Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

ENHANCING REALISM IN SIMULATION THROUGH DEEP LEARNING

Muhammad Shalihin Bin Othman
Gary Tan

Department of Computer Science, School of Computing
National University of Singapore
13 Computing Drive
Singapore City, 117417, SINGAPORE

ABSTRACT

Modeling and simulation have been around for years and its application to study several different systems and processes have proven its practical importance. Various research has sought to optimize its performance and capabilities, but few address the issues of generating realistic inputs for simulating into the future. In this paper, some issues in the commonly used simulation flow were identified and deep learning was introduced to enhance realism by learning historical data progressively, so as to generate realistic inputs to a simulation model. We focus on improving the input generation phase and not the model of the system itself. To the best of our knowledge, this is the first work that realizes the possibility of integrating deep learning models directly into simulation models for general-purpose applications. Experiments showed that the proposed methods are able to achieve higher overall accuracy in generating input sequences as compared to current state-of-art.

1 INTRODUCTION

In today's world, systems and business processes are sought to be improved and optimized day to day. Modeling and simulation is a powerful tool that has been used to improve and optimize systems and processes (Carson 2005), with applications in several disciplines such as Engineering, Medicine and many more. Input data to a simulation model is the foundation of a good simulator (Ungureanu et al. 2005). It drives the simulation and can be generated either by identifying a theoretical distribution if a good fit for the data exist, or an empirical distribution if there is not, or even information gathered from different sources such as expert's opinion if no data exist (Biller and Gunes 2010). While these methods have been successfully used for several works (Frost and Melamed 1994), there is still much that can be argued as to how effective these generated data can represent a system under study. This invoked several questions that our research aims to seek a solution for. *What if a system changes significantly over time? What if some of the system variables are dependent on certain events? How can we influence the input generator based on such events effectively? What if there is a relationship between data from further back in the historical records? How can a generator "remember" such representations of data?*

A theoretical distribution certainly will not be able to accommodate such changes in the system without manual intervention. An empirical distribution, may have the flexibility (Shanker and Kelton 1991) to change its distribution as the system changes, but does not have the properties of "remembering" events that are repeating itself or "relate" the influence an event might have to the data sample. If an input generator could effectively recognize such events and "remember" them based on historical data, the realism of the simulation can certainly be improved, yielding results for a more effective analysis. Thus, we investigate the use of Representation Learning (RL), also known as Machine Learning (ML), that has been successfully used in several applications (Witten et al. 2016) over the years.

Machine learning is a set of algorithms that allow the automatic discovery of representations from raw data needed for detecting features or classification. Another form of representation learning that has gained much popularity in recent years uses deep neural networks (DNN), also known as Deep Learning (DL). It has achieved close to perfect accuracy on applications such as visual imagery with the Inception-ResNet (Szegedy et al. 2017) for image classification, the Recurrent Neural Network (RNN) on text classification (Liu et al. 2016), and many more.

This paper aims to explore the issues in current state-of-art for generating inputs to a simulation model and apply deep learning techniques to overcome these issues. We will leave the realistic modeling of the system to the modeller and focus our research on improving the realism of input data generation. Hence, our main contribution to the state-of-art in the *input generation phase*, is the novel method for generating inputs and not the model of the system itself. The aims and objectives of this paper are to produce a conceptual contribution for integrating deep learning techniques into simulation modeling in order to achieve a better sense of realism and propose a novel method to generate realistic inputs to a simulation model based on historical data learned over time. In this paper, we use the term "enhancing realism" to refer to our novel method of using deep learning to learn from data and generate more realistic inputs to a simulation model, as opposed to current methods of modeling stochastic distributions. We would like to highlight that we are not making any contributions to existing deep learning algorithms, but rather to leverage on recent successes to improve a specific part of the simulation framework, the input generation phase. Additionally, the scarcity of data may prevent the effective use of our methods due to its learning needs. However, moving towards a future of Internet-of-Things (IoT), we can see that data can be much easily collected. The novelty of this paper also comes from the idea that if we have enough data, we can utilize them to carry out better simulations with accurate measurements for effective analysis.

We start off by looking at the literature of some related works done in these areas in Section 2. Section 3 will formally define the problem this paper aims to solve and Section 4 will delineate the proposed methods. The experiments and evaluation of methods proposed will be presented in Section 5. A simple simulator implemented with the proposed methods is described in Section 6. Finally, we conclude along with further work in Section 7.

2 RELATED WORK

Modeling and simulation has been around for years and its evolution is accredited to the advancements of diverse disciplines in Computer Science such as Systems Theory, Systems Engineering, Software Engineering, Artificial Intelligence, and more. In this paper, we review some related work that concerns the input modeling phase in simulation.

Shanker et. al. (Shanker and Kelton 1991) highlighted the problem of input-distribution selection in simulation modeling and suggested that any estimator of a true distribution should be generalizable and have the capability of modeling a variety of distributional shapes. Hence, empirical input distributions was proposed as an alternative to standard input distributions for simulation modeling since the observed data themselves would be used in some way to form a distribution function. This allows the input model to be more flexible to changes in data. In a tutorial paper by Biller et. al (Biller and Gunes 2010), the authors reviewed standard input models and identified cases where they fail to adequately represent the input data concerned. The authors reviewed cases where an input process may have characteristics not captured by standard distributions, cases where data exhibit dependence, and cases where data changes over time. However, the paper concludes with separate solutions for each case without reviewing data where all 3 characteristics may be involved. In a paper by Zouaoui et. al. (Zouaoui and Wilson 2001), the authors proposed a Bayesian approach to probabilistic input modeling. The proposed methods accounts for the parameter and stochastic uncertainties in the data, that yields valid predictive inferences about outputs of interest. An experimental performance evaluation demonstrated the advantages of their approach. This brings us to the advancements in simulation that involves some form of learning in order to make better predictions.

