TRAFFIC CONGESTION PREDICTION USING DEEP REINFORCEMENT LEARNING IN VEHICULAR AD-HOC NETWORKS (VANETS)

Chantakarn Pholpol and Teerapat Sanguankotchakorn

Telecommunications Field of Study,
Asian Institute of Technology, Pathum Thani, Thailand

ABSTRACT

In recent years, a new wireless network called vehicular ad-hoc network (VANET), has become a popular research topic. VANET allows communication among vehicles and with roadside units by providing information to each other, such as vehicle velocity, location and direction. In general, when many vehicles likely to use the common route to proceed to the same destination, it can lead to a congested route that should be avoided. It may be better if vehicles are able to predict accurately the traffic congestion and then avoid it. Therefore, in this work, the deep reinforcement learning in VANET to enhance the ability to predict traffic congestion on the roads is proposed. Furthermore, different types of neural networks namely Convolutional Neural Network (CNN), Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) are investigated and compared in this deep reinforcement learning model to discover the most effective one. Our proposed method is tested by simulation. The traffic scenarios are created using traffic simulator called Simulation of Urban Mobility (SUMO) before integrating with deep reinforcement learning model. The simulation procedures, as well as the programming used, are described in detail. The performance of our proposed method is evaluated using two metrics; the average travelling time delay and average waiting time delay of vehicles. According to the simulation results, the average travelling time delay and average waiting time delay are gradually improved over the multiple runs, since our proposed method receives feedback from the environment. In addition, the results without and with three different deep learning algorithms, i.e., CNN, MLP and LSTM are compared. It is obvious that the deep reinforcement learning model works effectively when traffic density is neither too high nor too low. In addition, it can be concluded that the effective algorithms for traffic congestion prediction models in descending order are MLP, CNN, and LSTM, respectively.

KEYWORDS

VANET, traffic congestion, neural network, deep reinforcement learning, SUMO.

1. INTRODUCTION

VANET is a special form of mobile ad-hoc networks (MANETs), which provides intelligent communications for Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I). The study of automobile technology is the first step to an era of self-driving vehicles, which makes drivers feel more convenient. Furthermore, there is a list of potential benefits, such as transport safety enhancement, traffic congestion reduction, road capacity increment, car size and parking space reduction.

As the number of vehicles on the roads has been growing dramatically, traffic congestion is inevitable, which results in a huge wasteful time and resources. Recently, a large number of

researches regarding the Vehicular Ad-Hoc Networks (VANETs) have been carried out, especially to apply VANETs for solving the traffic problem, making it a challenging research topic [3], [13], [15], [18], [19], [20], [24], [25], [26], [28]. However, VANET itself has a limitation and also lack of flexibility to find the congested route and to reroute to the alternative paths. For example, traffic congestion might occur in a very short period of time before getting sparse again, which means rerouting is not necessary. On the other hand, there are also some events that the congestion occurring after the vehicle has already entered that path, and no other roads are available at that time. It is perceived that to adopt only a typical VANET may not be enough for such a situation. In addition, the overall transportation industries are expected to be revolutionized by making it safer, more efficient and more reliable.

To overcome all these problems and limitations, the prediction model for traffic is necessary. A good illustration is that when the congestion spot is detected, instead of immediately performing rerouting, we should be able to repeatedly calculate and forecast if that spot is still congested at around the time a vehicle approaches closed enough to that spot. This is, for preventing an unexpected change in traffic flow after the vehicle has already chosen the new path. In this case, the vehicle will be able to choose another road in time and save the travelling time. However, traffic prediction is one of the difficult and challenging problems, therefore, an intelligent algorithm which can solve such a difficult problem is needed.

In this work, a machine learning technique called deep reinforcement learning is proposed to apply to such a problem. This is a combination of deep learning and reinforcement learning (RL). Deep learning is a promising step to artificial intelligence (AI) that is capable of using multiple layers to progressively extract high dimensional information and discover specific patterns that cannot be done by humans [4], whereas reinforcement learning has ability to learn and improve automatically from mistakes or experiences in the past. Therefore, deep reinforcement learning can be used for solving complex decision-making assignments that were previously unreachable for a machine. Not only it is capable of learning how to accomplish a difficult task, but also it can optimize a particular dimension over many steps [8], [29]. It is anticipated that the problem of traffic prediction model can be handled properly by using deep reinforcement learning as this technology is likely to become more and more popular. Besides, the transportation is one of those factors that affect the quality of daily life.

The performance of our proposed method is illustrated by simulation. The traffic environment in our simulation is created by traffic simulator called SUMO. In order to enhance the efficiency of our prediction model, the simulation has to be repeatedly run many times utilizing the feedback it receives from our created traffic environment. In deep reinforcement learning architecture, there is also a component called an agent that has a major role to decide the actions that it will be performed based on the returning reward. In this work, three different neural network algorithms [17], i.e., Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM), are simulated and compared to find the most appropriate one for our model. It is well-known that in this model, the greater the number of programs running, the more the congested routes are accurately detected [18]. Hence, each vehicle is able to avoid the congested path and take the least amount of travelling time. The performance of our proposed method is evaluated using two metrics; average travelling time delay and average waiting time delay.

According to the simulations results, it is obvious that by adopting our proposed method, both average travelling time delay and average waiting time delay can be reduced drastically. Among the 3 neural network algorithms under consideration, MLP provides the lowest average travelling time delay and average waiting time delay.

The rest of this paper is structured as follows: In section 2, we describe the relevant literature review. Section 3 presents the proposed algorithms and the simulation set up in detail. Section 4 presents the deep reinforcement learning implementation. Section 5 shows the simulation results as well as detailed discussion. Finally, we conclude our work and future work in section 6.

2. RELATED WORKS

Recently, a large number of diversified researches to improve the transportation in the city have been carried out [3], [13], [15], [18], [19], [20], [24], [25], [26], [28]. In this section, the related works regarding VANET as well as traffic simulators and deep reinforcement learning for VANET are presented.

Over the past few years, VANET routing protocols have been studied to improve efficiency and reliability of the packet delivery rate, accident and traffic congestion avoidance and many other services. In [20], the topology-based routing protocols, AODV, AOMDV, DSDV and DSR, are simulated to compare the performance. It is obvious that DSR has the highest average throughput and the lowest average time delay. However, DSDV outperforms the others in terms if the highest packet delivery ratio.

In [3], the position-based routing protocols are studied. It is found that they are suitable in considerably changing environment. Each protocol requires different support of the performance and demands. For example, an extra agency such as street map is usually required by GSR. Both GPSR and B-MFR are used for highway traffic while many others are more suitable for city traffic. Lastly, a high delivery rate can be achieved using IGRP and RBVT.

In [10], a new position-based routing protocol called Efficient Routing Algorithm (ERA) is presented. The shortest and stable path can be generated by predicting the future position of vehicle in SDN Internet of vehicle. The ERA has better packet delivery ratio comparing with some other protocols, such as GPSR and AODV.

