Data-driven Microscopic Traffic Modelling and Simulation using Dynamic LSTM

Htet Naing
htetnain001@e.ntu.edu.sg
Nanyang Technological University
Singapore

Wentong Cai
aswtcai@ntu.edu.sg
Nanyang Technological University
Singapore

Nan Hu
nan.h@alibaba-inc.com
Alibaba Cloud
Singapore

Tiantian Wu
qiutian.wtt@alibaba-inc.com
Alibaba Cloud
China

Liang Yu
liangyu.yl@alibaba-inc.com
Alibaba Cloud
China

ABSTRACT

With the increasing popularity of Digital Twin, there is an opportunity to employ deep learning models in symbiotic simulation system. Symbiotic simulation can replicate multiple what-if simulation instances from its real-time reference simulation (base simulation) for short-term forecasting. Hence, it is a useful tool for just-in-time decision making process. Recent trends on symbiotic simulation studies emphasize on its combination with machine learning. Despite its success and usefulness, very few works focus on application of such a hybrid system in microscopic traffic simulation. Existing application of machine (deep) learning models in microscopic traffic simulation is confined to either predictive analysis or offline simulation-based prescriptive analysis. Thus, there is also lack of work on updating parameters of a deep learning model dynamically for real-time traffic simulation. This is necessary if the learning-based model is to be used as part of the base simulation so that "Just-in-time (JIT)" what-if simulation initialized from the model can make better short-term forecasts. This paper proposes a data-driven modelling and simulation framework to dynamically update parameters of Long Short-term Memory (LSTM) for JIT microscopic traffic simulation. Extensive experiments were carried out to demonstrate its effectiveness in terms of more accurate short-term forecasting than other baseline models.

CCS CONCEPTS

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

KEYWORDS

just-in-time simulation; real-time simulation; symbiotic simulation; digital twin; data-driven modelling; deep learning; online learning

SIGSIM-PADS '21, May 31-June 2, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8296-0/21/05…$15.00
https://doi.org/10.1145/3437959.3459258

ACM Reference Format:
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 (SIGSIM-PADS '21), May 31-June 2, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3437959.3459258

1 INTRODUCTION

Conventional simulation are usually used for medium-term or long-term forecasting. Hence, they are suitable for situations where decision making process is not urgent [23]. However, when an operational decision has to be made urgently and requires to run a quick simulation to address the on-going situation of a physical system, traditional simulation may not be ideal. Hence, a simulation system that can offer efficient and effective short-term forecasting is required in this case. Symbiotic simulation system has the capability to meet this requirement. It consists of a real-time simulation, that represents a Digital Twin, running in synchronization with a physical system. Such a real-time simulation is usually referred to as "base" or "reference" simulation in the context of symbiotic simulation [2, 19]. Like real-time simulation and dynamic data-driven application system, it incorporates real-time sensor measurement data to the base simulation to produce more accurate simulation results. Unlike those simulation, symbiotic simulation also involves multiple what-if simulation (WIS) that could be replicated from the base simulation for short-term forecasting. Possible alternative scenarios can then be studied using WIS so that more informed decision can be made just in time. Since the base simulation model is always updated with online data, WIS instances that are initialized from the base simulation just before short-term forecasting, are able to produce more accurate results than conventional simulation. Thus, symbiotic simulation system has been an efficient and effective tool to run "Just-in-time (JIT)" what-if simulation for proactive decision making. It has been successfully used in many real-world applications of several industries including manufacturing, transportation, information technology, healthcare and so on [23].

Recently, there has been growing interest in combining symbiotic simulation with machine learning to build a hybrid simulation system [23]. Such a hybrid system can potentially make even more

accurate forecasts for better prescriptive analytics. In a recent webinar hosted by AnyLogic [21], it has showcased the benefits of using machine learning, as a component of a simulation system. In this context, common application of machine learning is to model more realistic behaviours of certain agents that are involved in the simulation. This is because a machine learning model can learn from real-world data and are more flexible to incorporate more input features than rule-based or mathematical models. Despite its usefulness, very few works focus on the application of such a dynamic data-driven hybrid system to microscopic traffic simulation.

Due to the rise of big data analytic in Intelligent Transportation Systems [39], increasing number of researchers have adopted machine learning approaches such as classic machine learning [24, 25] and deep learning [34–38] to implement microscopic traffic models. These learning-based techniques are used to model either car-following or lane-changing behaviours or both. Those models that use recurrent neural networks such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) show superior performance over both classic machine learning models and equation-based models. In literature, application of these models is confined to either predictive analysis or offline simulation-based prescriptive analysis. The usage of real-world data in these works is limited to offline model training and evaluation or scenario initialization. Furthermore, if a learning-based model is to be used as a component of symbiotic simulation, it should be able to learn from online data so that the base simulation can closely reflect the physical system. Only then, JIT what-if simulation that are branched out from the base simulation can produce more accurate short-term prediction results without any error propagation. However, there is lack of research works that provide potential solutions to resolve these issues.

This paper aims to address the above-mentioned limitations by providing a dynamic data-driven modelling and simulation framework with a case study on microscopic highway traffic simulation. Our framework could be used to conduct JIT what-if simulation for short-term forecasting that could facilitate just-in-time decision making process. Our main contributions are as follows:

(1) a LSTM-based car-following model is developed using offline traffic data.
(2) a data-driven framework is proposed to dynamically update LSTM model parameters with online traffic data for "Just-in-time (JIT)" microscopic traffic simulation.
(3) an extensive set of experiments were conducted to demonstrate the advantages of the proposed framework.

The rest of the paper is organized as follows: Section 2 discusses related works on data-driven microscopic traffic models, online machine learning, real-time simulation and symbiotic simulation. In Section 3, we present our proposed data-driven modelling and simulation framework that involves dynamic LSTM. Section 4 analyzes our experiment results for both base simulation and JIT simulation when using different models. Finally, Section 5 concludes the paper with future research recommendation.

2 RELATED WORKS

2.1 Data-driven Microscopic Traffic Models

Microscopic traffic models are composed of (1) car-following models, and (2) lane-changing models. For accurate traffic forecasting, it is required to capture individual driver's behaviours in a traffic flow model. Traditional well-known car-following models consist of Gipps Model, Intelligent Driver Model (IDM), Krauss Model, etc. These models are based on mathematical or physical equations which render them more restrictive to incorporate new variables that can capture realistic driving behaviors. Furthermore, their parameter calibration process can be time-consuming and tedious as well. Comparison of their performance and more detailed review can be found in [15, 30].