Although a number of research that aims to improve realism in simulation and modeling have been explored (Kimko and Lee 2017), not much work have been done to enhance realism in simulation using deep learning. A paper by Fishwick et. al. (Fishwick 1989) made a comparison for traditional methods and a Multi-Layer Perceptron (MLP) model as a simulation model. However, accuracy problems arise because the neural network model does not capture the system structure characteristic of all physical models. Following this work, a paper on a hybrid model that aims to integrate big data and deep learning into simulation (Tolk 2015), also used an MLP model to capture correlation in huge amounts of data so as to model the system under study more accurately. The paper proposed big data processing to evaluate the information, then deep learning to harmonize the processes by discovering underlying common functionality. Simulation is then used to execute the resulting system of systems representations. In another related work, a conceptual framework for integrating ML models into simulation, Elbattah et. al. (Elbattah and Molloy 2018) proposed a few key ideas that were somewhat similar to our work. However, no experiments or projects were done to realize the concept. This paper also supports our idea for incremental learning that would enable automated updates to the models, where variables may constantly change, without human intervention.

In our recent work (Othman et al. 2017), a Multi-Layer Perceptron & Linear Regression (MLP-LR) model was proposed to predict congestion in traffic. Thereafter, we proposed a traffic simulation model (Othman and Tan 2018b; Othman and Tan 2018a) that is able to take possible congestion into account when running simulations. However, they are focused on predicting a specific variable and therefore not general enough for other applications. To the best of our knowledge, this is the first work that realizes the integration of deep learning models directly into simulation models, focusing on a general method to use deep learning in generating inputs that influences the system behaviour.

3 ISSUES IN CURRENT PRACTICE

The process of building a typical simulation model starts by collecting the essential data for the purpose of studying the system. The distribution of the data are then studied to find the best function that can generate values randomly throughout the simulation time. These values are then fed to a simulation model that replicates the system under study, and produces output measurements for analysis. Our focus, which is the *input generation phase* and not the model of the system itself, includes the processing of data, analyzing the data, and using the data to generate meaningful synthetic data for use in a simulation model, either to test the robustness of a system or seek optimization strategies. As mentioned in Section 1, we will discuss the two main categories for deriving data distributions to generate synthetic data.

3.1 Theoretical Distributions

A theoretical distribution simply studies the distribution of data and fits a known distribution, such as Poisson, Exponential, Gamma, etc., by optimizing the goodness of fit through statistical tests such as the χ-square or KolmogorovSmirnov (K-S) test, etc. However, this method is not capable of modeling a variety of distributional shapes.

To illustrate an example, in traffic management, transport authorities would like to improve the public transportation services by providing commuters with services that meet their needs and at the same time cost effective for the transport operators. In this instance, commuters would want the shortest waiting time possible while transport operators would want their services to be fully utilized. In order to achieve this, the frequency at which buses are deployed should be regulated. Here, the frequency can be tuned and outputs to show the bus utilization will be dependent on passenger volumes as well as the transport service travel times. Based on our initial experiments, we find that the data for travel times from the same day and the same route, just different yet consecutive weeks, produced very different best-fit theoretical distribution. However, both their sequence of average travel time per hour between 2 bus stops throughout a typical weekday shows common patterns that reflects real-world expectations.

Spikes in the travel times are experienced during peak periods, for buses that travel along busy roads and tend to get caught in congestion when people would be going to work and coming back home from work. Since a theoretical distribution is fixed once sample data is computed, it can only generate values based on data it has seen. If the system changes over time, this method would not be flexible enough to capture the changes in distributional shapes.

3.2 Empirical Distribution

In order to tackle the issues of inflexibility, an empirical distribution is used. This method is useful for data samples that cannot fit into a known theoretical distribution. Since the computation for its cumulative probability distribution (CDF) includes the data samples, an empirical cumulative distribution function (ECDF) is more flexible to changes in the system's behaviour. The ECDF is computed as follows, given a data sample $X_1 \dots X_n$:

$$ECDF(\tau) = \frac{1}{n} \sum_{i=0}^{n} 1_{X_i \leq \tau}$$

The ECDF of τ simply computes the number of elements lesser than or equal to τ divided by the total number of elements in the sample. This would give us the probability of τ occurring in the sample. We can then create a random number generator from data samples, by estimating the inverse ECDF function using interpolation. The line plot of the ECDF Random Number Generator (RNG) against data samples for the actual sequence of travel times on the same route for two consecutive Mondays showed that although the actual data changes between the 2 weeks, the Empirical RNG is still able to adapt to the change and generate values accordingly to the latest data samples, with a root mean squared error (RMSE) of only 0.08. However, if we consider simulating into the future using this method, the distribution can only be based on the latest week where data is available. Thus, it assumes that the next week would have the same distribution as the current week. As we have discovered in Section 3.1, this may not be true for variables that are dependent on time or events. Hence, we found that the current state-of-art for input data generation is lacking in the sense for future simulations where it does not retain any memory of data and make relations to realistically predict future weeks where data is not yet available, for example; congestion during peak hours, etc.