VANET is another topic that recently has drawn a lot of attention in research community, especially VANET adopting deep learning technique. In [28], the VANET and artificial neural network that are used in driverless cars to prevent malicious attacks and increase the security on vehicles are studied. The attacks on VANET could be accomplished by different ways: DOS attacks, impersonation attacks, brute force attacks, jamming the communication channel, etc. The implementation of deep learning makes it safer due to attacks on VANET being blocked. Moreover, the probability of vehicle collision could also be decreased.

There are some researches about using deep learning method on VANET for traffic prediction [10], [22]. In [10], the deep learning for predicting the future position of vehicles in dynamical environment is proposed. The model is trained based on real-time data from the vehicles, such as speed, direction, time, velocity, etc. The Efficient Routing Algorithm (ERA) is used for communication, which results in a large number of stable paths for packet delivery rate. The future position of vehicles is predicted based on the information from V2V and V2I communication with respect to the current position. After that, the predicted topology will be updated in order for the packets to be sent smoothly within the expected time. On the other hand, the traffic prediction model that is studied in [22] focuses on forecasting the real-time traffic flow

in the city. Instead of solving packet delivery rate, the work in [22] aims to solve the traffic congestion problem using deep convolutional neural network.

At the same period, research on traffic congestion avoidance using deep learning is carried out [18]. The simulation of all traffic scenarios is performed in grid network generated by using a software called SUMO (Simulation of Urban Mobility). As a result, when the congested routes are detected during the vehicle's trip to their destinations, those vehicles would be allowed to change their path from their current locations.

Deep reinforcement learning architecture is developed by merging the artificial neural networks with the reinforcement learning model. It has an agent which is capable to learn the most possible appropriate actions in any environment in order to achieve its aims. It solves the complex problems by correlating immediate actions before receiving the delayed returns as rewards from environment. Its algorithms sometimes have to wait a while to see the consequence of actions over multiple simulation steps [8], [29].

The topics about deep reinforcement learning have been studied since a few years ago [2], [23]. The deep reinforcement learning is introduced in [2] to develop an intelligent agent and being set up at the RSU (Road Side Unit) so that it would learn and understand a policy from multi-dimensional inputs. The agent derives reliable representations of the environment and learns from its experiences in the past to build a safe vehicular network to succeed admissible levels of Quality of Service (QoS).

In the following years, the reinforcement learning method to optimize vehicle routes in SUMO is proposed in [11]. This study shows that a vehicle can be trained by the reinforcement learning model to become more intelligent to avoid congested routes and arrive at the destination properly. The research using deep reinforcement learning for controlling the traffic light in a wide-scale grid network is studied in [23]. By adopting the deep reinforcement learning method, a signal-control policy with wide-ranging discrete actions is performed. As a result, the queue length and waiting time delay of vehicles on this type of network during traffic congestion are reduced.
Any tasks performed by deep reinforcement learning algorithms are anticipated to gradually become more and more reliable in uncertain actual environments while taking many possible actions. It has been projected to be precious to reach ambitions in the real-world industries, which might lead to the potential steps to the powerful AI.

3. PROPOSED ALGORITHMS AND SIMULATION SET UP

3.1. Proposed Algorithm

In this section, our proposed algorithm, the traffic prediction model using deep reinforcement learning model in VANET is described and illustrated by flowchart diagram in figure 1.

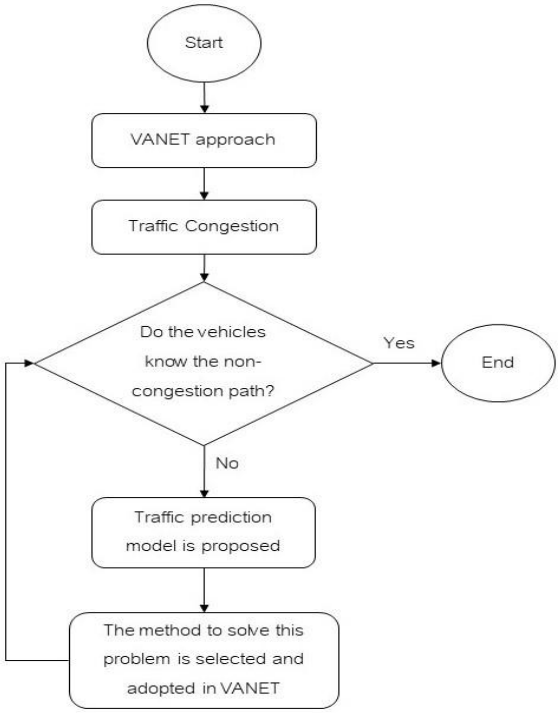

Figure 1. Flowchart Diagram of our Proposed Algorithm

As shown in figure 1, VANET can exchange information regarding of traffic congestion and utilize this information to avoid the congested paths. However, when the vehicles receive the information, it has already been too late to change the route. Therefore, the traffic prediction model is proposed to handle this limitation. In addition, the vehicles need to make decision to change the paths intelligently. To overcome this problem, deep reinforcement learning techniques are proposed. The details of traffic prediction model and deep reinforcement learning model are described in great detail in the subsection later on.

3.2. Traffic Simulator

In this work, the traffic simulator called SUMO (Simulation of Urban MObility), which is a space-continuous and time-discrete traffic flow simulation platform is used. It is a traffic simulator that helps organizing and simulating the traffic scenarios. The SUMO real-world road networks consist of nodes representing the intersections, and edges representing the roads. The edges are the connections of two nodes consisting of a set of lanes, while the intersections include shapes, positions, or traffic signals. A lane consists of a vehicle's information, geometry and maximum velocity [7], [12].

SUMO has been developed considerably since it was first introduced. According to the study of SUMO in [12], it mainly focused on SUMO road-network building using different methods, traffic flow, vehicle-to-anything communication, as well as the most recent development in that year, e.g. emission and noise modeling. On the other hand, in the present day, its capacity is extended significantly, and more features have been added. As a result, SUMO has already been able to import new data formats (i.e. OpenDRIVE), model networks with left-hand traffic, import public transport information, create 3D topology, edit additional simulation infrastructure and generate multi-modal networks for vehicles, bikes and pedestrians [7], [16]. Moreover, there is some planned features, such as adding and merging of sub-networks into the existing network [7].

In some previous works [11], [23], SUMO road networks are generated in the form of traffic grid networks, as shown in figure 2, using an application named "netgenerate". However, several works [10], [19], [25] including this one, require more realistic road networks. Therefore, it is necessary to import a digital road map, called Open Street Map (OSM), before using "netconvert" to convert it into SUMO-XML file [1], [21], which is the real-world road networks, as shown in figure 3.

Each vehicle in SUMO is classified by its departure time from source, its route through the network and some other identities. A vehicle's route contains a list of connected edges from vehicle's starting point to its destination. Each of which can also be set up using some properties; for example, the lane, the velocity and the exact position on an edge that it uses. Moreover, some other types, i.e., the vehicle's physical properties, the variables of its mobility, as well as the pollutant emission and fuel consumption classes can also be assigned to each vehicle. Furthermore, the vehicle's appearance within the GUI (Graphical User Interface) of the simulation can be defined by additional variables.

