GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

TRAFFIC MANAGEMENT WITH AI (ARTIFICIAL INTELLIGENCE)

A Project Report of Capstone Project - 2

Submitted by

ANKIT KUMAR
(1613101142)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE

Under the Supervision of

Mr. Dhruv Kumar, Asst. Prof

May/June - 2020

GALGOTIAS UNIVERSITY

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report " TRAFFIC MANAGEMENT WITH AI" is the bonafide work of " ANKIT KUMAR (1613101142)" who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL,
PhD (Management), PhD (CS)
Professor & Dean,
School of Computing Science &
Engineering

SIGNATURE OF SUPERVISOR

Mr. Dhruv Kumar,
M.Tech.,
Asst. Professor
School of Computing Science &
Engineering

TABLE OF CONTENTS

Abstract

Learning-based traffic control algorithms have recently been explored as an alternative to existing traffic control logics. The reinforcement learning (RL) algorithm is being spotlighted in the field of adaptive traffic signal control. However, no report has described the implementation of a RLbased algorithm in an actual intersection. Most previous RL studies adopted conventional traffic parameters such as delays and queue lengths to represent a traffic state, which cannot be exactly measured on-site in real-time. Furthermore, the traffic parameters cannot fully account for the complexity of an actual traffic

state. The present study suggests a novel artificial intelligence that uses only video images of an intersection to represent its traffic state rather than using handcrafted features. In simulation experiments using a real intersection, consecutive aerial video frames fully addressed the traffic state of an independent 4-legged intersection, and an imagebased RL model outperformed both the actual operation of fixed signals and a fully actuated operation.

List of tables

List of figures

Introduction

An adaptive traffic signal control (ATSC) depends upon logic that ranges from the simplest example wherein green lights are assigned for each movement on a longest-queue-first basis to the most complex signal control system that depends on optimal control theory. Learning-based ATSC has recently emerged as an alternative to the existing ATSCs, and the reinforcement-learning (RL) algorithm is receiving the greatest share of the spotlight after it was used to solve many difficult dynamic problems. A RL agent can control traffic signals for a single intersection or a group of intersections in real-time by continuously improving its control performance. In addition, the RL-based model can be trained by self-generating data examples, unlike many supervised learning technologies that require a predetermined set of training data. Several studies (Abdulhai and Kattan, 2003; Bazzan and Klgl, 2014; Bazzan, 2009; Liu, 2007; Mannion et al., 2015) have reviewed the application of RL to solve traffic signal control problems, and have concluded that RL-based traffic control performs better in simulations involving various traffic control circumstances, when compared with the existing traffic control schemes. These studies also concluded that the RL technology will dominate traffic control in the near future. However, the reality is that intersection controllers with an RL-based algorithm are rarely found in the real world, despite a plethora of related studies based on simulation experiments.

The reason is two-fold. First, recognizing traffic states is a key to the communication between an RL agent and its surrounding environment. Such communication allows an agent to take the optimal action continuously (see the next section). Almost all previous RL studies have adopted conventional traffic parameters such as delays and queue lengths to represent the entire traffic state (El-Tantawy et al., 2014; Abdoos et al., 2011; Abdoos at al., 2014; Abdulhai et al., 2003; Arel et al., 2010; Bakker et al., 2010; Jin and Ma, 2015; and Isa, 2006). The noted exceptions are the few studies that have used vehicular positions from cellular automata simulation (Wiering, 2000; Khamis and Gomaa, 2014). However, both the traffic parameters and the vehicle position data cannot be accurately measured on-site in real-time under existing traffic surveillance systems. Installation of these models into the real world must wait until some form of vehicle-to-junction communication is available. However, a world where every vehicle will be equipped with an onboard unit for this type of communication is not expected any time soon. The reality is that the existing RL-based traffic control models make sense only for virtual simulation environments, although they chose real intersections as the target site for simulation.

Second, even if an accurate measure of the traffic parameters were possible, there remains the more fundamental question of whether known traffic parameters such as delays, queue lengths, throughput, travel time, etc., can fully account for a complex traffic state. Most researchers might believe that the conventional list of traffic parameters can fully represent a real traffic state. However, the reality is that much information beyond what the conventional parameters can explain has long been ignored in traffic engineering.