Unlike traditional car-following models, data-driven models that are based on classic machine learning methods are more flexible to incorporate new input features. Hence, more relevant additional inputs can be used for these models to capture driving behaviours [25]. Since these machine learning models can learn from real-world traffic data, they are able to replicate more realistic driving behaviours. In [25], locally weighted regression (loess) was used to compare against Gipps' car-following model in terms of follower speed estimation accuracy. Loess method has achieved lower estimation error across different datasets than Gipps' model. In another work [24], Panwai and Dia used different types of artificial neural networks (ANNs) to estimate follower's speed. Evaluation of ANN-based models was done by integrating the models to a microscopic traffic simulator, AIMSUN. The simulator uses Gipps' car-following model as its default model. During simulation experiment, they overrode leading vehicle speed with ground-truth observation data and used respective models to produce follower's speed. Their results have shown that by integrating a simple ANN to the simulator, error metrics have been reduced by about 17-20% as compared against Gipps' model.

With the advance of big data analytic in Intelligent Transportation Systems (ITS) [39], researchers have adopted high-dimensional modelling techniques such as deep learning to model driving behaviours. Most works are based on recurrent neural networks and its variants such as Long Short-term Memory (LSTM) or Gated Recurrent Unit (GRU). The major advantage of using recurrent neural networks is that they can capture temporal dependency which is important for modelling driving behaviours. Since a driver's decision for a certain action may depend on not only current state and previous state but also states from other past time steps, it is necessary to incorporate all these temporal information together when modelling microscopic driving behaviours. Generally, these works can be categorized into three groups – (1) deep learning-based car-following models [34, 35], (2) deep learning-based lane-changing models [36], and (3) deep learning-based mobility model that captures overall driving behaviours [37, 38]. All these deep learning-based microscopic traffic models have been shown to outperform traditional machine learning methods because they have the internal mechanism to capture temporal dependency from past vehicle states.

Existing application of deep learning-based microscopic traffic models is limited to offline scenario experiment and analysis. In other words, real-world data is only used for either offline model

training and evaluation or initializing simulation parameters. With the increasing popularity of Digital Twin [16] and recent trends in real-time traffic simulation [26], there is a need to integrate these data-driven models to real-time simulation. Moreover, the learning-based model used should be capable of learning from online data so that real-time simulation does not deviate from the actual state of a physical system. Hence, our work aims to fill this gap in the domain of microscopic traffic simulation.

2.2 Online Machine Learning

Online learning is a sub-field under machine learning that focuses on updating a machine learning model with data that are sequentially arriving in an online manner. It is suitable for tasks in real world applications that involve usage of big data. In a recent survey paper [12], a variety of popular online learning algorithms are elaborated with very useful insights. In this section, we briefly describe some related aspects of their review. Online learning techniques can be generally divided into three categories as follows:

(1) **Online supervised learning**: In this category, full feedback information is available for a machine learning model to learn and update its parameters at every time step. In other words, true target output or label is fully available to assess how accurate the model predicted output is.
(2) **Online learning with limited feedback** In this case, only partial feedback information is available for the model at every time step. This means that there is no true target output/label to explicitly tell the model how accurate its prediction is, but instead only partial information about the true output is available.
(3) **Online unsupervised learning** In this scenario, only the sequence of data instances is available at every time step without any additional information about true target output. Hence, there is no feedback available for the model to update and learn and thus, it has to rely on unsupervised learning approaches to perform online learning.

In our problem setting, we make an assumption that at every time step, real-time traffic data is already processed and available at our disposal. Hence, true target output values are explicitly available for our deep learning model to learn and update its internal parameters. Thus, our problem falls under the first category, online supervised learning. It is also concerned with a branch of online learning called incremental batch learning since more than one data instance will be used to update LSTM at each time step [12]. Nevertheless, in this paper, the general term, online learning, is used without distinguishing nuances of different branches of online learning. How LSTM's parameters are updated by using online learning will be discussed in Section 3.3.

2.3 Real-time Simulation

According to [4], simulation can generally be divided into three categories – (1) offline simulation that runs faster than physical system, (2) offline simulation that runs slower than physical system, and (3) real-time simulation that runs in synchronization with physical system. In real-time simulation, computational work that is necessary for simulation such as updating states of simulated agents or estimating their behaviours, updating parameters of models used, etc. should be completed within the interval between two successive simulation steps. Only then, such simulation will be ready to incorporate periodically incoming real-time sensor data and run in pace with its physical system.

Since there is growing popularity in Digital Twin [14, 16], real-time simulation becomes the center of attention again. Digital Twin is a virtual system that can be synchronized with real-time data that comes from its physical counterpart. Different industries are taking advantage of Digital Twin for intelligent decision-making to gain their competitive edge. Recent trends on real-time traffic simulation is discussed in [26]. Real-time simulation of traffic flow involving pedestrians, vehicles and bicycles and their interactions is implemented and applied in real world in the city of Liverpool (NSW, Australia) [3]. In [10], a methodology designed for dynamic data-driven microscopic simulation is used to estimate different performance measures of an arterial road in real-time. As mentioned in [31], "The next exciting step for large-scaled, data-driven, agent-based simulations is to make them live." Therefore, the contribution of our work targets towards that research direction as well.

2.4 Symbiotic Simulation

The concept of symbiotic simulation is closely associated with real-time simulation or dynamic data-driven application system. It means that symbiotic simulation also incorporates real-time sensor measurement data to drive the simulation. However, the key differentiating factor is the emphasis on what-if analysis (WIA) in a symbiotic simulation system [2]. In symbiotic simulation, there is a "base" or "reference" simulation that acts as a Digital Twin of a physical system while multiple offline simulation could be replicated from the base simulation to analyze alternative scenarios for short-term operational decision making [2, 19]. Hence, it is suitable for WIA process that needs to be done efficiently and effectively for just-in-time decision making. Conversely, conventional simulation are more appropriate for medium or long-term forecasting [23] since the time available for decision making is sufficient for the simulation to run and generate required statistics. As mentioned in [23], symbiotic simulation system has been successfully applied in many areas such as manufacturing, drone control, transportation planning, data center monitoring and decision making in healthcare.

Recent works also focus on the integration of artificial intelligence, particularly machine learning as a component of symbiotic simulation system [21, 23]. A machine learning model can then be used to capture behaviours of certain agents involved in the simulation. Offline data is applicable for model training and evaluation while online data can be used for dynamically updating the model parameters. The usage of both data sources will enhance the accuracy of a symbiotic simulation system. Instead of treating simulation and machine learning as mutually exclusive subjects, the interplay between the two could benefit both as highlighted in a recent webinar hosted by AnyLogic [21]. Despite its success and usefulness, very few works focus on application of such a hybrid system in microscopic traffic simulation. For that reason, we aim to utilize it in our proposed framework for "Just-in-time (JIT)" microscopic traffic simulation.