Therefore, if an input generator is able to make such connections, simulations into the future would certainly benefit from more realistic outputs. Several researches, as discussed in Section 2, have been involved in trying to improve or speed up simulation, but none are focused on enhancing the level of realism in predicting the inputs for simulation into the future.

4 PROPOSED METHODS

Inputs that are time-dependent or event-dependent cannot be produced realistically by current state-of-art methods, resulting in inaccurate output analysis. Hence, a deep learning approach, using the Long-Short-Term Memory (LSTM) network, is proposed to generate inputs in a sequence that is both time dependent and conforms to the distribution of data. Other deep/machine learning models were also considered, however, for sequence to sequence prediction that we need for generating inputs, LSTM stands as the current state-of-art.

The LSTM, well known for its ability to remember sequences through time, was proposed specifically to tackle the issue of long term dependencies that a regular Recurrent Neural Network (RNN) suffers from. It does so by introducing the cell state C_t and feeding it to the next LSTM together with the output, instead of just the output alone like the RNN. The logical sequence of LSTM is quite straightforward as it commits to retaining and disposing information just like the neurons in our brains would (Hochreiter and Schmidhuber 1997).

The LSTM has seen many successes over the years and is indeed an ideal solution for sequence prediction, although optimal hyper-parameter tuning is still an ongoing research area. The novelty of this paper is also attributed to the usage of the LSTM for generating realistic input sequences from learned data since this has not been done before or made any significant impact with regards to generating inputs to a simulation model. Thus, this work explores that possibility, coming up with an improved general framework that can integrate seamlessly with the rest of the simulation flow. Figure 1 shows the general concept we proposed, for a typical simulation flow using the LSTM for input generation.

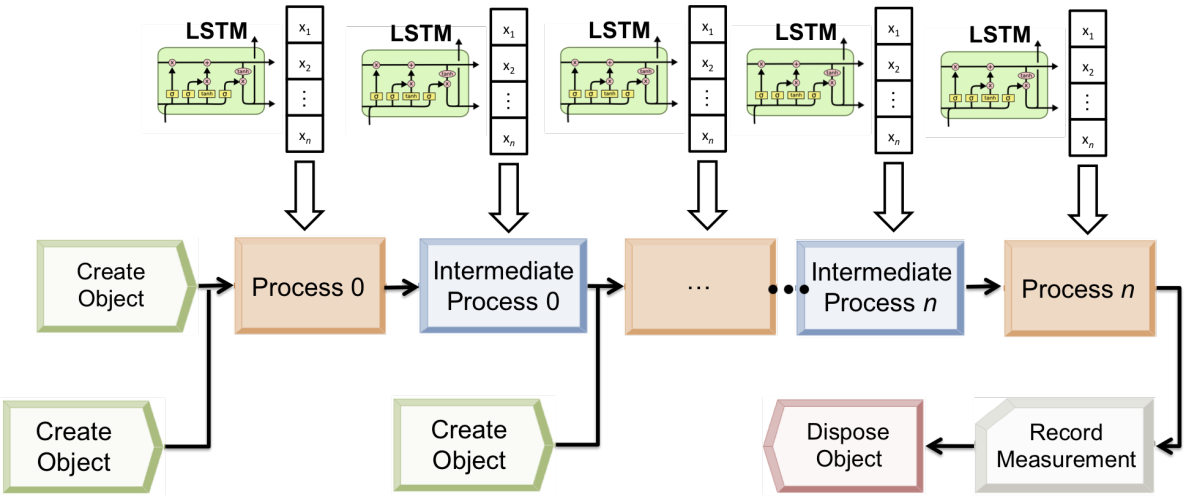


Figure 1: Overview of a typical simulation flow with deep learning.

Based on the simulation flow in Figure 1, entities are created and passed through a series of processes where an input distribution is required to determine how long an entity will spend in the process or how the process should affect the entity. At each process, an LSTM model can be used to generate sequences of inputs throughout the simulation time. Output measurements would then be recorded during the simulation before all entities are disposed. For each variable that needs a sequence to be fed to the simulation model, we can train the model as shown in Figure 2.

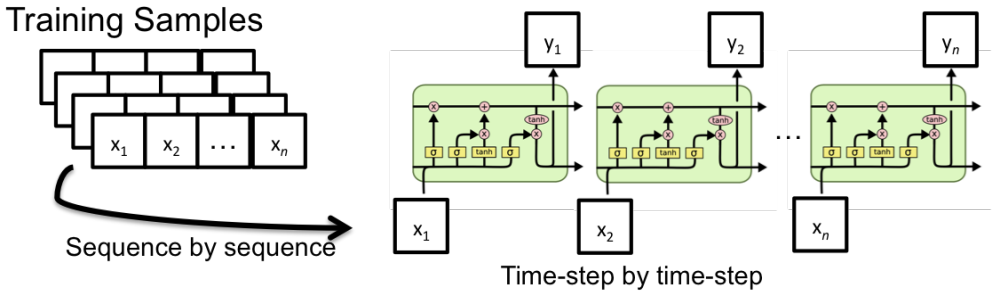


Figure 2: Training sequences with LSTM.

