing. The data instances that are associated with lanes 6, 7 and 8 were eliminated. Original measurement unit is in feet and hence, the measurement values in feet were converted to meter. The resulting data were further pre-processed to extract surrounding vehicle distances. Since there is a sharp increase and decrease in terms of vehicle density at the beginning and end of each dataset due to the camera recording transition, the data instances from these periods were filtered out when preparing training data for JTPG-LSTM. Normal passenger cars form considerably large proportion of the data (97%) while motorcycles and trucks only constitute very small proportions (1% and 2%, respectively) of the data. The trajectory data associated with motorbikes were removed since they caused overlapping trajectories among vehicles. The inconsistent data points that have negative front or rear gap distances were also removed. Upon detailed investigation, these instances were caused by a total of 40 problem vehicles and hence, they were also excluded from the simulation. Finally, the required training input and output data for both LSTM and IDM were prepared by ensuring a given follower has the same leader in both the current time step *(t)* and the next time step *(t+1)*. The *Dataset 1* and *Dataset 2* having 53,748 and 69,340 data instances respectively were used as the training data while the performance of different models was tested on *Dataset 3* with 75,850 data instances.

4.3 Evaluation Metrics

4.3.1 ***$RMSE_a$***. For N test data instances, the root mean squared error (RMSE) between the actual ground-truth acceleration and predicted acceleration is used to assess the car-following accuracy of different models. It is defined as:

$$RMSE_a = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i^{act} - a_i^{pred})^2} \tag{18}$$

where *act* and *pred* stand for "actual" and "predicted" respectively. However, the above $RMSE_a$ is insufficient to assess whether a given car-following model is safe. The information about the updated vehicle position is required in order to do so.

4.3.2 ***$RMSE_v$ and $RMSE_x$***. Updated position and velocity values can be derived via ballistic update method which is described in Equations 16 and 17 respectively. Then, the following RMSEs for both velocity and positions could be calculated and used for performance comparison as well.

$$RMSE_v = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(v_i^{act} - v_i^{pred})^2}; \quad RMSE_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i^{act} - x_i^{pred})^2} \tag{19}$$

4.3.3 ***Number of Collisions (NOC).*** Using the updated positions of following vehicles, we could verify if any of them has collided with their leaders by checking whether their front gap distances (after the update) are less than or equal to zero for a given test sample. Accordingly, the total number of collisions can be calculated simply by counting those test samples that meet this criteria.

$$NOC = \sum_{i=1}^{N} \mathbb{1}[gap_{front} \leq 0]; \quad gap_{front} = x_l^{act} - length_l^{act} - x_f^{pred} \tag{20}$$

where f and l stand for "follower" and "leader" respectively.

4.3.4 ***Velocity Trajectory Deviation Error (VTDE).*** To quantitatively assess the performance of both base and JIT simulation, the velocity trajectory deviation error (VTDE) is defined in Equation 21 by calculating the deviation between the observed and simulated velocity throughout the entire trip of each vehicle.

$$VTDE = \sqrt{\frac{1}{V}\sum_{j=1}^{V}[\frac{1}{L}\sum_{i=1}^{L}(v_{ji}^{act} - \mathbb{1}_i[v_{ji}^{sim}])^2]} \tag{21}$$

where L is the length of trip duration (in sec) for j^{th} vehicle; V is the total number of vehicles simulated during the study period; v_{ji} is the velocity of a j^{th} vehicle at i^{th} time, and *act* and *sim* refer to actual and simulated respectively. $\mathbb{1}_i[v_{ji}^{sim}]$ is an indicator function to check if the simulated velocity of a vehicle exists at i^{th} time.

4.4 Offline Modelling

The goal of this offline modelling experiment is to select the best offline car-following model considering the balance between model accuracy and safe driving. The parameters of all offline models are static and trained by using historical offline data. Although each model was trained for 150 epochs, their training losses converged much earlier than the scheduled end time. The following models are considered for performance comparison.

(1) **Default IDM** is the pure physics-based IDM with default parameters inferred from [27, 40].
(2) **Static LSTM** is the pure data-driven LSTM developed in our previous work [30].
(3) **DTPI-LSTM** is the "disjointly trained physics-informed LSTM" developed based on our reference work [27]. In this hybrid model, the IDM parameters were calibrated separately via backpropagation. The calibrated IDM with fixed parameters was then used in the LSTM training process for calculating physics loss.
(4) **JTPI-LSTM** is the "jointly trained physics-informed LSTM" developed with reference to [27] and modified accordingly so that the IDM parameters could be trained by using the data loss during the joint training process.
(5) **JTPG-LSTM** is our new car-following model that is extended from [27] so that it can produce both accurate and collision-free car-following behaviours. To differentiate between JTPG-LSTM and the two models: DPTI-LSTM and JTPI-LSTM, it can be referred to Section 3.2.8.