3 DYNAMIC DATA-DRIVEN MICROSCOPIC TRAFFIC MODELLING & SIMULATION

3.1 Overview

In this paper, a dynamic data-driven modelling and simulation framework is proposed for carrying out just-in-time (JIT) what-if simulation. The framework is data-driven in three main aspects since real-world traffic data is used - 1) to develop a LSTM-based car-following model using historical data, 2) to dynamically update the model parameters using online data, and 3) to initialize multiple JIT simulation for short-term what-if analysis. The overview of the framework is illustrated in Figure 1.

3.1.1 ***Static LSTM.*** As mentioned in Section 2.1, recurrent neural networks and its variants such as LSTM and GRU are successful in capturing more accurate driving behaviours by incorporating temporal information in their internal mechanism. LSTM is chosen as our deep learning model that will be used to predict velocity of a vehicle. More information about these models and our selection choice will be discussed in Section 3.2. Before LSTM is used, it has to be pre-trained with historical offline data. However, this pre-trained LSTM is static and hence, it is not ideal for real-time simulation.

3.1.2 ***Base Simulation.*** Base simulation is a real-time simulation that is running in pace with a physical system. LSTM is embedded as a component of the base simulation to capture car-following behaviour. However, in real-time traffic simulation, driving behaviours may change over time depending on the current traffic situation. As a result, overall traffic patterns may turn out to be different from those past patterns observed in offline training data. Therefore, this part of the framework ensures the learning-based model integrated to a simulator is always up-to-date with latest knowledge of the physical world. It can be achieved by using online learning to constantly update the underlying model with sequentially arrived data. Hence, the static LSTM becomes dynamic with online learning process. In this paper, we first take the initiative to update LSTM's parameters with online data for modelling dynamic car-following behaviour. Future works should consider more sophisticated deep learning model that can account for overall driving behaviour including both car-following and lane-changing.

Among open-source microscopic traffic simulators compared in [29], SUMO [18] is the most suitable in our case since it focuses on simulation of microscopic traffic behaviours by modelling each vehicle individually. The dynamic LSTM is integrated with SUMO via its Traffic Control Interface, TraCI, which overrides SUMO's default car-following model (i.e., Krauss model [17]).

3.1.3 ***Just-in-time What-if Simulation.*** The main difference between the usage of just-in-time (JIT) simulation and conventional simulation lies in the urgency of operational decision making process. The former can be used for short-term forecasting to help just-in-time decision making process while the latter is typically used for medium-term or long-term forecasting to facilitate decision making of medium-term or long-term objectives [23]. A base simulation that can realistically capture current state of the physical system can expedite the JIT simulation process. This is because we can readily branch out multiple instances of the base simulation model to carry out different JIT what-if simulation. The simulation model has been also updated with real-time data up to the latest time step and hence, it is more suitable for quick JIT simulation. Without base simulation, we could not achieve this efficiency since several factors could slow down the JIT simulation process such as organizing relevant recent data, calibrating the simulation model with those data, etc. By the time the simulation model is ready to be used, it could have been already late for just-in-time decision making. Our proposed framework will be explained in more details in the following sections.

3.2 Long Short-term Memory (LSTM)

In microscopic traffic simulation domain, LSTM has been successfully applied in recent works [13, 36, 38] and proven that it can capture both realistic and accurate driving behaviours. As discussed in Section 2.1, some works also consider using another variant of RNNs called, Gated Recurrent Unit (GRU) with competitive performance as comparable as LSTM [32, 34, 35]. GRU and LSTM share similar properties and capabilities except that it requires less internal parameters in its architecture than LSTM [34]. Since our goal is to demonstrate the effectiveness of an online deep learning model during real-time simulation, LSTM is selected as our default model choice. Performance comparison between these two models in the context of real-time simulation is left as future work. For LSTM implementation, we use an open-source Python library, Keras [5] to implement our model. According to its documentation, the LSTM layer implementation follows the original paper [11]. Internal mechanism of LSTM is controlled by gates and states and the final output can be calculated based on the equations below.

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

$$i_t = \sigma(W_i s_t + U_i h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma(W_o s_t + U_o h_{t-1} + b_o) \tag{3}$$

$$c_t = (f_t \circ c_{t-1}) + (i_t \circ tanh(W_c s_t + U_c h_{t-1} + b_c)) \tag{4}$$

$$h_t = o_t \circ tanh(c_t) \tag{5}$$

$$y_t = V h_t + b \tag{6}$$

y_t is the final output after going through the last linear layer; *t* and *t-1* denotes 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.

3.2.1 ***Input and Output Features.*** Output of our LSTM network is a follower's velocity of the next time step. Among existing works, different kind of features are used as inputs to LSTM. In this paper, we have conducted experiments to find out which set of features is the most appropriate when estimating the follower's velocity. Movement of a vehicle can be affected by its neighboring vehicles even if a vehicle is said to be only following its immediate leader on the same lane. For instance, consider scenarios where a vehicle is

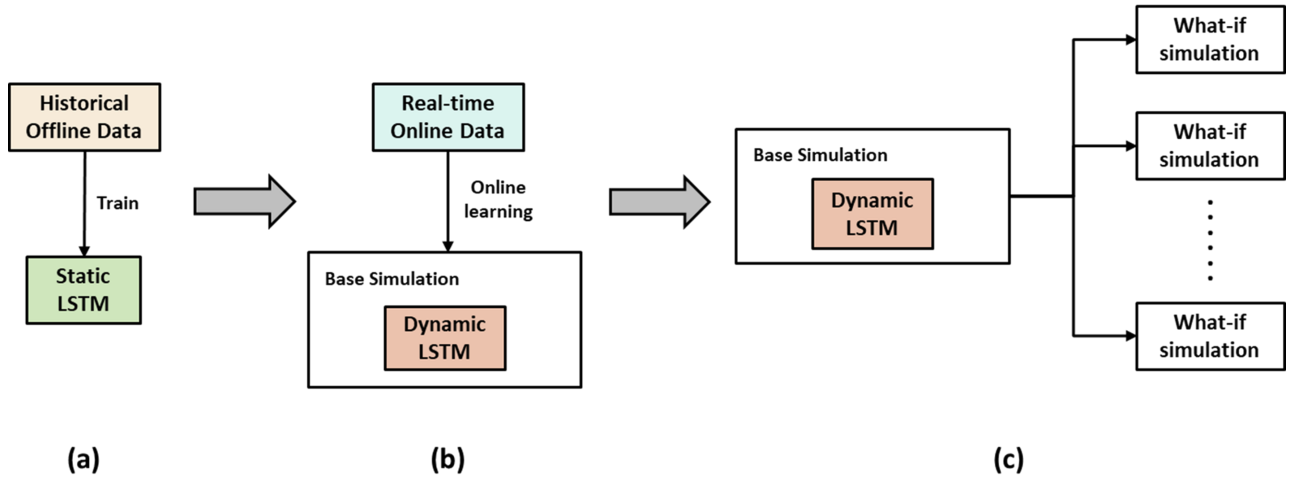

Figure 1: Overview of dynamic data-driven microscopic traffic simulation framework - (a) Static LSTM is trained using offline data, (b) Dynamic LSTM is trained using online data in base simulation, (c) Multiple just-in-time (JIT) what-if simulation are initialized from base simulation for short-term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