If traffic simulation contains a few vehicles in a small area [1], [11], [23], the vehicle's route and vehicle's properties can be set manually. In contrast, the traffic simulation of a large city, e.g. [10], [19], [25], which may contain more than a million vehicles, will be hard to manually define the traffic demand. Therefore, the use of applications in SUMO is needed to create the route of each vehicle.

For each simulation, SUMO generates the number of outputs in XML format, including each vehicle's position in each simulation step, inductive loops, information of all roads and lanes and travelling trip information for each vehicle. Not only the measurement of conventional traffic, but SUMO also has additional options for fuel consumption, together with pollution or noise emission model.

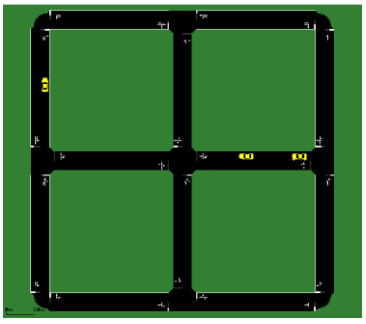

Figure 2. Traffic grid network

Figure 3. Real-world road network

A brief process when traffic congestion occurs, as shown in figure 4, can be described as follows:

- Any vehicle entering the road is assumed to have random source and random destination.
- The information of each vehicle is passed to traffic congestion prediction model provided by deep reinforcement learning module to get the future topology and non-congested route.
- Later on, if there is a sign of congestion, the vehicle will be rerouted to the alternative paths.
- Otherwise, the vehicles could go to the same route that they have already selected at the beginning.
- After all vehicles arrive at their destinations, the average travelling time delay and average waiting time delay are calculated.

However, in the first program running, it might not be able to choose the best path which results in the smallest amount of travelling time delay and waiting time delay. Therefore, the program will be run more than once. As the data of action taken is memorized in the model, the higher number of programs running will increase the prediction efficiency of the congested path, which results in the decrement of the travelling time delay and waiting time delay. This process is kept repeating until the time delay does not decrease, which means it reaches the minimum value.

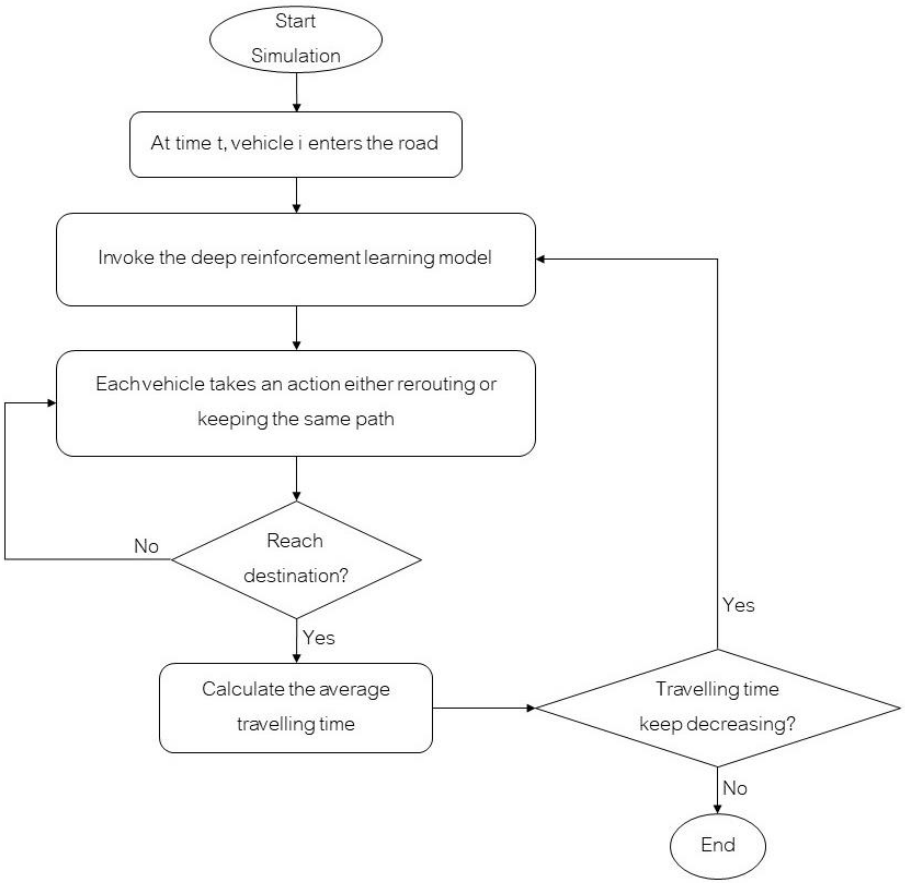

Figure 4. Flowchart of traffic congestion prediction concept

3.3. Simulation Setup of Real-world Road Network

To simulate road traffic, SUMO (Simulation of Urban Mobility) is used for handling large road networks. Firstly, an area in the real map is selected and imported into SUMO in order to generate traffic scenario before being integrated with deep reinforcement learning model.

In this work, a digital street map named OSM (Open Street Map) [6] is accessed to locate the edges and nodes of road to provide high accurate results. OSM is an open-source software of street level map of the world, created by the community of mappers. The area and several streets, i.e., lanes, main roads, motorways, footways, railways, subways, traffic lights, bridges, and intersections, etc. could be searched and selected. To perform the simulation, the city map with the main highway is exported from OSM into map.osm file, which is a text file written in XML-formatted data containing information of nodes, streets and tags (object properties).

3.4. Traffic Scenario Creation

Figure 5 shows the flowchart of SUMO configuration file creation with city map from OSM. The procedure to create the map can be described step-by-step as follows:

- After digital road network from OSM has been imported into map.osm file, SUMO-road network (map.net.xml) can be generated using NETCONVERT in command window.
- Next, run the python script which is provided in the SUMO tools (randomTrips.py). This script contains random() function to randomly generate the traffic routes (map.rou.xml). The data in the route file includes names of edges (street) where vehicles pass and the departure time of each vehicle.
- Then, the additional file (typemap.xml) containing obstacles i.e., land, water, building, residential, etc., is also created and converted to map.poly.xml using POLYCONVERT to add all these obstacles into map.
- Finally, SUMO configuration file (map.sumocfg) is created, which is the file where the inputs are given. Simulation time can be set in this step. Moreover, it indicates the files which should be selected to perform road network, vehicles' route and object properties. In this simulation, the files called map.net.xml, map.rou.xml and map.poly.xml, that have already been generated in the previous steps, are selected. With the configuration file, the traffic scenario can be simulated.

As a result, the traffic scenario is created.