Training sample $x_1, ..., x_n$ is the sequence of the variable under study through a single day. Input $x_1, ..., x_n$ will produce outputs $y_1, ..., y_n$ that represents the following week's sequence. The following week's sequence is used to propagate back through the LSTM, correcting its weights and biases, before it is fed as the input for the next forward propagation phase. By training the LSTM in this manner, the LSTM will be able to learn the sequence pattern for a specific variable through the day, week after week. The weights and biases for the LSTM are learned using back-propagation through time (Werbos 1990), where errors at the end of each time step are computed and derivatives are calculated to correct the weights and biases as it rolls back through the time sequence using the chain rule.

After training the model, predictions can be made by passing in a full day sequence as inputs and get a full day sequence as output. For example, if we would like to predict for the whole day tomorrow, we can pass in the sequence gathered from today as input to the model that returns an output representing tomorrow's sequence. With the output sequence, we can also feed it back to the model to get the predicted sequence for the following day. It can go on for as long as required for the simulation. Intuitively, the accuracy would certainly start to decrease as we try to predict further into the future. Different configurations of the LSTM model were explored, such as using different number of hidden layers, different number of neurons, as well as applying different configurations to the parameters. The final LSTM model used consists of 3 hidden layers with 20, 40 and 10 neurons respectively and trained using an Adam Optimizer (Kingma and Ba 2015) with 0.01 learning rate over 500 epochs. A dropout layer with 0.05 dropout rate was also added between each hidden layer and finally, a dense layer before producing the output sequence. The dropout layer was introduced to "forget" some measure of the outputs so that we would not retain information that has not occurred in a long time.

As this is a general concept for applying deep learning methods to a simulation model, we will focus our experiments on the capability of the deep learning model in generating sequences that are more realistic so as to effectively simulate into the future.

5 EXPERIMENTS AND RESULTS

The measurements and outputs from a simulation model are dependent on its inputs. Hence, we can justify that regardless of the case scenarios applied in simulation, the input parameters will ultimately determine the accuracy of the outputs. Therefore, it is sufficient to show that the model can predict sequences and distributions more accurately than a random number generator based on a distribution function. Since we have shown in Section 3 that theoretical distributions are inflexible to data changes over time, we make comparisons in our experiments for the LSTM against random values generated from an empirical distribution function. Since the ECDF we have discussed is the most common method for generating stochastic data that does not conform to a known distribution, we will evaluate the accuracy of predicting sequences as well as the hypothesis test for data generated by the LSTM model, against values generated by the ECDF.

5.1 Experimental Setup

To narrow down the scope and achieve certainty in the experiments, traffic data was chosen as the primary focus. This is because in traffic management, variables such as travel time or passenger volumes (bus utilization), etc., are time-dependent as well as event-dependent, i.e. changes during peak and non-peak periods, working days and non working days, holidays, etc. We will discuss the sources of data and how we generate input values from the LSTM as well as the Empirical RNG.

The Land Transport Authority (LTA) of Singapore has published a wide variety of transport-related datasets. The datasets include several information such as real-time incidents, scheduled maintenance, schedules and routes of public transports, etc. More specifically, we will be using the following datasets provided through the API; (a) Bus Services, (b) Bus Routes and (c) Bus Stops. In order to carry out a thorough experimentation, 8 different bus services going through congested and non congested routes, were selected for Monday to Sunday. Figure 3 shows the congested and non-congested stretch of road, where the 8 different bus services travel on. The routes circled in red (top 4) are congested routes while the routes circled in green (bottom 4) are non-congested routes.

Firstly, information for the 8 different bus services were collected from API (a) and the routes for each service were retrieved through API (b). This will give us a list of bus stop codes covered by a specified bus service. We can then get detailed bus stop information for each bus stop code from API (c) such as the latitudes and longitudes.