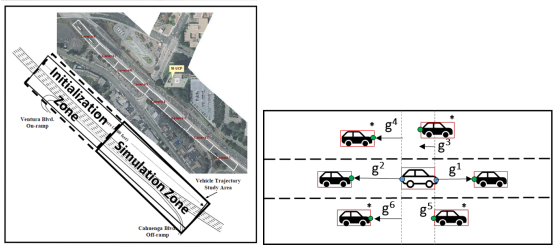

Figure 2: Study area (left) (Photo adapted from [7]) and surrounding gap distances (right)

being tailgated by another vehicle from behind or another vehicle from an adjacent lane is suddenly trying to cut in front of the vehicle. Therefore, longitudinal neighboring gap distances (See Figure 2) are also considered with reference to how SUMO calculates them in order to extract these surrounding features. The experiment results are shown in Section 4 and the final optimal set of input features is:

$$s = \{x, lane, vel, acc, length, g^1, g^2, g^3, g^4, g^5, g^6\} \tag{7}$$

$$g^{\mathrm{i}} = \begin{cases} x_{veh_l} - length_{veh_l} - x_{veh}, & \text{if } veh_l \text{ is present} \\ x_{veh} - length_{veh} - x_{veh_f}, & \text{if } veh_f \text{ is present ; } i \in \{1, ..., 6\} \\ 0, & \text{otherwise} \end{cases}$$

Let veh, $\mathrm{veh_l}$, $\mathrm{veh_f}$ 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; vel is its velocity; acc is its acceleration/deceleration; length is a vehicle's length; $\mathrm{g^i}$ denotes the longitudinal gap distance between veh and another vehicle; 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.

3.2.2 ***Input Trajectory Length.*** The vehicle state vector s in Equation 7 represents the state of a vehicle at a particular time step ($\mathrm{s_t}$). For a given vehicle, its velocity estimation problem can be described as follows.

$$vel_t(\theta) = f(s_{t-n}, s_{t-n-1}, ..., s_{t-1}|\theta) \tag{8}$$

The function f is used to estimate the follower's velocity given vehicle state inputs from past time steps and parameters, θ. LSTM is then used to approximate this function f by learning the optimal values of parameters θ (i.e., weights and biases of the network) from training data. The length of the vehicle state trajectory (n) depends on time step resolution in consideration. According to [34], time resolution choice of 1 second is adopted in this paper for drivers are not able to perform too many actions during a very short period of time. Past trajectory length of 10 seconds is also used as LSTM's inputs in our model according to the systematic empirical study done by [35]. In their work, a GRU neural network using an input trajectory length of 10 seconds is able to produce more realistic commonly observed traffic phenomena than another network using shorter trajectory length.

3.2.3 ***Loss Function.*** Mean squared error between the actual and predicted velocity is used as the loss function during LSTM training process. It is widely used for regression tasks and in [38], it is also applied for simultaneous modeling of car-following and lane-changing behaviours during LSTM training process.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(vel_i^{act} - vel_i^{pred})^2 \tag{9}$$

vel_i^{act} and vel_i^{pred} are actual and predicted velocity in an $\mathrm{i^{th}}$ training data instance.

3.2.4 ***Optimization Method.*** Gradient descent (GD) optimization algorithms are commonly used to minimize a loss function during training process of deep learning models. Based on an empirical study conducted for comparison of different GD techniques [28], RMSprop tends to head off instantly towards the region where the global minimum exists. Since online learning will be performed during the interval between two successive time steps, RMSProp is selected due to its nature of fast convergence, along with the learning rate value of 0.001.

3.2.5 ***Other Hyperparameters.*** In the context of machine learning, hyperparameters are parameters that can be configured through sensitivity analysis or trial and error in order to achieve the possibly optimal performance of a deep learning model. These parameters will not be learnt during model training process and instead, they control how a model should be learnt. Since there are too many possible combinations of these parameters, our hyperparameters were determined first based on a set of optimal parameters reported in [38] and then, through trial and error. We have identified the best set of hyperparameters to be used for LSTM as follows: Dimension of the input layer=11; dimension of the output layer=1; batch size = 16; input trajectory length=10 (sec); number of LSTM layer=1; number of neurons in LSTM layer=10; validation data=20 percent of training data; training epochs=10; loss function=mean squared error; optimization method=RMSProp; learning rate=0.001.

3.3 Dynamic LSTM

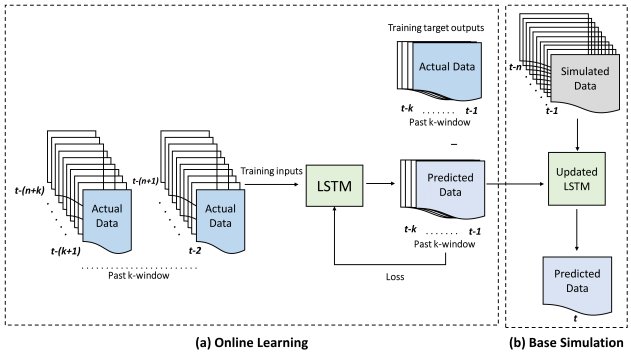

Figure 3: (a) Online learning process of dynamic LSTM; (b) Base simulation using dynamic LSTM

LSTM described in the previous section is static since it is trained using only offline data. If this LSTM is to be used as part of a base simulation model, it should also incorporate newly arrived data instances so that it is updated with the latest knowledge of the physical world. To support real-time traffic simulation, dynamic LSTM is hence developed by integrating LSTM with online learning capability. It is important to note that different optimization algorithms used during online parameter update can affect the model's performance as well as time taken for the update. The algorithm chosen in this paper, RMSProp is a first-order optimization algorithm that use the first order derivative information for online learning. According to [12], second-order online learning algorithms are able to reach the optimization convergence more quickly, but can incur higher computational complexity.