The offline modelling experiment results are presented in Table 2. Different RMSEs were used to assess the model accuracy while NOC was used to evaluate safe driving behaviours. The parameter values for each calibrated IDM are also tabulated. According to the results, the best model accuracy is achieved by static LSTM which is a pure data-driven model. Despite being the best, it resulted in a total of 9 collisions which is the second-worst performance in terms of safe driving criteria among all models. This is because our experimental setting was controlled in a way that all models were subject to "distributional shift", a phenomenon where there is a mismatch between the training data distribution and the test data distribution [19, 26, 36]. In other words, the training data is

composed of *Dataset 1* and *Dataset 2* in which the traffic was the least congested and moderately congested respectively. However, the test dataset, *Dataset 3*, contains the most congested traffic data. Consequently, upon encountering the test data that contains unprecedented scenarios, the acceleration produced by the pure data-driven LSTM were no longer safe. It emphasizes the fact that the most accurate model is not necessarily the best since a good car-following model should produce accurate driving behaviours while being aware of the trade-off between the accuracy and the safe driving. On the contrary, even the IDM with default parameters, that was completely unaware of the training data, suffices to avoid collisions in spite of its worst accuracy.

As for our reference models – DTPI-LSTM and JTPI-LSTM, although they were informed by a physics-based model (IDM) during the training process, their model outputs were solely determined by the data-driven model (LSTM). Thus, they still could not guarantee the longitudinal collision-free property, rendering them prone to collisions due to the weakness of the data-driven model. As a result, total 14 and 9 collisions with leading vehicles were observed when testing both DTPI-LSTM and JTPI-LSTM respectively. Finally, our proposed model, JTPG-LSTM, resulted in an error decrease of about 20.14% as compared to Default IDM whereas the error increased by about 3.54% with respect to the most accurate static LSTM. Most importantly, no collision was observed due to the predicted acceleration of JTPG-LSTM. It shows that our JTPG-LSTM is able to learn a desirable balance between model accuracy and safety during the joint-training process. Moreover, it could also ensure collision-free car-following due to the safe acceleration upper bound provided by its physics-based IDM. Even though the test data was subject to the distributional shift, JTPG-LSTM successfully maintained its safe car-following behaviour by slightly compromising its accuracy.

We also varied the amount of training data used (from 20% to 100%) to compare the performance between static LSTM and JTPG-LSTM. According to the results (provided in our supplementary materials [29]), JTPG-LSTM outperformed static LSTM by about 3.9% in terms of acceleration prediction when using only 20% of training data. In the mean time, the former had zero collisions, but the latter led to 29 collisions. In alignment with our reference work [27], these findings suggest that the physics-guided JTPG-LSTM is also more data-efficient than the pure data-driven static LSTM.

Table 2. Offline car-following model performance comparison

Model	$RMSE_a$	$RMSE_v$	$RMSE_x$	NOC	v_0	T	s_0	a_{max}	b_{max}
Default IDM	1.5494	1.1363	0.7352	**0**	30	1.5	2	0.73	1.63
Static LSTM	**1.1951**	**0.7196**	**0.5897**	9	*Nil*	*Nil*	*Nil*	*Nil*	*Nil*
DTPI-LSTM	1.2091	0.7388	0.5923	14	33.3232	0.621	1.8375	0.2801	3.406
JTPI-LSTM	1.2057	0.7332	0.5915	9	33.3178	0.6218	1.8148	0.28	3.4069
JTPG-LSTM	**1.2374**	**0.7559**	**0.6005**	**0**	33.3143	0.6298	1.8529	0.2803	3.4069

4.5 Online Modelling - Dynamic JTPG-LSTM

This online modelling experiment aims to investigate how much recent data should be considered during each online parameter update for dynamic JTPG-LSTM. Different sizes of past window (k) can affect both JTPG-LSTM prediction performance and online update duration. The same test dataset (as in Section 4.4) was used for validating dynamic JTPG-LSTM.

According to the results shown in Table 3, dynamic JTPG-LSTM has achieved the best prediction performance via online learning with k=10. Incorporating more past data in the online update process did not lead to better performance when larger k values were used. It has only reduced the acceleration prediction error by 0.53% (from 1.2374 to 1.2309) compared to its offline version.

This is insignificant compared to an error reduction of 7.97% observed in our previous work [30]. It might be due to two reasons as follows. The first reason is that the generalisation capability of static JTPG-LSTM has improved unlike the pure data-driven static LSTM developed in our previous work. With better generalisation ability, it becomes more resistant to the distributional shift and hence, online learning might not help improve its original performance that much. Another reason could be the regularization effect introduced by the physics-based model (from the physics loss) which hinders the online data fitting process. Since we made an assumption that the noise in the data is minimal, the online learning performance may be improved by using only the data loss to dynamically update the model parameters. Research challenge on how to dynamically prioritize the physics loss and the data loss based on the quality of real-time data still remains to be addressed and left for future works.

In the experiments, the actual observation data were used as inputs to JTPG-LSTM. Thus, the model performance, when employed in the base simulation, has not been tested yet. This will be assessed in the next section by comparing against other benchmark models.

Table 3. Acceleration prediction for following vehicles using dynamic JTPG-LSTM

Past window size (k)	$RMSE_a$	$RMSE_v$	$RMSE_x$	NOC
0	1.2374	0.7559	0.6005	0
1	1.233	0.7523	0.5983	0
5	1.232	0.7524	0.5984	0
10	**1.2309**	**0.752**	**0.5983**	0
15	1.231	0.7524	0.5984	0
20	1.2311	0.7525	0.5984	0
25	1.2312	0.753	0.5987	0

4.6 Base Simulation

The purpose of this section is to determine the best mobility model suitable for the base simulation for our dynamic data-driven simulation system. It is important to choose the mobility model that can closely reflect the real-world driving behaviours. If a base simulation is inadequate to model the physical system, then it has to be restarted frequently to prevent significant deviation from reality [3]. Furthermore, since JIT what-if simulation is carried out by relying on the base simulation, an incompetent base simulation model can propagate its error to JIT simulation as well. Hence, it is required to use a mobility model that can address the above issues.