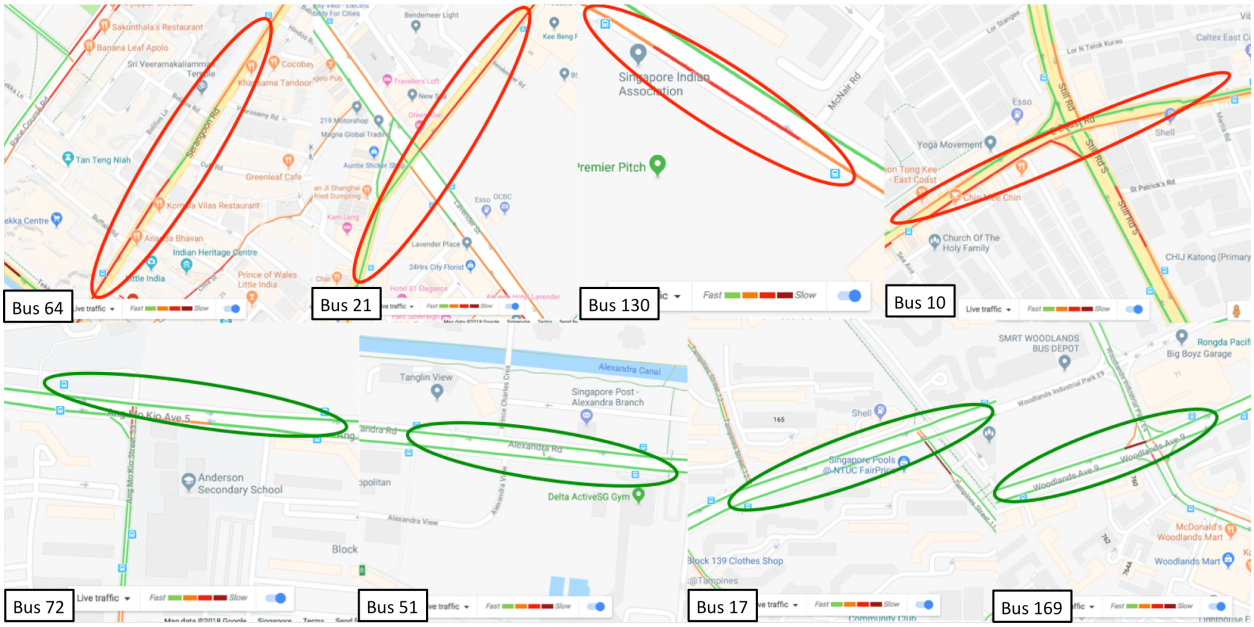


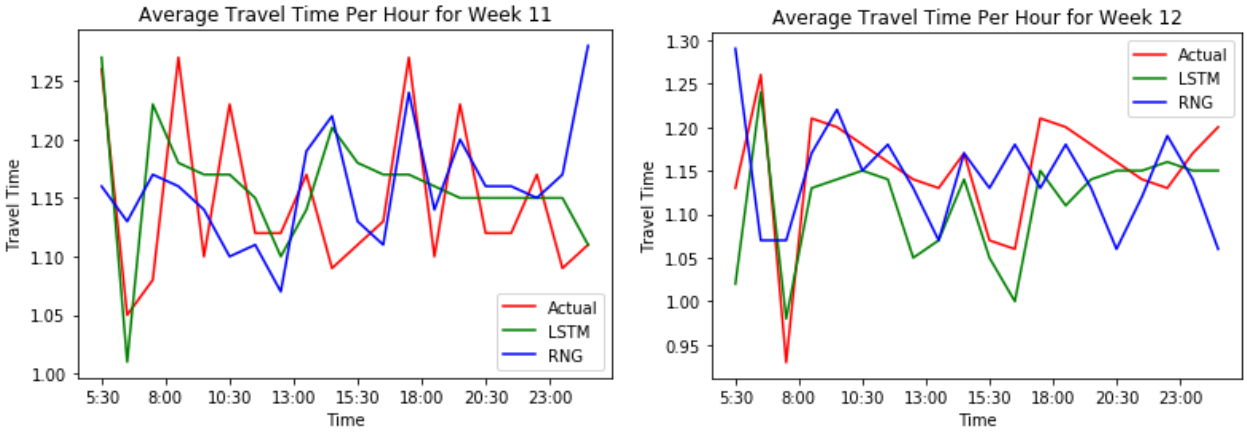
Figure 3: Selected road stretch for experiments.

Since the arrival times from LTA's API are fixed and does not reflect the actual distribution and sequence well, we use Google Maps Distance Matrix API to get the actual travel time from 1 stop to another. Google Maps incorporates real-time congestion in their API through the "duration_in_traffic" variable. We repeat this process for Monday to Friday over 12 weeks, which is roughly 3 months' worth of data. Now, we have 12 samples for each bus stop to bus stop for Monday through to Sunday. Each sample will have the sequence of travel times for the whole day. The length of the sequence may differ for different bus services hence for training, the average over 20 intervals were computed so that we have the average travel time per hour throughout a particular day. After some analysis, we found that for travel times in traffic, the distribution changes from day to day. Hence, we modelled each day independently based on the data collected for the day through 12 consecutive weeks. Therefore, for each day, we have 12 independent samples for training and evaluation.

Based on the model we explained and built in Section 4, using Google's Tensorflow (Abadi et al. 2016) library, and the 12 samples we have collected, the LSTM model was trained over 9 weeks worth of data, where the 9^{th} week's data was trained using the 10^{th} week's data. The 10^{th} week's data was then used to predict Week 11's sequence and the predicted sequence was fed back to the model to predict the sequence for Week 12. Since we have the actual data for Week 11 and Week 12, we can realistically evaluate the predictions made by the LSTM model. Hence, the LSTM model itself will not see any data from Weeks 11 and 12. For comparisons with current state-of-art methods, a random number generator (RNG) based on the Empirical Cumulative Distribution Function (ECDF) described in Section 3.2 was created by estimating its inverse using interpolation. Uniformly distributed random values are then passed through the RNG to produce input values. In order to make comparisons with the predicted values from the LSTM model, we used Week 1 to Week 10's data to generate values for Week 11 and Week 12. This is fair since the LSTM model only sees data up to Week 10 as well. Therefore, both the values from the ECDF and LSTM would generate inputs to represent the travel time throughout a particular day. The data generated by the empirical random number generator will be simply referred to as RNG in the following sections.