We describe dynamic LSTM's online learning process with reference to Figure 3. During the short interval between two successive time steps, two tasks are executed – (a) to update LSTM with arrived data using online learning, (b) to use the updated LSTM to estimate each vehicle's velocity based on their previously simulated vehicle states. Assume that real-time traffic data has already arrived at time step *t-1*, this information can be used to assess model estimation error between true vehicle velocity and predicted velocity. The resulting error loss will be used as feedback to update the model's parameters using RMSProp optimization algorithm. After that, for each vehicle running in base simulation, the updated LSTM is used to predict their velocity for the next time step *t*. It is important to note that we rely on previously arrived actual data for online learning while we only use simulated vehicle data as inputs to predict next vehicles' velocity in base simulation.

The online learning process is described in Algorithm 1. At the beginning, the parameters θ in LSTM are already updated using offline training data. Additionally, the chosen window size (k) and trajectory length (n) are also defined. The algorithm sets one constraint to ensure the training data have sufficient trajectory length to be used as the LSTM's inputs. At every time step, real-time traffic data that contain actual vehicle states are arriving at our disposal (*line 4*). Then, the training target outputs (See Figure 3(a)) are prepared for updating the LSTM's parameters. In *lines 5-9*, k-most recent output data are collected and stored in D_{output} which contains actual velocity of the vehicles from k-most recent time steps. Then, training input data that are required to estimate the collected D_{output} are prepared in *line 10*. To estimate a vehicle's velocity at *t*, past vehicle states of the vehicle from *t-1* to *t-n* are required. Certain vehicles may not have sufficient past vehicle states (if they have not travelled long enough in the study area). Hence, valid training input data D_{input} are only taken from the vehicles that have sufficient trajectory length (n). In doing so, all valid inputs from k-most recent time steps are also used for estimating vehicles' velocity at the respective time steps. In *lines 11-15*, the estimated output velocity of vehicles are compared against D_{output} to calculate mean squared error loss which is then used to update the existing parameters θ via RMSProp optimization method. The above-mentioned steps are captured in Figure 3(a). After that, as illustrated in Figure 3(b), the LSTM with updated parameters θ_{t+1} is used to predict the next velocity for vehicles running in base simulation by using their previous simulated states of the vehicles as inputs. Since the online learning process is being executed as part of the base simulation, the algorithm terminates at the end of the simulation model time *T*.

3.4 Base Simulation using Dynamic LSTM

In our framework, base simulation is a real-time traffic simulation involving dynamic LSTM that models car-following behaviours. The benefits of having the base simulation are three-fold - 1) It can assess whether the current simulation model is good enough 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's

Algorithm 1: Updating LSTM Parameters Online

Result: LSTM with updated parameters $\hat{\theta}$
Initialization: LSTM, parameters θ, window size k, trajectory length n;
Constraints: C_1(input vehicle trajectory length $\geq$ n);
for t = 1, ..., T **do**
  Data D_t, arrive at t, where actual vehicle states $S_t \in D_t$;
  if $t \geq k$ **then**
    $D_{output} = D_t \cup D_{t-1} \cup ... \cup D_{t-k+1}$;
  else
    $D_{output} = D_t \cup ... \cup D_1$;
  end
  Get valid input data D_{input} w.r.t C_1;
  if D_{input} *exist* **then**
    Predict Y_{output} using $LSTM(D_{input}|\theta_t)$;
    Calculate MSE loss between D_{output} and Y_{output};
    θ_{t+1} = Update θ_t using MSE loss and RMSProp;
  end
end

parameters are constantly updated with online data for better short-term forecasting.

Base simulation is executed as follows. Upon the arrival of real-time traffic data, online learning (See Algorithm 1) is first carried out to dynamically update parameters of LSTM. 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 LSTM with updated parameters is used to estimate next velocity of simulated vehicles (See Figure 3(b)). During execution of the base simulation, all these necessary computational work that are required for online LSTM parameter update, estimating vehicle's velocity, vehicle initialization and removal, SUMO's internal computational overheads, etc. have to be finished within the time duration of less than 1 second.

3.5 Just-in-time What-if Simulation

For JIT what-if simulation, the simulation model that is already calibrated up to the latest time step can be used. Such model is able 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 JIT simulation could be branched out from the base simulation and run faster (since it does not have to wait for real-time data arrival) than the physical system for short-term forecasting of different performance measures. Therefore, necessary operational management decision can be made just in time to address the on-going situation of a physical system in an efficient manner.

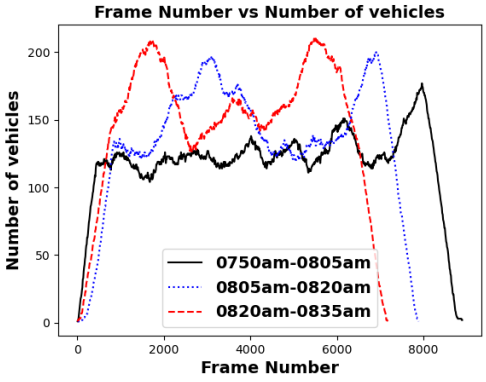

Figure 4: Vehicle density as captured through video frame sequences in each dataset

4 EXPERIMENTS

4.1 Experiment Setup

The experiments were conducted by using a laptop operating on Microsoft Windows with an Intel i7 @ 2.6GHz processor with 8 CPUs and 16GB memory. The popular Next Generation Simulation (NGSIM) US Highway 101 dataset [7] 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 4. The following assumptions were made in the implementation of base simulation that consists of dynamic LSTM.

(1) **Real-time simulation** is only pseudo real-time simulation based on the popular Next Generation Simulation (NGSIM) US Highway 101 dataset [7].
(2) **Simulation coverage** is defined to only consider vehicles that are travelling on five major lanes from lane 1 to lane 5 (See Figure 2). 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 [6].
(3) **Data quality** is assumed to be high with minimal noise involved due to the recent advancement of big data collection and processing technology in Intelligent Transportation Systems [39].
(4) **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.
(5) **Warm-up period** is required for the simulation to reach its stable state. During warm-up period, no vehicle is simulated but instead, all the movement of vehicles are based on actual observation data. In this case, it is similar to using the simulator merely for the animation purpose.

The study area was divided into two zones - 1) Initialization zone and 2) Simulation zone as illustrated in Figure 2. 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 velocity predicted by dynamic LSTM. Simulation output statistics were also collected from this zone for analysis of different performance measures.

4.2 Data Pre-processing

Each dataset has a resolution of 100 millisecond among total 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 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 recording transition, data instances from these periods were filtered out when preparing training data for LSTM.