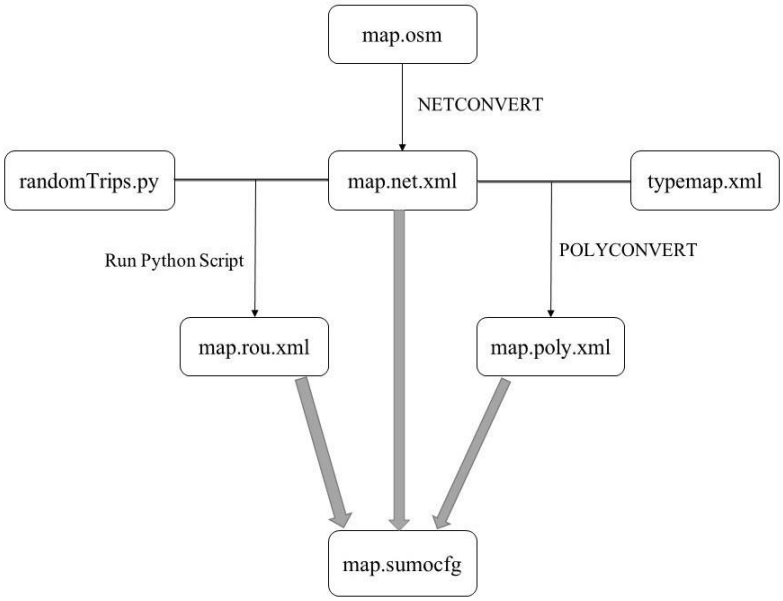

Figure 5. The creation of SUMO configuration file with city map from OSM

Generally, the SUMO simulator operates a default routing technique called DuaRouter, the specific function in SUMO, producing a vehicle-routing file (map.rou.xml). To build the routes using DuaRouter, a road network (map.net.xml) is created and traffic demands, such as trip and flow definitions have to be stated. The demand definitions given by the source and the destination edges help computing the shortest route for dynamic user assignment (DUA). However, if SUMO is approached by deep reinforcement learning, this SUMO default routing method will be replaced by a new more optimal and intelligent strategy that the agent has learnt in deep reinforcement learning algorithm.

SUMO simulator could be controlled by using TraCI library, while connecting to the library allows for a deep learning approach [27]. In this work, Python programming language is used for controlling both TraCI and deep reinforcement learning. A tool that helps deep reinforcement learning model being integrated into SUMO is PyCharm, the well-suited integrated development environment (IDE) for computer programming like Python.

3.5. Simulation Procedure

Figure 6 illustrates the flowchart of simulation procedure in this work. The whole process is controlled by Python programming language using TraCI library. Anyway, it is necessary to set up the traffic simulation and create map.sumocfg file before starting this process which is explained in detail as follows:

- To run SUMO simulation, the simulator has to be opened by TraCI library first. By using *traci.start(),* the configuration file (map.sumocfg) can be performed, resulting in SUMO window being shown up to start the traffic simulation.
- Then, getting a list of vehicles and edges ID using *traci.vehicle.getIDList()* and *traci.edge.getIDList()* method.
- Each vehicle in the list will enter the road in different time step. This is the time that deep reinforcement learning model start remembering the vehicle's action.
- The vehicles which are travelling to the congested path will take an action either rerouting to the alternative road or changing lane. The step can be carried out using

traci.vehicle.rerouteTraveltime() when it needs to reroute, and *traci.vehicle.updateBestLanes()* when it needs to change lane. The vehicles can choose the new routes or lanes, also known as edges, in a list of edge ID it got from the previous step.

- After every vehicles finish making decision and arriving their destination, the traffic simulation will stop performing.
- Using *traci.edge.getTraveltime()* and *traci.edge.getWaitingTime()* method, the travelling time and the waiting time of vehicles on each edge can be returned once the vehicles arrive at the next edges. When the simulation ends, the final average travelling and waiting time can be calculated.
- Later on, before SUMO simulation is ended, deep reinforcement learning model will learn the experience based on the value of reward it gets and analyze the vehicle's decision using Q-Learning algorithm (described in section 4.3). To calculate the reward, *traci.edge.getLastStepVehicleNumber()* is used for returning the total number of driving vehicles, and *traci.edge.getLastStepHaltingNumber()* is used for returning the total number of waiting vehicle (The further details will be provided in section 4.3).
- Lastly, set the number of program runs to 15, which is enough for the model to perform at its best, bringing about the travelling time and the waiting time to stop decreasing. If the number of program runs is lower than 15, SUMO window will be opened again, and the whole process will be repeated.

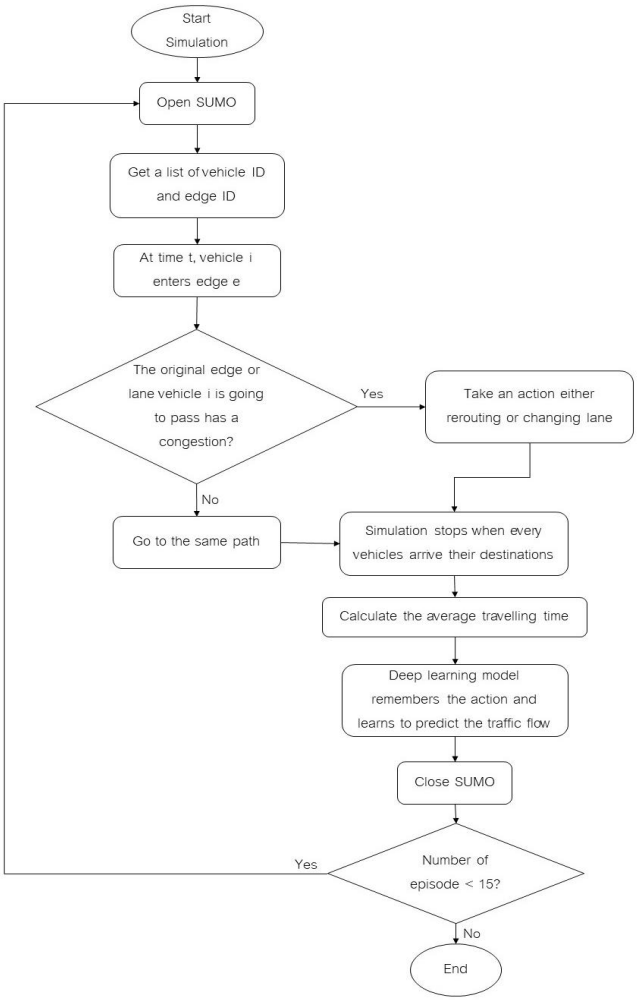

Figure 6. Flowchart diagram of simulation procedure

4. DEEP REINFORCEMENT LEARNING IMPLEMENTATION

In this work, it is assumed that the communications between vehicles and infrastructures (V2I) and among vehicles (V2V) taking place on the road. Generally, road traffic data that is exchanged between vehicles and roadside units will be useful for deep reinforcement learning. As the scenario has already been conducted in SUMO, traffic information can be gathered.

All deep reinforcement learning models use the similar concepts of agents, environments, states, actions and rewards. In figure 7, the deep reinforcement learning process is illustrated. The element called environment is responsible to convert an action taken in the particular state into a new state and reward. After that, the result values will be altered again by an agent which transforms them into the next action. An agent attempts to take actions in the environment, providing the maximum rewards.