The *Dataset 3* was used as the online test data to assess the performance of different mobility models. All the models were pre-trained by using two offline datasets, *Dataset 1* and *Dataset 2*. During the base simulation, dynamic JTPG-LSTM had used periodically arrived real-time data to update its parameters via online learning. A pseudo real-time traffic simulation was set up using the following configuration parameters (see Figure 5(b)): simulation road length=670m; initialization zone=0-269m; simulation zone=270-670m; simulation duration=560 steps (about 9.33 min); warm-up period=0-219 steps (about 3.65 min); online learning period=120-560 steps (about 7.33 min); base simulation period=220-560 steps (about 5.68 min).

All car-following models used in this section rely on MOBIL [18] for lane-changing process. This lane-changing model always has to be coupled with a car-following model for estimating acceleration of surrounding following vehicles. It then calculates total acceleration gain or loss due to a lane change and makes the final decision with the aim of "minimizing overall braking induced by the lane change". It is usually used along with IDM, in literature, to serve as an overall

microscopic traffic simulation model. In this paper, for simplicity, when using MOBIL, only the physics-based model, IDM (out of the combined model, JTPG-LSTM), was used for estimating acceleration of the following neighbors. More fine-grained coupling which is indeed required for more accurate lane-changing decision is left as our future works. The MOBIL parameters are referred to the original paper [18] and fixed as follows: politeness factor (p)=0.1; lane-changing threshold (Δa_{th})=1; maximum safe deceleration (b_{safe})=b_{max} taken from IDM.

The mobility models compared in this experiment are:

(1) **Mobility model 1 - Static IDM** consists of the pure physics-based model IDM and MOBIL. The IDM was pre-trained through backpropagation using the offline training data and thus, its parameters were calibrated with the gradient-based optimization rather than other gradient-free methods such as Genetic Algorithm (GA). This was intended for fair comparison with our proposed model, JTPG-LSTM, which uses the same optimization technique.

(2) **Mobility model 2 - Dynamic IDM** is the online version of Mobility model 1. Dynamic IDM would make use of periodically arrived real-time data to calibrate its parameters through the same optimization process. The dynamic parameter calibration was done during the same period that was allocated for online learning of dynamic JTPG-LSTM. Furthermore, the same amount of recent data was used during the online update process.

(3) **Mobility model 3 - Static JTPG-LSTM** comprises the offline version of our extended car-following model, JTPG-LSTM, formulated earlier in Section 3.2 and MOBIL as its lane-changing model.

(4) **Mobility model 4 - Dynamic JTPG-LSTM** is the online version of Mobility model 3 that involves dynamic JTPG-LSTM with online learning (refer to Section 3.3) and MOBIL.

A total of 30 independent simulation runs were carried out. Due to the deterministic nature of our simulation model, these runs were only helpful towards the average wall-clock time measurement used for the feasibility assessment. If the underlying simulation model is stochastic in nature, it is an interesting direction for future works to incorporate data assimilation techniques [13, 45, 51] in the proposed framework in order to handle uncertainty in terms of both data and model.

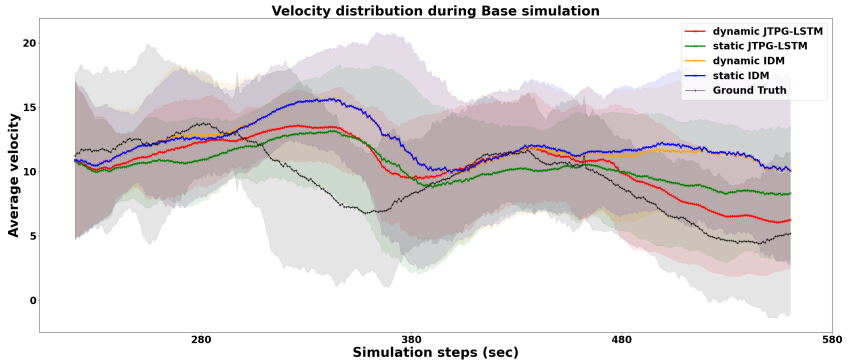

Fig. 7. Velocity distribution during real-time simulation

4.6.1 ***Macro-level Assessment***. After collecting the simulation output statistics, we compared simulated velocity distributions with respect to ground-truth data distribution. This can help assess whether mobility models can reproduce the same variability of driving patterns observed in the physical world and hence, it is also commonly used in existing studies [25, 44, 54]. The real-time simulation experiment results are shown in Figure 7. The solid lines indicate the average velocity of

vehicles in the simulation zone while the shaded regions/bands show the corresponding 95 percent confidence interval. When using pure physics-based models (both static IDM and dynamic IDM), the simulated trends deviate from the actual observed trend noticeably in two simulation periods: *Period 1* - around 300-380 and *Period 2* - around 480-560. In both periods, they tend to overestimate the velocity of the simulated vehicles. Although dynamic IDM is an online model, the impact of online learning seems to be insignificant for IDM because its simulated trend is found to be nearly identical to that of static IDM. This might be due to two reasons – 1) IDM has low model complexity since it only uses five constant parameters in its model formulation, and 2) the feasibility region becomes narrower and limited due to the boundary values defined in the optimization process for keeping its parameters within reasonable ranges. As a result, its parameter values did not change that much during online learning.