The *Dataset 1* was used along with 3-fold cross validation (CV) method to test the velocity prediction performance of static LSTM. The pre-processed dataset was divided into three groups. For instance, in the cross validation group CV1, the first group was used as test data while the two remaining groups were used as training data, and vice versa. Each cross validation group has 44,691 training data instances and 22,345 test data instances. Training sequences were shuffled during the LSTM learning process, but test sequences were not shuffled and processed in ascending frame order. Each instance takes the form of input and output to the function given by Equation 8.

4.3 Static LSTM

The evaluation metric for this experiment is based on RMSE among actual ground-truth and predicted velocity for all N training data instances as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(vel_i^{act} - vel_i^{pred})^2} \tag{10}$$

The purpose of this section is to investigate the performance of static LSTM when using different sets of feature. The output target variable is the velocity of a vehicle at next time step *t*. Different sets of input features were tested as below. Each set of input forms a vehicle state at one time step. Since the input trajectory length of 10 (sec) is considered, an input data instance contains past 10 vehicle states from *t-1* to *t-10*.

(1) **Feature set 1 (F1)** : $\{vel\}$
(2) **Feature set 2 (F2)** : $\{x, lane, vel, acc, length\}$
(3) **Feature set 3 (F3)** : $\{\text{F2}\} \cup \{g^1, g^2, g^3, g^4, g^5, g^6\}$
(4) **Feature set 4 (F4)** : $\{\text{F2}\} \cup \{T_g\}$, where $T_g = \sum_{i=1}^{G} g^i$

All feature values were scaled between the range [-1, 1] by using min-max normalization so that their contribution towards the target output are objectively considered without having any bias due to their original range of values. Feature sets 1 and 2 reduce the velocity estimation to univariate and multivariate time-series prediction problems respectively. Feature set 3 extends Feature set 2 by considering neighboring gap distances. Feature set 4 borrows idea from occupancy map pooling, as applied in Social LSTM [1], to combine gap distances. From Table 1, the lowest RMSE is achieved when using F3. This indicates that allowing LSTM to learn importance of features from data by itself can produce more accurate prediction results. Hence, F3 was used as our optimal feature set for both static and dynamic LSTM.

Table 1: Follower's velocity estimation using static LSTM

Metric	RMSE			
Input	**F1**	**F2**	**F3**	**F4**
CV1	0.92	0.8939	0.864	0.8834
CV2	1.0442	0.9886	0.9797	0.9897
CV3	1.0191	1.0387	1.0429	1.1025
AVG	0.9944	0.9737	**0.9622**	0.9919

4.4 Dynamic LSTM

The same evaluation metric (defined in Equation 10) was used to find out how much recent data should be considered during each online parameter update for dynamic LSTM. Different size of past window (k) can affect both LSTM prediction performance and online update duration. Same CV groups (as above) were used for training and testing dynamic LSTM. According to the results shown in Table 2, dynamic LSTM has achieved the best prediction performance via online learning with k=5. It has also reduced the estimation error by 7.97% (from 0.9622 to 0.8855) compared to static LSTM. This shows the advantage of dynamic LSTM over static LSTM when observed traffic patterns are dynamically changing. Online learning duration was measured between every two consecutive time steps when using different values of k. Across all window sizes of k, mean update time is much less than 0.25 sec with a few outliers exceeding over 1 second. It indicates that integration of dynamic LSTM to real-time traffic simulation (with resolution of 1 sec) is feasible. Up to this stage, actual observation data were used as the inputs to both static and dynamic LSTM and their performance have not been tested when employed in the base simulation yet. Thus, next section will assess their performance during the base simulation by comparing against other benchmark models.

Table 2: Follower's velocity estimation using dynamic LSTM

Metric	RMSE				
Past window size (k)	**1**	**5**	**10**	**15**	**20**
CV1	0.8207	0.8185	0.8192	0.8207	0.8223
CV2	0.9379	0.9338	0.9353	0.937	0.9388
CV3	0.9081	0.9043	0.9088	0.9149	0.92
AVG	0.8889	**0.8855**	0.8878	0.8909	0.8937

4.5 Base Simulation

The aim of this section is to select the best mobility model for base simulation. 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 [2]. Furthermore, since JIT what-if simulation are carried out by relying on the base simulation model, a bad base simulation model can propagate its error to JIT simulation as well. Hence, it is required to use a mobility model that can alleviate the above issues.

The *Datasets 1* and *2* were used as offline training data for learning parameters of static LSTM using batch size of 64, training epochs of 20 with the rest of hyperparameters unchanged. Offline parameter calibration of Krauss car-following model was also performed using these two datasets. Dataset 3 was used as test data. During base simulation using test data, dynamic LSTM had made use of periodically arrived real-time data to update its parameters via online learning. Pseudo real-time traffic simulation was set up using the following configuration parameters (See Figure 5): 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).

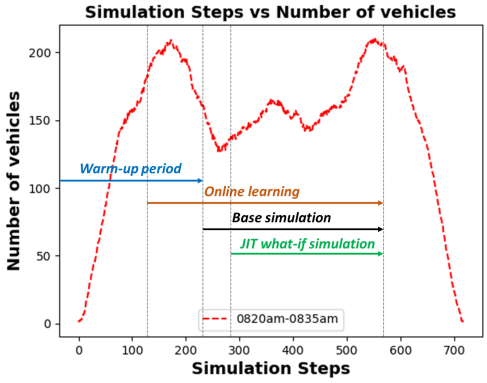

Figure 5: Dynamic data-driven microscopic traffic simulation using Dataset 3

The performance of base simulation when using different mobility models was assessed. Since all car-following models used rely on LC2013 [8] for lane-changing behavior, mobility models will be addressed based on the names of their respective car-following models. More details on Krauss car-following model and its equations are provided in our supplementary materials [22].

(1) **Mobility model 1 - Krauss** consists of SUMO's default Krauss car-following model and LC2013. With reference to [27], a Genetic Algorithm (GA) was used to calibrate Krauss car-following model's parameters. Both optimization variables and objective functions were adapted to suit our case. The model was calibrated based on each vehicle type – motorcycle, passenger car and truck. The calibration procedure, final optimal values and GA parameters used are reported in [22].

(2) **Mobility model 2 - Dynamic Krauss** is an extension of mobility model 1. Dynamic Kruass would make use of periodically arrived real-time data to calibrate its parameters. Dynamic parameter calibration was done during the same period that was allocated for online learning of dynamic LSTM. However, since it usually takes some time for GA to converge, current parameters were updated with newly calibrated set of parameters at every 10 sec (10 time steps). Parameters used in GA for dynamic calibration is also described in [22].