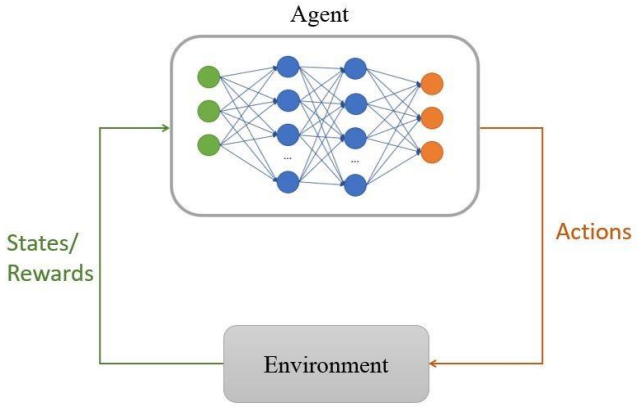

Figure 7. Deep reinforcement learning

4.1. Environment Model

The environment is the world where the agent performs an operation. In this case, it is the road network created by SUMO. To build an environmental model for deep reinforcement learning, TraCI library is used in PyCharm to connect to SUMO and get road network information, i.e., edges, lanes and nodes. After getting this data, multidimensional arrays and matrices of road and junction position, along with velocity, are created using Numpy which is the library for high-level mathematical functions operated on these arrays.

4.2. Agent Model

The agent has a role to make decisions based on the rewards and observations. Figure 7 also shows the agent's inside where neural networks are built. Figure 8, which shows the agent's inside, consists of neural network algorithms, namely CNN, MLP and LSTM. It can be seen that the input layer receives the velocity matrices and position matrices (longitudes and latitudes) as states from the environmental model, while at the output layer, the possible actions are decided by the agent, based on the rewards it receives.

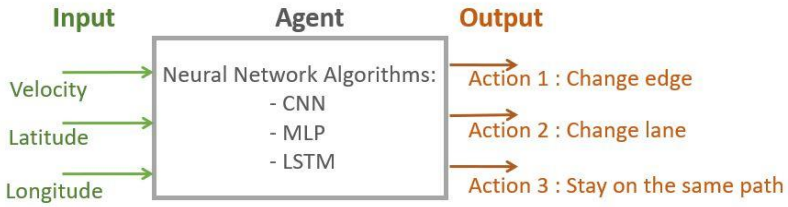

Figure 8. Deep reinforcement learning agent

4.3. Q-Learning

Before the agent can take the best actions, the Q-learning model is required. It is a model for the reinforcement learning algorithm to understand the procedure to inform an agent what action or decision it needs to perform [14]. The update rule that Q-learning uses is the Bellman equation for the optimal Q-value [5]. In this step, the values of the next state are predicted in order to calculate the target value for state *s*, if specific action *a* is taken. The target value, denoted as $Q(s,a)$, can be computed using the following equation:

$$Q(s,a) = R(s,a) + \gamma.max\hat{Q}(s,a) \quad (1)$$

where γ is the discount rate, and $max\hat{Q}(s,a)$ is the maximum expected future reward given the new state and all possible actions at that new state. $R(s,a)$ represents the reward for taking that actions at that state as expressed by the following Equation:

$$R(s,a) = n(Driving) - n(Waiting) \quad (2)$$

Here, $n(Driving)$ is the total number of driving vehicles for each time step, and $n(Waiting)$ is the total number of waiting vehicles (or the vehicles that have the speed below 0.1 m/s) for each time step.

4.4. Algorithms of Neural Network Used in the Proposed Model

To build neural network layers, an open-source neural network library called *Keras* is used. The reward function is measured by the total number of vehicles and the number of waiting vehicles in certain roads. The agent will choose to take an action which results in the higher value of reward. *TraCI* library is necessary to run the program with deep reinforcement learning model in SUMO.

In this work, three different types of neural networks are investigated and compared to find the most effective one for traffic scenario:

- Multilayer Perceptron (MLP): Containing input, output and 3 hidden layers, all of which are fully connected; therefore, the Density function of *Keras* is used. In the hidden layers, each layer contains 128, 64 and 32 units, respectively.
- Convolutional Neural Network (CNN): Containing convolutional layer at the beginning, and the layers with fully connected at the end. This algorithm uses Conv2D function to create two-dimensional convolutional layers with 16-4×4 filters. It is followed by Flatten function to convert the data from a vector of two-dimensional matrices into the correct format for dense layers (fully connected layers) which have 64 and 32 units, respectively before reaching the output layer.

- Long Short-Term Memory (LSTM): Unlike standard feedforward neural networks, LSTM has back-propagation connections. This algorithm requires an Embedding function with a maximum number of words = 2500 and embedding dimension = 128 for mapping discrete objects to vectors and real numbers. It is followed by 200-unit LSTM function and a 32-unit Density function.

4.5. Complexity of the Proposed Algorithms

The complexity of our proposed algorithms is also analyzed using Big O notation, which is the method using to describe the complexity and execution time required by an algorithm [9]. In this deep reinforcement learning model, the maximum time that the algorithms can perform is in cubic time written as $O(n^3)$, where n is the size of inputs. That means the running time is proportional to the cube of the input size. In figure 9, the Big-O complexity of this algorithm in the operation unit ($\times 10^9$) is illustrated. The input size is the number of vehicles in the traffic simulation.

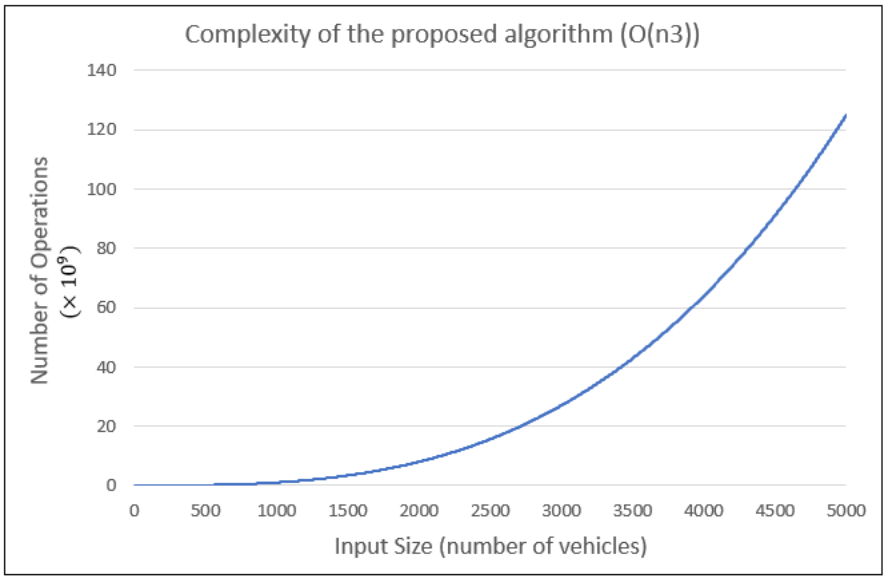

Figure 9. Complexity of the proposed algorithm ($O(n^3)$)

4.6. Simulation Parameters

All the environment parameters are listed in Table 1. The network simulation parameters used in this system are shown in Table 2.