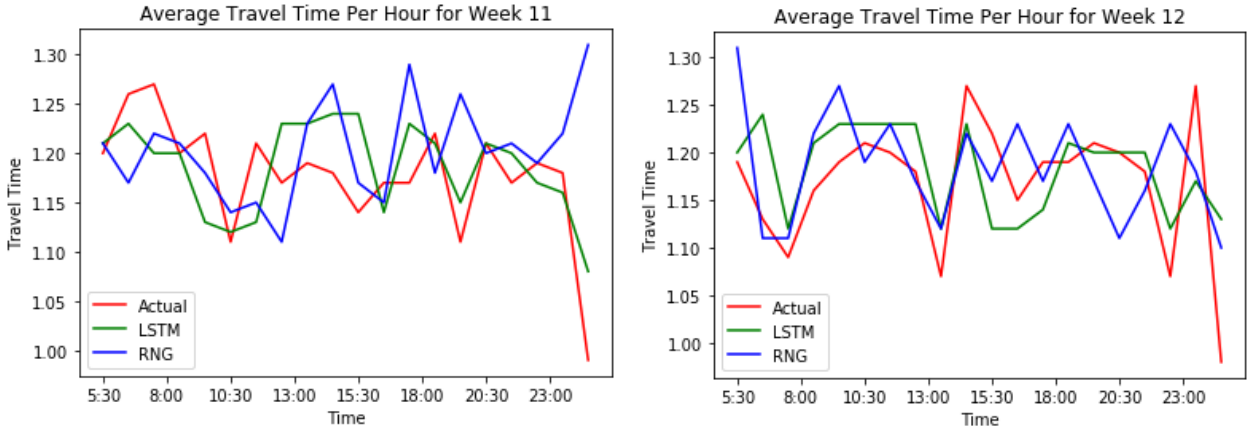
5.2 Sequence Accuracy of LSTM vs RNG

The synthetic data generated by the RNG and the predicted sequence produced by the LSTM model are plotted against the actual data for a congested route in Figure 4(a) for Week 11 and Figure 4(b) for week 12, as well as for a non-congested route in Figure 5(a) for Week 11 and Figure 5(b) for Week 12.



(a) Bus 64 Week 11 (b) Bus 64 Week 12

Figure 4: Travel time sequence for congested routes.



(a) Bus 17 Week 11 (b) Bus 17 Week 12

Figure 5: Travel time sequence for non-congested route.

We can roughly see from Figures 4 and 5 that the LSTM is able to learn the sequence through training and eventually produce a sequence prediction better than the RNG. Experiments were repeated for different days of the week on the 8 different routes and results were consistent. Even though some routes that were less busy produced a rather uniformed sequence, the LSTM model was still able to capture it and predict fairly accurate future sequences simply based on the previous week's data. In order to clearly distinguish the performance of the two different techniques, we computed the Root Mean Squared Error (RMSE) of the actual data against the LSTM model and the RNG as follows:

$$RMSE = \sqrt[2]{\frac{\sum_{i=0}^{n}(x_i - y_i)^2}{n}}$$

Table 1 shows the error scores for the LSTM and the RNG for Week 11 while Table 2 shows the scores for Week 12. Only 2 bus services for congested route (64 & 21) and 2 bus services for non-congested route (72 & 17) for Monday, Thursday and Saturday are presented for visual clarity.

Table 1: Week 11 RMSE Scores for LSTM vs RNG.

Bus Services	Mon		Thu		Sat	
	LSTM	RNG	LSTM	RNG	LSTM	RNG
64	0.08	0.06	**0.07**	0.08	**0.16**	0.17
21	0.10	0.10	**0.08**	0.09	**0.08**	0.10
72	**0.05**	0.06	**0.07**	0.09	0.11	0.08
17	**0.08**	0.09	**0.08**	0.10	0.11	0.11

Table 2: Week 12 RMSE Scores for LSTM vs RNG.

Bus Services	Mon		Thu		Sat	
	LSTM	RNG	LSTM	RNG	LSTM	RNG
64	**0.08**	0.10	0.09	0.08	0.10	0.08
21	**0.06**	0.09	**0.09**	0.10	**0.09**	0.11
72	0.10	0.08	**0.07**	0.09	**0.10**	0.12
17	0.08	0.07	0.09	0.07	0.08	0.08

The scores in bold text indicates the LSTM performing better than the RNG. Basically, the deep learning model outperforms the RNG more than 70% of the time. This justifies that the LSTM model can generate sequences more accurately to actual data than the RNG.

5.3 Hypothesis Testing: LSTM Sequence Prediction

In order to show that the LSTM can produce samples similar to the actual data, we evaluate the generated data from the LSTM model using the t-distribution test. We first we define a null hypothesis (H_0) and its alternate hypothesis (H_1) as follows:

Hypothesis H_0. Deep learning can generate accurate input representations of real-world systems.

Hypothesis H_1. Deep learning cannot generate accurate input representations of real-world systems.

With a level of significance set at $\alpha = 0.05$, we compute the sample mean $\overline{X}$ and the standard deviation S of LSTM over 20 samples, against the mean of the actual data as target ρ to compute the t_0 scores. For two-sided test, we reject H_0 if $|t_0| > t_{a/2,n-1}$. Based on the t distribution table, $t_{0.05/2,20-1} = 2.093$. Table 3 shows the $|t_0|$ scores for 2 congested routes and 2 non-congested routes over 3 different days, as visualization samples.

Table 3: $|t_0|$ scores for LSTM model.

Bus Services	Mon		Thu		Sat	
	Week 1	Week 2	Week 1	Week 2	Week 1	Week 2
64	0.27	0.04	0.53	1.90	0.45	1.39
21	0.65	0.62	0.07	1.23	1.05	0.01
72	0.28	0.18	1.82	0.05	0.76	0.43
51	0.61	0.24	0.92	0.01	0.26	0.30

Since for all cases, $|t_0| < 2.093$, we cannot reject H_0. We can then safely conclude that the statement in H_0 holds, and the rest of the results from our experiments supports it as well. The improvements seen through the epochs also show that with continuous training, the LSTM model can make better predictions and close up the gap with actual data over time. Therefore, with this model, we can successfully introduce a better sense of realism when simulating into the future, producing more insightful results for better analysis.