Based on the figure, it can be seen that both static and dynamic JTPG-LSTM perform better than pure physics-based models. However, they also overestimated the vehicle velocity in *Period 1*, but this was not the case for dynamic LSTM (a pure data-driven model) developed in our previous work [30]. The main reason for this overestimation is that in this paper, we did not inject any actual ground-truth data at all to the vehicles travelling in the simulation zone. Conversely, in our previous work, when a vehicle entered the simulation zone with insufficient trajectory length (less than 10 seconds), the actual observed vehicle speed was used for the movement of that particular vehicle. In this paper, for unbiased performance comparison among different models, no actual vehicle speed was used , but instead, only the IDM model (out of JTPG-LSTM) was used to predict vehicle acceleration in the case of insufficient trajectory length. This issue can be resolved by modelling LSTM such that it can handle dynamically variable input trajectory length.

Finally, as in *Period 2*, it is evident that dynamic JTPG-LSTM was able to adapt to the observed trend unlike other models. Unlike dynamic JTPG-LSTM, the performance of static JTPG-LSTM had degraded considerably in this period because it did not employ online learning to adjust its parameters. On the contrary, dynamic JTPG-LSTM benefits from the online learning process that dynamically updates its parameters so that its average velocity trend could follow closer to that of the ground-truth data. Therefore, it could be concluded that online learning is still an effective technique for JTPG-LSTM for capturing real-world driving patterns that are dynamically changing through time. It is also highly probable that this gap between the simulated and observed trends will become narrower as the real-time simulation runs for a longer period of time.

4.6.2 ***Qualitative Assessment.*** The simulation snapshot captured at the time step 555 (5 seconds before the simulation end) can be seen in Figure 8. The number of simulated vehicles on the road observed when using dynamic JTPG-LSTM is the closet to that of the ground-truth situation. Although static JTPG-LSTM is an offline model, it also showed relatively competitive results as compared to its dynamic counterpart. This could be due to the improved generalisation capability of our extended model in tackling unseen scenarios. Only sparse simulated traffics were observed when using pure physics-based models due to their velocity overestimation causing more vehicles to arrive at the end of the road more quickly than expected.

Table 4. Velocity trajectory deviation error for the (real-time) base simulation

Model	Static IDM	Dynamic IDM	Static JTPG-LSTM	Dynamic JTPG-LSTM
VTDE	5.3015	5.2811	4.4549	**4.1699**

4.6.3 ***Quantitative Assessment.*** In this section, the performance of each model is assessed quantitatively by using the *VTDE* defined in Equation 21. According to Table 4, the best performance

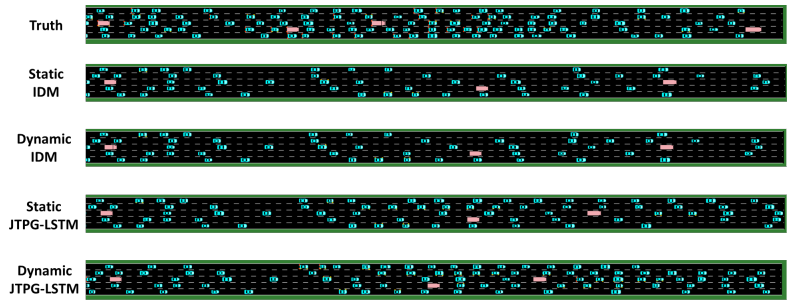

Fig. 8. Base simulation snapshot captured at the simulation step 555 when using different models

is achieved by dynamic JTPG-LSTM among all mobility models. When compared against static JTPG-LSTM, there is an error decrease of 6.4% and this improvement could be attributed to online learning. The base simulation performance when using static IDM is the default performance that could be achieved without using our proposed model. Compared to this default baseline, dynamic JTPG-LSTM has reduced the *VDTE* by 21.34%. Hence, it is an effective model for capturing dynamic car-following behaviours both accurately and safely. All mobility models were observed to be collision-free during the base simulation.

4.6.4 ***Feasibility Assessment.*** The average wall-clock time (WCT) taken at each simulation step when using different mobility models is shown in Figure 9. The total duration measured (in milliseconds) at a given simulation step is incurred by: 1) online parameter learning/update, 2) advancing simulation that involves vehicle agent state updates based on lane-changing and car-following prediction results, and 3) other simulation overheads such as addition and removal of vehicles, etc. When using JTPG-LSTMs, according to their average WCT trends, there is a strong correlation between the vehicle density and the time taken for each simulation step. Even at the most congested period (around the simulation time 550), it only took less than 0.4 second to execute one simulation step for both JTPG-LSTMs. The additional delay introduced by online learning for dynamic JTPG-LSTM was merely within 10-14 milliseconds. It is important to note that JTPG-LSTM has significantly higher model complexity than IDM since the former has a total of 976 parameters (971 from LSTM and 5 from IDM respectively) while the latter only has five parameters. Consequently, the base simulation runs much faster when using the two IDMs. Nevertheless, this has proved the feasibility of our dynamic data-driven simulation system that could run in synchronization with the physical system by using 1 second time resolution. Future works should consider network latency for more practical implementation of real-world scenarios.