Table 1. Simulation Environment

Parameters	Environment
Traffic Simulator	SUMO v 1.2.0
Geographical Information System (GIS)	Open Street Map (OSM)
Deep Learning Module	TensorFlow v 2.0
Computer Language	Python v 3.6
Integrated Development Environment (IDE)	PyCharm

Table 2. Simulation Parameters

Parameters	Values
Simulation Time	10,000 seconds
Number of Vehicles	500 – 5,000
Vehicle Velocity	0 - 80 m/s
Map Size	1,000 m × 1,000 m
Area	Phra Nakhon District, Bangkok, Thailand
Deep Learning Algorithms	CNN, MLP and LSTM

4.7. Performance Evaluation Metrics

The performance of our proposed algorithm is evaluated by the following metrics:

- Average Travelling Time Delay: The average time spent by all the vehicles under consideration to travel from starting point to destination.
- Average Waiting Time Delay: The average time duration that all vehicles under consideration have no movement or move with speed lower than 0.1 m/s.

5. RESULTS AND DISCUSSION

In this section, the simulation results are illustrated with 95% confidence interval.

5.1. Average Travelling Time Delay

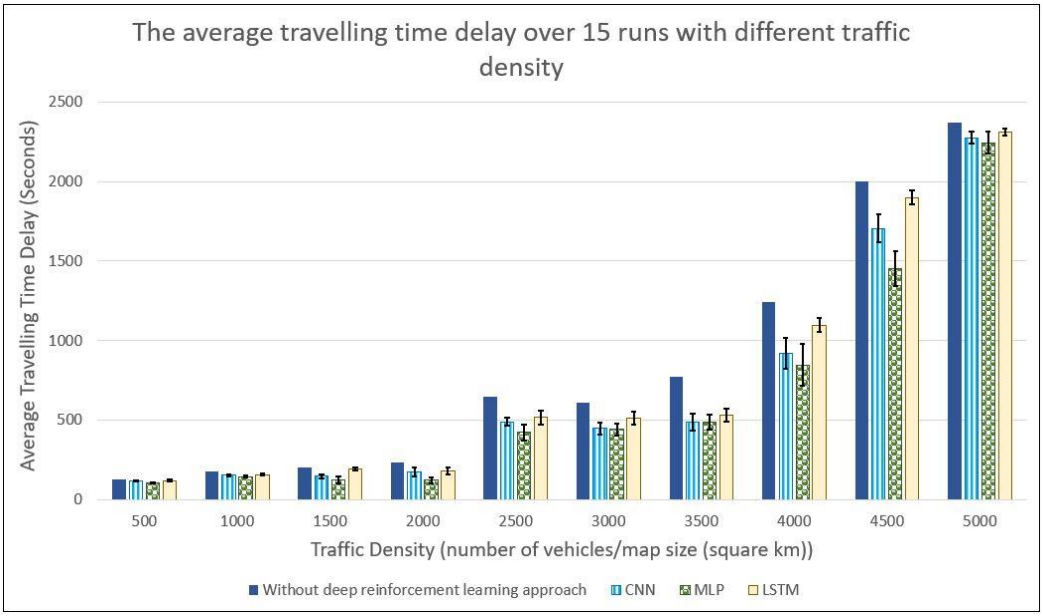

Figure 10. Comparison of the Average Travelling Time Delay of all algorithms with and without deep reinforcement learning approach over 15 runs

Figure 10 shows the average travelling time delay of all algorithms with and without deep reinforcement learning approach over 15 runs versus the traffic density. According to the results, it is clear that the comparison results are not much different when traffic density is low, since the congestion does not occur at this level of traffic density. The vehicles drastically begin to take more travelling time when traffic density is 2,500 vehicles/km^2. At this point, the differences

between the results with three deep learning and those results without deep learning algorithms are obvious. The deep reinforcement learning model apparently becomes more effective when the traffic density is around 3,500 – 4,500 vehicles/km^2. If the traffic density continues to increase until it reaches 5,000 vehicles/km^2, the rerouting seems to be less possible, since there are not enough available roads to move. Based on the results shown in figure 10, it can be concluded that the most effective deep learning algorithm for traffic congestion prediction model is MLP, followed by CNN, while the least effective one is LSTM.

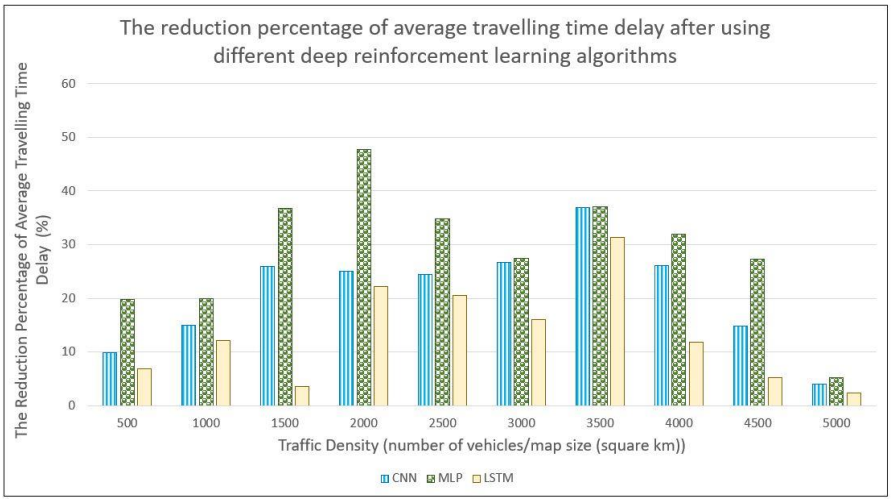

Figure 11 Comparison of the reduction percentage of Average Travelling Time Delay after using three deep reinforcement learning algorithms

In Figure 11, it shows the reduction percentage in average travelling time delay after using three different deep reinforcement learning algorithms. It is obvious that MLP provides the highest percentage of average travelling time reduction among all three algorithms.

5.2. Average Waiting Time Delay

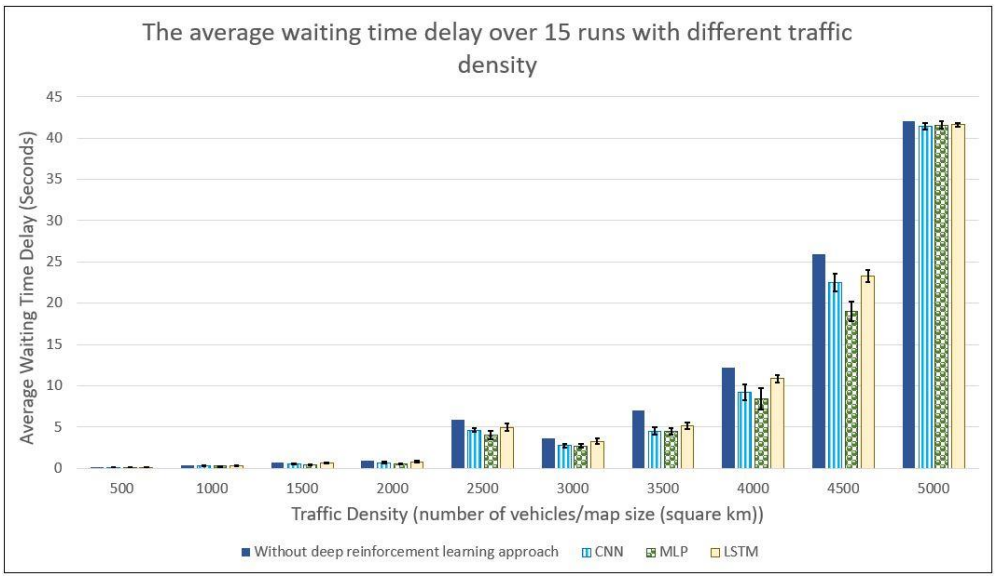

Figure 12. Comparison of the Average Waiting Time Delay between the algorithms with and without deep reinforcement learning approach over 15 runs