6 SIMULATION AND DEEP LEARNING: A CASE STUDY

A case study of the public transportation system was done to demonstrate the viability of integrating our proposed methods into real-world simulations. Algorithm 1 shows the pseudo code for the simulator, developed in Python, to evaluate the utilization of public buses.

Algorithm 1: Bus utilization simulator.

```
Require: Predicted travel times for each bus stop through the day
Require: start_time, end_time, passenger arrival rate
simulation_time, next_deployment_time = start_time;
while simulation_time < end_time do
    if simulation_time == next_deployment_time then
        Deploy bus from origin:
         stop_number = 1;
         arrival_time = simulation_time + travel_time;
        next_deployment_time + bus_frequency;
    end
    for each deployed bus do
        if simulation_time == arrival_time then
            Alight passengers and board passengers at bus stop;
            stop_number + 1;
            arrival_time = simulation_time + travel_time;
        end
    end
    Generate commuters at each stop;
    simulation_time + 1;
end
```

The simulation model will deploy buses at scheduled times according to the schedules set out by the transport operators. Commuters are then generated at each stop from origin to the stop before the destination, where they will board the bus when the bus arrives at the bus stop. Between each stop, the bus will travel for a period of time and at each stop, passengers in the bus may alight and commuters at the bus stop may board. The capacity of the bus will be updated and if it reaches maximum capacity, the commuters at the bus stop will have to wait for the next bus. This flow will continue throughout the day until the last bus is deployed. For every stop that has passengers alighting, we record the total waiting time for reporting and analysis. Measures of bus utilization and the passengers' waiting time will be recorded upon successfully boarding the bus.

We verified the correctness of the simulation model based on the planned schedules we retrieved from the LTA Data Mall as well as the prediction accuracy we have shown in Section 5.2. The following justifications based on real-life scenarios, are made to ensure that the model has high face validity; each bus service travels non-stop from origin to destination, the travel times (> 0 mins) include the time taken to board and alight passengers, no passengers should be alighting at origin but all passengers need to alight at destination, and there may be 0 or more passengers boarding the bus at every boarding station.

In a terminating solution, the method of independent replications is most commonly used to analyze the outputs. Based on 4 independent replications, we test whether the utilization meets a certain standard using the Student's t-test. We define the following hypothesis for utilization (ρ):

$$\textbf{Hypothesis } \mathbf{H_0}. \ \rho \geq 0.90$$

$$\textbf{Hypothesis } \mathbf{H_1}. \ \rho < 0.90$$

For sample purposes, the test statistics for 2 bus services, 130 (congested route) and 17 (non-congested route) are computed with t_0 scores of 1.65 and 1.12 respectively. Based on the t distribution table, $t_{0.05/2,4-1} = 3.182$. Since both the $|t_0|$ scores for Bus 130 and 17 is < 3.182, we cannot reject H_0. Hence, the utilization condition where the bus is utilized more than 90% holds. The simulation and analysis were repeated with the actual travel times for Weeks 11 and 12 against the empirical distribution RNG. We ran each of them over the same number of replications and computed the $|t_0|$ scores. We found that the average squared difference between the $|t_0|$ scores using the predicted travel times is only about 9.44 while the empirical RNG scored about 27.46, which is more than two times higher. This concludes that the predicted inputs can indeed produce simulation results that are more realistic than the current state-of-art methods.

7 CONCLUSIONS AND FURTHER WORK

We showed that our method can achieve more accurate results for both congested and non-congested routes, where non-congested routes (less travelled on) should perform well even for traditional methods since the travel times should be somewhat regular throughout the day. Our methods can accommodate for both cases without manual intervention. The proposed methods still require multiple what-if scenarios for simulation, but the input to the simulation, such as travel times, volumes, etc. would be a more accurate representation of the real-world. This in turn provides output measurements that are much more realistic. By introducing deep learning methods to study variables that changes their distribution over time, we could effectively predict values for a more realistic simulation into the future. Additionally, we also showed how our methods can easily be applied to a simulation model. The deep learning models can also support variables that are not time-dependent, yet varies in distribution over time. Thus, the proposed method is ideal for auto-tuning a simulation model and inject a higher sense of realism, showing more than 70% accuracy as compared to less than 20% using an empirical RNG. This would be highly beneficial for systems that are highly critical to such accuracy, such as traffic management or crisis management.

Several avenues for further research can be extended from this work such as exploring other variables or data types that may impact the simulation results and improve their level of realism through the proposed methods. Other deep learning models that may be able to provide an even better accuracy when making sequence predictions on variables that are volatile may also be explored. Additionally, since these models can operate independently for each variable, parallelism is also very much possible. With the ability of the LSTM model to change its output prediction as the training data changes, we can see huge potential for implementing this automated modeling into a real-time system, where time is of the essence. Several systems and processes can certainly benefit from such effective and highly realistic simulation framework.

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. 2016. "TensorFlow: A System for Large-scale Machine Learning". In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, 265–283. Berkeley, California: USENIX Association.