4.7 Just-in-time What-if Simulation

Multiple JIT simulations were triggered at the time step 360 to assess which mobility model could provide the best simulation accuracy in terms of short-term forecasting. They were simulated until the time step 560 for a total duration of 3.33 min (see Figure 5(b)). Although online learning is still being executed in the base simulation (which continuously runs in pace with the physical system), it could not be performed in JIT simulation branches because they are used for offline short-term forecasting ahead of the physical system. Thus, both dynamic JTPG-LSTM and dynamic IDM were calibrated with online data only up to the time step 360. The following assumptions were made for this experiment:

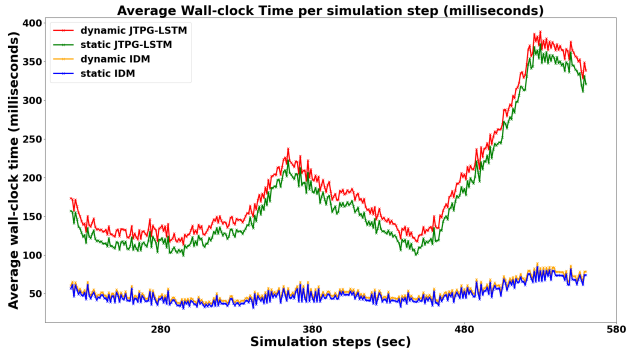

Fig. 9. Average wall-clock time taken for each simulation step during the (real-time) base simulation

(1) The same test data (used for the real-time simulation) was used for JIT simulation. In practice, the required test data cannot be obtained beforehand and hence, different sets of data should be used to conduct JIT what-if simulation.
(2) The JIT simulation was not implemented in distributed manner as it should be according to our framework. The efficient implementation of the system using distributed and parallel simulation is left as future works.

4.7.1 ***Macro-level Assessment.*** In Figure 10, JIT simulation results when using different mobility models are shown. Since online learning was not performed during the simulation, dynamic JTPG-LSTM's performance had degraded. Both static and dynamic JTPG-LSTM underestimated the observed vehicle velocity in the simulation period 400-460 while the pure physics-based IDMs surprisingly performed very well during that period. This might be a weakness in our model formulation, since the final output of JTPG-LSTM is always the lesser value between the LSTM and IDM outputs to guarantee longitudinal collision-free property (see Equation 11). As for dynamic JTPG-LSTM, this underestimation might be exacerbated by the downward trend of observed velocity near the period before the JIT simulation was triggered. It encourages the updated parameters to lean towards the tendency to predict lower velocity values. However, from the simulation step 500 to 560 (which is the most congested period), dynamic JTPG-LSTM outperformed all other models. The short-term forecasting performance of all mobility models will be quantitatively assessed in the next section.

4.7.2 ***Quantitative Assessment.*** According to the experiment results shown in Table 5, the *VDTE* signifies the superior performance of dynamic JTPG-LSTM over others. This is because starting around the simulation step 470, other mobility models overestimated the velocity of simulated vehicles making their trip duration shorter than the actual trip duration. Conversely, the JIT simulation results given by dynamic JTPG-LSTM are, in fact, closer to the observation data and thus, its *VDTE* over the entire trip duration is lower than the rest. Regarding static IDM as our default baseline model, the *VDTE* has been reduced by 16.48% with static JTPG-LSTM and 27.16% with dynamic JTPG-LSTM respectively. Because the parameters of dynamic JTPG-LSTM were updated with online data in the base simulation (until the simulation step 360), it has certain advantages over other mobility models in terms of the simulation accuracy. As for safe driving aspect, all mobility models were collision-free during their respective JIT simulation. Based on

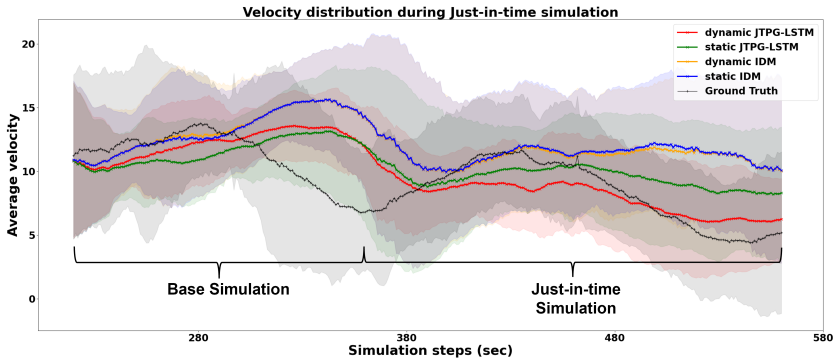

Fig. 10. Velocity distribution during JIT what-if simulation

these experiment results, dynamic JTPG-LSTM has still turned out to be the best model for offline JIT short-term forecasting.