Figure 12 shows the average waiting time delay with and without deep reinforcement learning approach over 15 runs versus traffic density. It is apparent that the average waiting time delay has similar trend as the average travelling time delay in figure 10. The comparison results illustrate explicitly the effectiveness of the algorithms when traffic density is more than 2,500 vehicles/km^2, which MLP algorithm is still the most effective one.

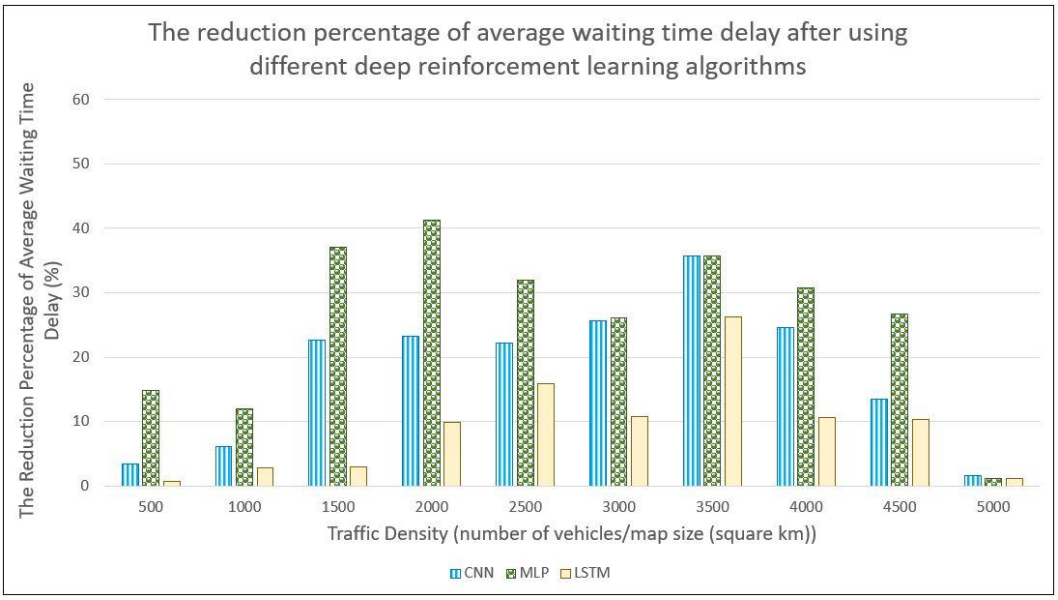

Figure 13. The reduction percentage of Average Waiting Time Delay after using three deep reinforcement learning algorithms

Figure 13 illustrates the comparison of reduction percentage in average waiting time delay after adopting three deep reinforcement learning algorithms under consideration. It is obvious that the results have the same trend as those in average travelling time delay shown in figure 11, where MLP and LSTM are the most and the least effective deep learning algorithm, respectively.

6. CONCLUSIONS AND FUTURE WORKS

In this work, deep reinforcement learning is adopted in VANET for traffic congestion prediction. In VANET, traffic information, such as vehicle's velocity, position, direction and movement pattern, can be exchanged through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication. This facilitates the machine to learn how to take actions to avoid congested paths. Deep reinforcement learning is a method that deep learning is combined with reinforcement learning. It helps solving the complex problems by taking action in the environment and getting feedback.

Before a deep reinforcement learning model is built, traffic simulation has to be set up by SUMO. The illustration of real-world roads can be generated by importing area from OpenStreetMap (OSM), which is used for creating SUMO road network. After that, vehicles' routes and additional objects in the map, such as traffic light are created. Then, traffic simulation can be performed and controlled by the SUMO configuration file. It will be later controlled by Python programming language and TraCI library of SUMO to integrate deep learning algorithms into traffic simulation.

In this traffic prediction model, the results of average travelling time delay and average waiting time delay are gradually improved better and better over the multiple runs. This is because the vehicles are able to learn how to choose the routes based on the rewards from the previous states. In addition, the simulation results without and with three different deep learning algorithms, i.e., Convolutional Neural Network (CNN), Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM), are compared. It is obvious that deep reinforcement learning model works effectively when traffic density is neither too high nor too low. Low traffic density means that there is no congestion on the roads, which is not necessary for rerouting. On the other hand, in highly dense traffic, there might be no available path for rerouting. Based on the comparison results among those three algorithms, it can be concluded that the effective algorithms for traffic congestion prediction model in the descending order are MLP, CNN, and LSTM, respectively. Regarding the complexity of our proposed algorithm, the complexity is analyzed and shown using Big-O notation. The complexity and maximum execution time required by our proposed algorithm is in cubic time written as $O(n^3)$, where *n* is the size of inputs. That is, the running time is proportional to the cube of the input size.

In future works, the traffic prediction model may be developed to become more efficient and more accurate by creating additional scenarios i.e., accidents and roadblocks, since they are also the causes of congestion on the roads. Moreover, vehicle-to-pedestrian (V2P) communication can be included. Besides, the model can be made more realistic by adding various types of vehicles, such as motorcycles and buses.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGEMENTS

The authors would like to thank the scholarship donor; His Majesty the King's Scholarship for providing the scholarship to study and carrying out this research.

REFERENCES

[1] Akhter S., Ahsan, M. N., Quaderi, S. J. S., Forhad, M. A. A., Sumit, S. H., & Rahman M. R. (2020) "A SUMO Based Simulation Framework for Intelligent Traffic Management System", *Journal of Traffic and Logistics Engineering*, Vol. 8, No. 1, pp.1-5

[2] Atallah, R., Assi, C., & Khabbaz, M. (2017) "Deep reinforcement learning-based scheduling for roadside communication networks*", The 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2017)*, Paris, France.

[3] Batish, S., Chahal, M., & Sofat, S. (2015) "Comparative study of position-based routing protocols in VANET*", Asian Research Publishing Network* (*ARPN*), Vol.10, No.15, pp. 6414–6418.

[4] Choi R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020) "Introduction to Machine Learning, Neural Networks, and Deep Learning", *Translational Vision Science & Technology (TVST)*, Vol. 9, No. 2, pp.1-12.

[5] Choudhary, A. (2019) "A Hands-On Introduction to Deep Q-Learning using OpenAI Gym in Python*",* https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-python/.