Biller, B., and C. Gunes. 2010. "Introduction to Simulation Input Modeling". In *Proceedings of the 42nd Winter Simulation Conference*, edited by B. Johansson, S. Jain, and J. Montoya-Torres, 49–58. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Carson, II, J. S. 2005. "Introduction to Modeling and Simulation". In *Proceedings of the 37th Winter Simulation Conference*, edited by N. Steiger and M. E. Kuhl, 16–23. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Elbattah, M., and O. Molloy. 2018. "ML-Aided Simulation: A Conceptual Framework for Integrating Simulation Models with Machine Learning". In *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, edited by F. Quaglia, A. Pellegrini, and G. K. Theodoropoulos, 33–36. New York, NY: Association for Computing Machinery.

Fishwick, P. A. 1989. "Neural Network Models in Simulation: A Comparison with Traditional Modeling Approaches". In *Proceedings of the 21st Winter Simulation Conference*, edited by E. A. MacNair, K. J. Musselman, and P. Heidelberger, 702–709. New York, NY: Association for Computing Machinery.

Frost, V. S., and B. Melamed. 1994. "Traffic Modeling for Telecommunications Networks". *IEEE Communications Magazine* 32(3):70–81.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory". *Neural Computation* 9(8):1735–1780.

Kimko, H., and K. Lee. 2017. "Improving Realism in Clinical Trial Simulations via Real-World Data". *CPT: Pharmacometrics & Systems Pharmacology* 6(11):727–729.

Kingma, D. P., and J. Ba. 2015. "Adam: A Method for Stochastic Optimization". In *3rd International Conference on Learning Representations*, edited by Y. Bengio and Y. LeCun. Amherst, Massachusetts: OpenReview.

Liu, P., X. Qiu, and X. Huang. 2016. "Recurrent Neural Network for Text Classification with Multi-task Learning". In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, edited by G. Brewka, 2873–2879. Palo Alto, California: Association for the Advancement of Artificial Intelligence Press.

Othman, M. S. B., S. L. Keoh, and G. Tan. 2017. "Efficient Journey Planning and Congestion Prediction through Deep Learning". In *Proceedings of the 3rd International Smart Cities Conference*, edited by IEEE Power & Energy Society, 1–6. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Othman, M. S. B., and G. Tan. 2018a. "Machine Learning Aided Simulation of Public Transport Utilization". In *Proceedings of the 22nd International Symposium on Distributed Simulation and Real Time Applications*, edited by E. Besada-Portas, Ó. R. Polo, R. E. D. Grande, and J. L. Risco-Martín, 253–254. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Othman, M. S. B., and G. Tan. 2018b. "Predictive Simulation of Public Transportation Using Deep Learning". In *Methods and Applications for Modeling and Simulation of Complex Systems*, edited by L. Li, K. Hasegawa, and S. Tanaka, 96–106. Singapore: Springer Singapore.

Shanker, A., and W. D. Kelton. 1991. "Empirical Input Distributions: An Alternative to Standard Input Distribution in Simulation Modeling". In *Proceedings of the 23rd Winter Simulation Conference*, edited by B. L. Nelson, 978–985. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Szegedy, C., S. Ioffe, V. Vanhoucke, and A. A. Alemi. 2017. "Inception-v4, inception-ResNet and the Impact of Residual Connections on Learning". In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, edited by S. P. Singh and S. Markovitch, 4278–4284. Palo Alto, California: Association for the Advancement of Artificial Intelligence Press.

Tolk, A. 2015. "The Next Generation of Modeling & Simulation: Integrating Big Data and Deep Learning". In *Proceedings of the Conference on Summer Computer Simulation*, edited by S. Mittal, I.-C. Moon, and E. Syriani, 1–8. San Diego, California: Society for Computer Simulation International.

Ungureanu, D., F. Sisak, D. M. Kristaly, and S. Moraru. 2005. "Simulation Modeling. Input Data Collection and Analysis". *The 14th International Scientific and Applied Science Conference ELECTRONICS ET*:43–50.

Werbos, P. J. 1990. "Back-propagation Through Time: What It Does and How To Do It". *Proceedings of the IEEE* 78(10):1550–1560.

Witten, I. H., E. Frank, M. A. Hall, and C. J. Pal. 2016. *Data Mining: Practical Machine Learning Tools and Techniques*. 4th ed. San Francisco, California: Morgan Kaufmann Publishers Inc.

Zouaoui, F., and J. R. Wilson. 2001. "Accounting for Input Model and Parameter Uncertainty in Simulation". In *Proceedings of the 33rd Winter Simulation Conference*, edited by B. A. Peters and J. Smith, 290–299. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

AUTHOR BIOGRAPHIES

MUHAMMAD SHALIHIN BIN OTHMAN is a third year PhD student at the National University of Singapore, School of Computing, Department of Computer Science. He received his B.Sc in Computing Science (1st class Honours) from the University of Glasgow. His research interests include traffic and crisis simulation, artificial intelligence as well as parallel and distributed computing. His email address is mshalihin@u.nus.edu.

GARY TAN Gary Tan received his M.Sc and Ph.D from the University of Manchester, UK. He is currently an Associate Professor and Vice Dean at the School of Computing at the National University of Singapore. His research interests include parallel and distributed systems and simulation, high level architecture, symbiotic simulation, traffic simulation and crisis management simulation. His email address is gtan@comp.nus.edu.sg. His website is https://www.comp.nus.edu.sg/∼gtan.