Table 5. Velocity trajectory deviation error for JIT what-if simulation

Model	Static IDM	Dynamic IDM	Static JTPG-LSTM	Dynamic JTPG-LSTM
VTDE	5.7533	5.7173	4.805	**4.1905**

5 CONCLUSIONS

In this paper, a new car-following model, JTPG-LSTM, is developed by coupling the data-driven LSTM and the physics-based IDM on a deeper level. The model is formulated to guarantee longitudinal collision-free car-following property while maintaining desirable simulation accuracy. This is achieved by – 1) learning the parameters of the two models jointly with physics loss and data loss during the training stage, and 2) by setting the safe acceleration upper bound during the deployment stage. In addition to that, JTPG-LSTM is equipped with online learning to capture dynamically changing car-following behaviours. It is then integrated to our dynamic data-driven simulation system that comprises a real-time base simulation. At the same time, it also allows efficient parallel instantiation of multiple just-in-time (JIT) simulation branches for what-if analysis. Extensive experiments were conducted to demonstrate the advantages of the proposed model from both modelling and simulation perspectives. According to the simulation results, as compared against the pure physics-based IDM, the velocity trajectory deviation error has been reduced by 21.34% for the base simulation and 27.16% for the JIT simulation when using dynamic JTPG-LSTM.

Future works that are recommended in the above sections include: 1) testing the proposed hybrid modelling approach using different types of physics-based or data-driven car-following models; 2) extending the LSTM model with the capability of handling variable input trajectory length; 3) balancing physics loss and data loss (during online learning) based on the quality of the real-time data; 4) coupling the proposed model with data-driven lane-changing models to achieve more benefits from real-world data; and 5) handling uncertainty by extending the framework with data assimilation techniques. One limitation of the current hybrid model is that it can only guarantee longitudinal collision-free property. As to prevent collisions with vehicles from adjacent lanes (either under cut-in or lane-changing scenarios), it relies heavily on the lane-changing model which prevents a certain vehicle from changing lane (or cutting in) if there is insufficient safe gap distance. Although this model assumption may be valid in high-way traffics where lane-changing is less

frequent, it should be modelled more appropriately in a study area where lane-changing occurs more frequently or abruptly. Hence, future works should further test the robustness of our model under a variety of traffic conditions.

Another interesting research direction is to extend our simulation system for large-scale simulation studies. As shown in our feasibility assessment, in the most congested traffic (about 380 vehicles on the highway road of length 670 m), it takes about 400 milliseconds (in wall-clock time) to complete one simulation step. Since the simulation system is required to run in synchronization with the physical system, total computational time should be kept less than the choice of time resolution (defined as 1 sec in this paper). This requirement can become more challenging in the context of urban traffic simulation since it involves more complicated simulation scenarios such as interaction among different agents (e.g., pedestrians, traffic lights, cyclists, etc.), leading to more computational overheads to execute a single simulation step. Therefore, in future works, parallel and distributed computing techniques need to be incorporated for the simulation system scalability. Furthermore, since there could be interaction among vehicles and pedestrians (or other road participants), more fine-grained vehicular mobility models need to be developed to capture realistic agent interaction behaviours. Lastly, in the urban traffic, the online learning process for updating simulation model parameters may not be as straightforward as high-way traffic due to several factors including incomplete or missing trajectory data caused by limited sensor deployment and repeated traffic state changes produced by traffic light signals. To this end, the interplay among simulation modelling, machine learning, and parallel and distributed computing is important to ultimately realize the notion of Digital Twin City.

6 ACKNOWLEDGEMENTS

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] Meta (a.k.a. Facebook). 2021. Libtorch. https://pytorch.org/cppdocs/.

[2] Saleh Albeaik, Alexandre Bayen, Maria Teresa Chiri, Xiaoqian Gong, Amaury Hayat, Nicolas Kardous, Alexander Keimer, Sean T McQuade, Benedetto Piccoli, and Yiling You. 2021. Limitations and Improvements of the Intelligent Driver Model (IDM). *arXiv preprint arXiv:2104.02583* (2021).

[3] Heiko Aydt, Stephen John Turner, Wentong Cai, and Malcolm Yoke Hean Low. 2009. Research issues in symbiotic simulation. In *Proceedings of the 2009 winter simulation conference (WSC)*. IEEE, 1213–1222.

[4] M Bando, K Hasebe, A Nakayama, A Shibata, and Y Sugiyama. 1994. Structure stability of congestion in traffic dynamics. *Japan Journal of Industrial and Applied Mathematics* 11, 2 (1994), 203–223.

[5] Huikun Bi, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng. 2016. A data-driven model for lane-changing in traffic simulation.. In *Symposium on Computer Animation*. 149–158.

[6] Prashant Shridhar Bokare and Akhilesh Kumar Maurya. 2017. Acceleration-deceleration behaviour of various vehicle types. *Transportation research procedia* 25 (2017), 4733–4749.

[7] Chenyi Chen, Li Li, Jianming Hu, and Chenyao Geng. 2010. Calibration of MITSIM and IDM car-following model based on NGSIM trajectory datasets. In *Proceedings of 2010 IEEE International Conference on Vehicular Electronics and Safety*. IEEE, 48–53.

[8] Benjamin Coifman and Lizhe Li. 2017. A critical evaluation of the Next Generation Simulation (NGSIM) vehicle trajectory dataset. *Transportation Research Part B: Methodological* 105 (2017), 362 – 377. https://doi.org/10.1016/j.trb.2017.09.018

[9] J. Colyar and J. Halkias. 2007. *US highway 101 dataset.* Technical Report FHWA-HRT-07-030. Federal Highway Admin. (FHWA), Washington, DC, USA.