[6] Contributors, OpenStreetMap "OpenStreetMap", https://www.openstreetmap.org/. Geofabrik GmbH: Karlsruhe, Germany.

[7] Contributors, SUMO Simulator "SUMO User Documentation", https://sumo.dlr.de/docs/.

[8] Fran¸cois-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018) "An introduction to deep reinforcement learning", *Foundations and Trends in Machine Learning, Cornell University*, Vol.11, No. 3-4.

[9] Huang, S., (2020) "What is Big O Notation Explained: Space and Time Complexity", https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-it-doesnt.

[10] Jibran, M. A., Abbass, M. T., Rafiq, A., & Song, W.-C. (2020) "Position prediction for routing in software defined internet of vehicles", *Journal of Communications*, Vol.15, No.20, pp.157-163.

[11] Koh, S. S., Zhou, B., Yang, P., Yang, Z., Fang, H., & Feng, J. (2018) "Reinforcement learning for vehicle route optimization in SUMO*", The 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems, Exeter, United Kingdom*, pp.1468-1473.

[12] Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012) "Recent development and applications of SUMO - Simulation of Urban MObility", *International Journal on Advances in Systems and Measurements*, Vo.5, No. 3 & 4, pp.128-138.

[13] Kumar, R., & Dave, M. (2011) "A comparative study of various routing protocols in VANET" *International Journal of Computer Science Issues (IJCSI)*, Vol. 8, Issue 4, No. 1, pp.643-648.

[14] Kwong, Y., Bolong, N., Kiring, A., Yang, S. S., Tze, K., & Teo, K. (2011) "Q-learning based traffic optimization in management of signal timing plan", *International Journal of Simulation: Systems, Science and Technology (IJSSST)*, Vol.12, No. 3, pp.29-35.

[15] Liang, W., Li, Z., Zhang, H., Wang, S., & Bie, R. (2015) "Vehicular ad hoc networks: Architectures, research issues, methodologies, challenges, and trends", *International Journal of Distributed Sensor Networks*, Vol. 11, No. 8, pp.1-11.

[16] Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann J., Flötteröd, Y., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., & Wießner, E. (2018) "Microscopic Traffic Simulation using SUMO", *International Conference on Intelligent Transportation Systems (ITSC)*, Maui, Hawaii, USA, pp.2575-2582.

[17] Mehta, A. (2019) "A Comprehensive Guide to Types of Neural Networks", https://www.digitalvidya.com/blog/types-of-neural-networks/.

[18] Perez-Murueta, P., G´omez-Espinosa, A., Cardenas, C., & Gonzalez-Mendoza, M. J. (2019) "Deep learning system for vehicular re-routing and congestion avoidance", *Journal of Applied Sciences: special issue Artificial Intelligence Applications to Smart City and Smart Enterprise*, Vol.9, No. 13. pp.1-14

[19] Rakkesh, S., Weerasinghe, A., & Ranasinghe, R. (2016) "A decentralized vehicle re-routing approach using vehicular ad-hoc network", *The 16th International Conference on Advances in ICT for Emerging Regions (ICTer2016)*, Negombo, Sri Lanka, pp.201-207.

[20] Sanwal, R., & Kumar, V. (2013 "Simulation based evaluation of proactive and reactive routing protocols in realistic vehicular network", *International Conference on Recent Trends in Computing and Communication Engineering - RTCCE 2013*, Hamirpur, India, pp. 127-131.

[21] Singh, S., Saraswat, A., & Yadav, S. (2019) "Traffic simulation integration using SUMO simulator", *International Journal of Scientific Research and Review*, Vol.7, No. 2, pp. 22–26.

[22] Sun, S., Wu H., & Xiang, L. (2020) "City-Wide Traffic Flow Forecasting Using a Deep Convolutional Neural Network", *Multidisciplinary Digital Publishing Institute (MDPI)*, Vol.20, No. 2, pp.1-15.

[23] Tan, T., Bao, F., Deng, Y., Jin, A., Dai, Q., & Wang, J. (2020) "Cooperative deep reinforcement learning for large-scale traffic grid signal control", *In IEEE Transactions on Cybernetics*, Vol.50, Issue 6, pp. 2687-2700.

[24] Saif Al-Sultan, G. Calandriello, Y. Mak, Tracy Ann Kosa, Stephen Marsh, H.Kim, J.Grover, M.Gaur, C. Lochert, B.Scheuermann, C.Wewetzer, A. Luebke, L. Nassar, M.Kamel, F. Karray, Ramu Panayappan, S. A. Khayam, J.J.Haas (2015) "Security and privacy in vehicular ad hoc network (VANET): A survey*", The IJCA Proceeding on National Conference on Advances in Computing Communication and Application (ACCA2015)*, pp.1-3.

[25] Wang, S., Djahel, S., & McManis, J. (2015) "An adaptive and VANETs-based next road rerouting system for unexpected urban traffic congestion avoidance", *The IEEE Vehicular Networking Conference (VNC2015)*, Kyoto, Japan, pp.196-203.

[26] Wedel, J. W., Sch¨unemann, B., & Radusch, I. (2009) "V2X-based traffic congestion recognition and avoidance*", The 10th International Symposium on Pervasive Systems, Algorithms, and Networks 2009*, Kaohsiung, Taiwan, pp.637-641.

[27] Wu, C., Parvate, K., Kheterpal, N., Dickstein, L., Mehta, A., Vinitsky, E., et al. (2017) "Framework for control and deep reinforcement learning in traffic", *The IEEE 20th International Conference on Intelligent Transportation Systems (ITSC2017)*, Yokohama, Japan, pp. 155-163.

[28] Ydenberg, A., Heir, N., & Gill, B. (2018) "Security, SDN, and VANET technology of driver-less cars", *The IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC2018)*, Las Vegas, NV, USA, pp.313-316.

[29] "A Beginner's Guide to Deep Reinforcement Learning", (n.d.) https://pathmind.com/wiki/deep-reinforcement-learning.

AUTHORS

Chantakarn Pholpol was born in Bangkok, Thailand in 1994. She received Bachelor's degree in Electrical Communication and Electronic Engineering from King Mongkut's University of Technology Thonburi in 2016. After graduation, she joined and had worked for two years as network engineer at United Information Highway Co., Ltd. In 2020, she earned degree of Master of Engineering in Telecommunications from Asian Institute of Technology.

Teerapat Sanguankotchakorn was born in Bangkok, Thailand on December 8, 1965. He received the B.Eng in electrical engineering from Chulalongkorn University, Thailand in 1987, M.Eng and D.Eng in information processing from Tokyo Institute of Technology, Japan in 1990 and 1993, respectively. In 1993, he joined Telecommunication and Information Systems Research Laboratory at Sony Corporation, Japan where he holds two patents on Signal Compression. Since October 1998, he has been with Asian Institute of Technology where he is currently an Associate Professor at Telecommunications Field of Study, School of Engineering and Technology. Dr. Sanguankotchakorn is a Senior member of IEEE and member of IEICE, Japan.