(3) **Mobility model 3 - Static LSTM** is made up of LSTM that is trained using offline data and LC2013.

(4) **Mobility model 4 - Dynamic LSTM** is our proposed mobility model that involves dynamic LSTM with online learning and LC2013.

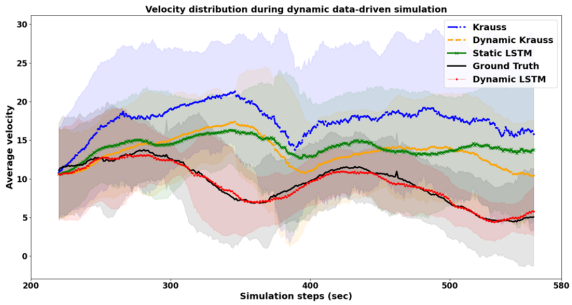

Figure 6: Velocity distribution during real-time simulation

4.5.1 ***Macro-level Performance Assessment.*** After collecting floating car data produced by SUMO as simulation output statistics, we compared simulated velocity distributions with respect to ground-truth observation data. They were calculated based on 10 independent simulation runs by taking arithmetic mean of velocity of all vehicles driving on the road at each time step. It can help assess whether mobility models can reproduce the same variability of driving patterns observed in the physical world and hence, it is commonly used in existing studies as well [20, 33, 38]. The real-time simulation experiment results are shown in Figure 6. Krauss and Static LSTM are offline models and hence, they are not able to adapt to varying traffic patterns observed through time and overestimate vehicles' velocity. Although dynamic Krauss can perform marginally better than static models, its results are still inferior to that of dynamic LSTM. From the results, it can be seen that dynamic LSTM can follow closely the trend of real-world driving patterns that are dynamically changing through time.

From our observations, both ground-truth and simulated distributions were not normal but heavily skewed. However, as empirically studied and suggested in [9, 38], even for heavily skewed distributions, t-tests should be used when large number of samples are available. According to Welch's t-test, only dynamic LSTM can produce statistically equal results with 95% confidence to actual velocity distribution with p-value of 0.3979 that is larger than 0.05. The rest of the mobility models are considered statistically different since their p-values are extremely smaller than 0.05.

4.5.2 ***Qualitative Performance Assessment.*** From Figure 6, we can conclude that all other mobility models except dynamic LSTM overestimate vehicles' velocity during the real-time simulation. Hence, simulated vehicles' trip duration becomes shorter than their actual duration when using these models. To visually check this hypothesis, two test vehicles were randomly selected and their longitudinal position trajectories are plotted in Figure 7. As observed, both of their trips end at the exact same time as the observation data when using dynamic LSTM whereas they end too early when using other mobility models. The simulation snapshot captured at the time step 555 (i.e., 5 sec before the simulation end) can be seen in Figure 8. In the figure, the traffic pattern observed when using dynamic LSTM is close to the ground-truth pattern. Therefore, the above experiment results show that dynamic LSTM is the most suitable model to be used in the base simulation.

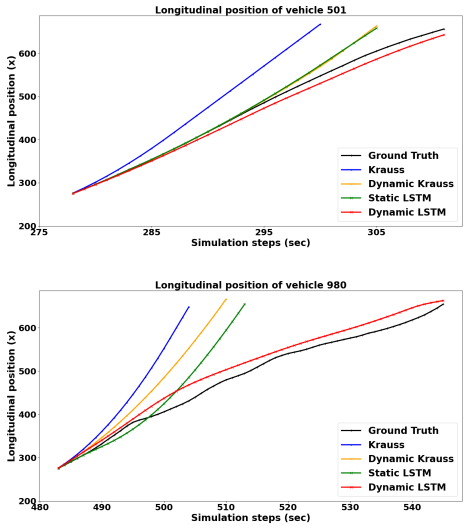

Figure 7: Longitudinal positions of two test vehicles

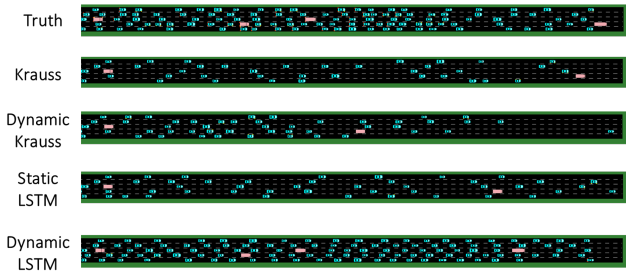

Figure 8: Qualitative comparison of different models

4.6 Just-in-time (JIT) What-if Simulation

The goal of this section is to compare performance of JIT simulation when using different mobility models. The experiment was set up with the same configuration used for base simulation. However, JIT simulation was triggered at the time step 280 and simulated until the time step 560 for a total duration of 4.67 min (See Figure 5). Starting from the time step 280 onward, online learning could not be performed since JIT simulation was used for offline short-term forecasting ahead of the physical system. Thus, both dynamic LSTM and dynamic Krauss were calibrated using online data only up to the time step 280. The following assumptions were made for this experiment:

(1) The same test data (used for 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 the proposed framework. The efficient implementation of the framework using distributed and parallel simulation is left as future work.

4.6.1 ***Macro-level Performance Assessment.*** In Figure 9, JIT simulation results when using different mobility models are shown. All mobility models tend to overestimate vehicle's velocity during JIT simulation. Since online learning was not performed during the simulation, dynamic LSTM's performance also degraded. Nevertheless, the trend of the dynamic LSTM is the closet to that of the observation data among all the models. Offline models such as Krauss and static LSTM even display opposite trends in some periods as compared to the observation data. The result indicates the short-term forecasting performance improvement due to the base simulation model because its parameters had been dynamically updated with online data.