[10] Anatoli Djanatliev and Reinhard German. 2013. Prospective healthcare decision-making by combined system dynamics, discrete-event and agent-based simulation. In *2013 winter simulations conference (WSC)*. IEEE, 270–281.

[11] Peter G Gipps. 1981. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological* 15, 2 (1981), 105–111.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Xiaolin Hu and Peisheng Wu. 2019. A data assimilation framework for discrete event simulations. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 29, 3 (2019), 1–26.

[14] Ling Huang, Hengcong Guo, Ronghui Zhang, Haiwei Wang, and Jianping Wu. 2018. Capturing drivers' lane changing behaviors on operational level by data driven methods. *IEEE Access* 6 (2018), 57497–57506.

[15] Ling Huang, Hengcong Guo, Ronghui Zhang, Dezong Zhao, and Jianping Wu. 2020. A data-driven operational integrated driving behavioral model on highways. *Neural Computing and Applications* 32, 16 (2020), 13017–13033.

[16] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. 2017. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering* 29, 10 (2017), 2318–2331.

[17] Arne Kesting and Martin Treiber. 2008. How reaction time, update time, and adaptation time influence the stability of traffic flow. *Computer-Aided Civil and Infrastructure Engineering* 23, 2 (2008), 125–137.

[18] Arne Kesting, Martin Treiber, and Dirk Helbing. 2007. General lane-changing model MOBIL for car-following models. *Transportation Research Record* 1999, 1 (2007), 86–94.

[19] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*. PMLR, 5637–5664.

[20] Stefan Krauss. 1998. Microscopic modeling of traffic flow: investigation of collision free vehicle dynamics. *Forschungsbericht - Dtsch. Forschungsanstalt fuer Luft - und Raumfahrt e.V.* 98-8 (1998).

[21] Lauri Lättilä, Per Hilletofth, and Bishan Lin. 2010. Hybrid simulation models–when, why, how? *Expert systems with applications* 37, 12 (2010), 7969–7975.

[22] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner. 2018. Microscopic Traffic Simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2575–2582. https://doi.org/10.1109/ITSC.2018.8569938

[23] Malcolm Yoke Hean Low, Kong Wei Lye, Peter Lendermann, Stephen John Turner, Reman Tat Wee Chim, and Surya Hadisaputra Leo. 2005. An agent-based approach for managing symbiotic simulation of semiconductor assembly and test operation. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. 85–92.

[24] Lijing Ma and Shiru Qu. 2020. A sequence to sequence learning based car-following model for multi-step predictions considering reaction delay. *Transportation research part C: emerging technologies* 120 (2020), 102785.

[25] Mohammed Yazan Madi. 2016. Investigating and Calibrating the Dynamics of Vehicles in Traffic Micro-simulations Models. *Transportation Research Procedia* 14 (2016), 1782 – 1791. https://doi.org/10.1016/j.trpro.2016.05.144 Transport Research Arena TRA2016.

[26] Andrey Malinin, Neil Band, German Chesnokov, Yarin Gal, Mark JF Gales, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, et al. 2021. Shifts: A dataset of real distributional shift across multiple large-scale tasks. *arXiv preprint arXiv:2107.07455* (2021).

[27] Zhaobin Mo, Rongye Shi, and Xuan Di. 2021. A physics-informed deep learning paradigm for car-following models. *Transportation research part C: emerging technologies* 130 (2021), 103240.

[28] Htet Naing. 2021. Supplementary Materials: Data-driven Microscopic Traffic Modelling and Simulation using Dynamic LSTM. https://github.com/Javelin1991/dynamic_traffic_simulation.

[29] Htet Naing. 2021. Supplementary Materials: Dynamic Data-driven Microscopic Traffic Simulation using Jointly Trained Physics-guided LSTM. https://github.com/Javelin1991/dynamic_traffic_simulation_with_jtpg_lstm.

[30] Htet Naing, Wentong Cai, Nan Hu, Tiantian Wu, and Liang Yu. 2021. Data-driven Microscopic Traffic Modelling and Simulation using Dynamic LSTM. In *Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. 1–12.

[31] Bhakti Stephan Onggo, Navonil Mustafee, Andi Smart, Angel A Juan, and Owen Molloy. 2018. Symbiotic simulation system: Hybrid systems model meets big data analytics. In *2018 Winter Simulation Conference (WSC)*. IEEE, 1358–1369.

[32] Sakda Panwai and Hussein Dia. 2007. Neural agent car-following models. *IEEE Trans. Intell. Transp. Syst.* 8, 1 (2007), 60–70. https://doi.org/10.1109/TITS.2006.884616

[33] Vasileia Papathanasopoulou and Constantinos Antoniou. 2015. Towards data-driven car-following models. *Transportation Research Part C: Emerging Technologies* 55 (2015), 496 – 509. https://doi.org/10.1016/j.trc.2015.02.016 Engineering and Applied Sciences Optimization (OPT-i) - Professor Matthew G. Karlaftis Memorial Issue.

[34] Vasileia Papathanasopoulou and Constantinos Antoniou. 2018. Flexible car–following models for mixed traffic and weak lane–discipline conditions. *European transport research review* 10, 2 (2018), 1–14.