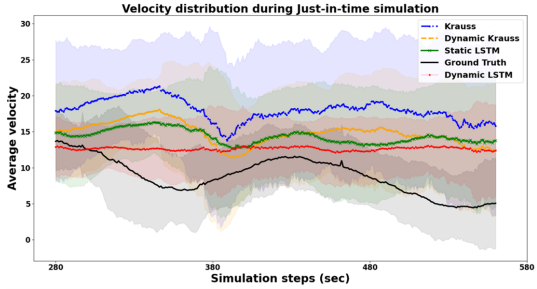

Figure 9: Velocity distribution during JIT what-if simulation

4.6.2 ***Quantitative Performance Assessment.*** The following error metric is defined to quantitatively assess the performance of JIT simulation based on observed and simulated velocity throughout the entire trip of each vehicle.

$$Error = \sqrt{\frac{1}{m}\sum_{j=1}^{m}[\frac{1}{L}\sum_{i=1}^{L}(vel_{ji}^{act} - \mathbf{1}_i[vel_{ji}^{sim}])^2]} \quad (11)$$

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

It can be seen in Table 3 that the new error metric signifies the superior performance of dynamic LSTM over others. This is due to the fact that simulated trip duration of vehicles produced by other mobility models are shorter than the actual trip duration because they overestimate the velocity of following vehicles. Conversely, JIT simulation results given by dynamic LSTM is closer to actual ground-truth observation data and thus, its error in terms of follower velocity estimation over entire trip duration is less than the rest. The JIT simulation result when using offline Krauss car-following model is the default performance that could be achieved without using the proposed framework. With the proposed framework, JIT short-term prediction error has been reduced by 38.2% because dynamic LSTM was updated with online data in base simulation until the time step just before the short-term forecasting.

Table 3: Error comparison among different models used for JIT what-if simulation

Mobility Model	Krauss	Dynamic Krauss	Static LSTM	Dynamic LSTM
Error	9.3585	7.4653	7.313	**5.7837**

5 CONCLUSIONS

In this paper, a LSTM-based car-following model is developed using real-world traffic data to capture realistic driving behaviours. A data-driven traffic simulation framework is also provided to dynamically update the parameters of LSTM with online learning, namely dynamic LSTM. Then, extensive experiments were conducted to demonstrate the effectiveness of the proposed framework for carrying out "Just-in-time (JIT)" microscopic traffic simulation. The experiment results have shown the performance improvement of short-term forecasting due to the proposed framework. According to the JIT simulation results, the average velocity deviation error measured throughout the trip of each vehicle has been reduced by 38.2% when using the proposed framework with dynamic LSTM as compared against the default offline car-following model.

As for future research recommendation, the proposed framework should be tested or extended further using more challenging sensor data sources such as radars, loop detectors rather than video camera. More sophisticated deep learning models that can capture both car-following and lane-changing behaviours should also be considered. This could help improve the short-term forecasting performance further. In doing so, one should also account for trade-off between model complexity and efficiency, especially in the context of dynamic data-driven simulation. It is highly recommended to implement the framework properly using parallel and distributed systems for more effective and efficient JIT what-if simulation. The proposed framework can be also generalized to other applications where the just-in-time decision making process is often involved.

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] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 961–971.

[2] 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.

[3] Johan Barthelemy and Pascal Perez. 2020. Towards Agent-Based Traffic Simulation Using Live Data from Sensors for Smart Cities. *Mabs 2020* 21st Inter (2020), 1–12.

[4] Jean Bélanger, P Venne, and Jean-Nicolas Paquin. 2010. The what, where and why of real-time simulation. *Planet Rt* 1, 1 (2010), 25–29.

[5] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

[6] 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

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

[8] Jakob Erdmann. 2014. Lane-changing model in SUMO. *Proceedings of the SUMO2014 modeling mobility with open data* 24 (2014), 77–88.

[9] Morten W Fagerland. 2012. t-tests, non-parametric tests, and large studies—a paradox of statistical practice? *BMC medical research methodology* 12, 1 (2012), 78.

[10] Dwayne Henclewood, Wonho Suh, Angshuman Guin, Randall Guensler, Richard M. Fujimoto, and Michael P. Hunter. 2016. Real-time data-driven traffic simulation for performance measure estimation. *IET Intell. Transp. Syst.* 10, 8 (2016), 562–571. https://doi.org/10.1049/iet-its.2015.0155

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

[12] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. 2018. Online learning: A comprehensive survey. *arXiv preprint arXiv:1802.02871* (2018).

[13] Xiuling Huang, Jie Sun, and Jian Sun. 2018. A car-following model considering asymmetric driving behavior based on long short-term memory neural networks. *Transportation Research Part C: Emerging Technologies* 95 (2018), 346 – 362. https://doi.org/10.1016/j.trc.2018.07.022

[14] David Jones, Chris Snider, Aydin Nassehi, Jason Yon, and Ben Hicks. 2020. Characterising the Digital Twin: A systematic literature review. *CIRP J. Manuf. Sci. Technol.* 29 (2020), 36–52. https://doi.org/10.1016/j.cirpj.2020.02.002

[15] Venkatesan Kanagaraj, Gowri Asaithambi, C.H. Naveen Kumar, Karthik K. Srinivasan, and R. Sivanandan. 2013. Evaluation of Different Vehicle Following Models Under Mixed Traffic Conditions. *Procedia - Soc. Behav. Sci.* 104 (2013), 390–401. https://doi.org/10.1016/j.sbspro.2013.11.132

[16] Y Kendrik, L Hong, Z Pai, and C Chun-Hsien. 2020. A state-of-the-art survey of digital twin: Techniques, engineering product lifecycle management and business innovation perspectives. *J. Intell. Manuf* 31 (2020), 1313–1337. https://doi.org/10.1007/s10845-019-01512-w

[17] 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).

[18] 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

[19] 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.

[20] 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.

[21] Arash Mahdavi, Niki Athanasiadou, and Heman Kapadia. [n.d.]. *Combining Simulation and Machine Learning*. Youtube. https://www.youtube.com/watch?v=PMFWQigpgZA&feature=youtu.be&ab_channel=AnyLogic

[22] Htet Naing. 2021. Supplementary Materials. https://github.com/Javelin1991/dynamic_traffic_simulation.

[23] 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.

[24] 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

[25] 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.

[26] Andreas Pell, Andreas Meingast, and Oliver Schauer. 2017. Trends in Real-time Traffic Simulation. *Transp. Res. Procedia* 25 (2017), 1477–1484. https://doi.org/10.1016/j.trpro.2017.05.175

[27] 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

[28] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[29] Mustapha Saidallah, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. 2016. A comparative study of urban road traffic simulators. In *MATEC Web of Conferences*, Vol. 81. EDP Sciences, 05002. https://doi.org/10.1051/matecconf/20168105002

[30] Mohammad Saifuzzaman and Zuduo Zheng. 2014. Incorporating human-factors in car-following models: A review of recent developments and research needs. *Transportation Research Part C: Emerging Technologies* 48 (2014), 379 – 403. https://doi.org/10.1016/j.trc.2014.09.008

[31] Samarth Swarup and Henning S Mortveit. 2020. Live Simulations. *Proc. 19th Int. Conf. Auton. Agents MultiAgent Syst.* Aamas (2020), 1721–1725.

[32] 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.

[33] 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

[34] 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

[35] 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

[36] 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

[37] 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

[38] 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

[39] 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.