[35] M. Pourabdollah, E. Bjärkvik, F. Fürer, B. Lindenberg, and K. Burgdorf. 2017. Calibration and evaluation of car following models using real-world driving data. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 1–6. https://doi.org/10.1109/ITSC.2017.8317836

[36] Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer. 2009. *Dataset shift in machine learning*. Mit Press.

[37] MN Sharath and Nagendra R Velaga. 2020. Enhanced intelligent driver model for two-dimensional motion planning in mixed traffic. *Transportation Research Part C: Emerging Technologies* 120 (2020), 102780.

[38] Rongye Shi, Zhaobin Mo, Kuang Huang, Xuan Di, and Qiang Du. 2021. A Physics-Informed Deep Learning Paradigm for Traffic State and Fundamental Diagram Estimation. *IEEE Transactions on Intelligent Transportation Systems* (2021).

[39] Tie-Qiao Tang, Yong Gui, Jian Zhang, and Tao Wang. 2020. Car-Following Model Based on Deep Learning and Markov Theory. *Journal of Transportation Engineering, Part A: Systems* 146, 9 (2020), 04020104.

[40] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical review E* 62, 2 (2000), 1805.

[41] Martin Treiber and Venkatesan Kanagaraj. 2015. Comparing numerical integration schemes for time-continuous car-following models. *Physica A: Statistical Mechanics and its Applications* 419 (2015), 183–195.

[42] Martin Treiber and Arne Kesting. 2013. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg* (2013), 983–1000.

[43] Martin Treiber and Arne Kesting. 2017. The intelligent driver model with stochasticity-new insights into traffic flow oscillations. *Transportation research procedia* 23 (2017), 174–187.

[44] F. Viti, S. P. Hoogendoorn, H. J. van Zuylen, I. R. Wilmink, and B. van Arem. 2008. Speed and acceleration distributions at a traffic signal analyzed from microscopic real and simulated data. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*. 651–656. https://doi.org/10.1109/ITSC.2008.4732552

[45] Minghao Wang and Xiaolin Hu. 2015. Data assimilation in agent based simulation of smart environments using particle filters. *Simulation Modelling Practice and Theory* 56 (2015), 36–54.

[46] Xiao Wang, Rui Jiang, Li Li, Yilun Lin, Xinhu Zheng, and Fei Yue Wang. 2018. Capturing Car-Following Behaviors by Deep Learning. *IEEE Trans. Intell. Transp. Syst.* 19, 3 (2018), 910–920. https://doi.org/10.1109/TITS.2017.2706963

[47] Xiao Wang, Rui Jiang, Li Li, Yi-Lun Lin, and Fei-Yue Wang. 2019. Long memory is important: A test study on deep-learning based car-following model. *Physica A: Statistical Mechanics and its Applications* 514 (2019), 786 – 795. https://doi.org/10.1016/j.physa.2018.09.136

[48] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. 2020. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919* 1, 1 (2020), 1–34.

[49] Dong Fan Xie, Zhe Zhe Fang, Bin Jia, and Zhengbing He. 2019. A data-driven lane-changing model based on deep learning. *Transp. Res. Part C Emerg. Technol.* 106, June (2019), 41–60. https://doi.org/10.1016/j.trc.2019.07.002

[50] Qi Xin, Rui Fu, Wei Yuan, Qingling Liu, and Shaowei Yu. 2018. Predictive intelligent driver model for eco-driving using upcoming traffic signal information. *Physica A: Statistical Mechanics and its Applications* 508 (2018), 806–823.

[51] Haidong Xue, Feng Gu, and Xiaolin Hu. 2012. Data assimilation using sequential Monte Carlo methods in wildfire spread simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 22, 4 (2012), 1–25.

[52] Da Yang, Liling Zhu, Yalong Liu, Danhong Wu, and Bin Ran. 2018. A novel car-following control model combining machine learning and kinematics models for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems* 20, 6 (2018), 1991–2000.

[53] Jian Zhang and Abdelkader El Kamel. 2018. Virtual traffic simulation with neural network learned mobility model. *Advances in Engineering Software* 115 (2018), 103 – 111. https://doi.org/10.1016/j.advengsoft.2017.09.002

[54] Xiaohui Zhang, Jie Sun, Xiao Qi, and Jian Sun. 2019. Simultaneous modeling of car-following and lane-changing behaviors using deep learning. *Transportation Research Part C: Emerging Technologies* 104 (2019), 287 – 304. https://doi.org/10.1016/j.trc.2019.05.021

[55] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. 2019. Big Data Analytics in Intelligent Transportation Systems : A Survey. 20, 1 (2019), 383–398.

[56] Meixin Zhu, Xuesong Wang, and Yinhai Wang. 2018. Human-like autonomous car-following model with deep reinforcement learning. *Transportation research part C: emerging technologies* 97 (2018), 348–368.

[57] Meixin Zhu, Yinhai Wang, Ziyuan Pu, Jingyun Hu, Xuesong Wang, and Ruimin Ke. 2020. Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving. *Transportation Research Part C: Emerging Technologies* 117 (2020), 102662.

[58] Fang Zong, Meng Wang, Ming Tang, Xiying Li, and Meng Zeng. 2021. An Improved Intelligent Driver Model Considering the Information of Multiple Front and Rear Vehicles. *IEEE Access* 9 (2021), 66241–66252.