Recently, the recognition of computer vision has met with great success (Krizhevsky et al.,2012; Simonyan and Zisserman, 2014; and Szegedy et al., 2015). Deep-learning-based algorithms have outperformed, in part, the human ability to recognize human faces, animals, and objects.

The present study began with an expectation that a deep-learning model could be successful in recognizing traffic states, which should be much less complex than distinguishing human faces or recognizing medical CT images. The present study suggests a novel RL-based traffic control methodology that represents both traffic states and rewards based solely on aerial images of an intersection. To control traffic lights, the present study utilized cutting edge technology that combines a RL learning model with a deep convolutional neural network (CNN). This combination is being regarded as a main breakthrough for the future of artificial intelligence (Silver, 2015). The present study is organized as follows. The next section introduces the basics of RL-based control problems and a deep-learning model to illustrate the states of an environment. The third section explains how to set up an RL-based traffic signal controller for a 4-legged intersection. Results from simulation experiments include comparisons with both an actual fixed-signal operation and a fully-actuated operation, which are summarized in the fourth section. The last section draws conclusions and suggests further studies to apply the present methodology to the real world of traffic control.

Reinforcement learning and a convolutional neural network

The Markov decision process (MDP) is the target problem to be solved by RL. The MDP represents a mechanism wherein an agent takes an action continuously by following an optimal policy to maximize its expected future reward, while interacting with the environment affected by the action. The expected future reward is a function of both the current traffic state and the action taken under the state, which is called a Q-function, and more specifically represents an action-value function.

$$Q(s_t,a_t) = E[r_{t+1},x_{t+2},y^2 r_{t+3}+\ldots|s_t,a_t] \quad (1)$$

where, s_t is the state of an environment at a time interval t, a_t is an action that is taken at a time interval t, $r_{t+1}$is a reward after taking a_t , x is a discount rate for future rewards, Q() denotes the Q-function, and [] E is an expectation symbol.

The most important point in a MDP is how an agent perceives the state of the environment that surrounds it. Most researchers have abstracted the state of an environment by using only a few handcrafted features. Recently, Mnih et al. (2015) proposed a revolutionary method to force an agent to recognize the state of an environment by applying a RL algorithm to the playing of several classic video games. That study provided an agent with only raw video images at a pixel level to represent a game state without offering any other features or any other rules for playing the game. The raw image frames were just an input for a deep convolutional neural network (CNN). A deep CNN model was set up to approximate the Q-function from input images. The proposed Q-learning algorithm, a RL algorithm utilizing Q-function updates (Baird, 1995), falls into the category of "model-free" and "off-policy" learning algorithms. "Model-free" means that the transition probabilities between states are unknown in the MDP to be solved, and "off-policy" refers to learning about a greedy policy ($a=\arg\max Q(S_{t+1},a`|w)$) along with a random selection at a certain exploration rate. On the other hand, "on-policy" learning algorithms such as SARSA and SARSA (lemda) require the next action to be chosen from a given policy. Namely, Q-learning algorithm updates the action-value function regardless of the current action, but with respect to the action that maximizes the value of the next state-action pair, while SARSA learns the value of the state-action function on the basis of the performed action (Corazza and Sangalli, 2015). However, when an agent always chooses the best action, these two algorithms converge to the same solution.

The present study adopted a Q-learning algorithm in accordance with Minh (2015) that has shown great success in implementing classic computer games. The details of Q-learning procedures will be addressed in the next section. The loss function of a Q-learning algorithm that can be minimized can be expressed in a very simple least-square form as in Eq. (2). When the loss function is minimized, the Bellman equation holds. Thus, the Q-learning algorithm converges to the condition of a Bellman equation.

$$L(w)=E[(r_{t+1}+\gamma\max Q(s_{t+1},a`|w) - Q(s_{t+1},a`|w))^2] \quad (2)$$

where,Q(s,a|w) denotes an approximated Q-function, and, thus, w is a set of weight parameters that are necessary in order to parametrize the original Q-function.

The stochastic gradient descent (SGD) algorithm is known to work well for minimizing the loss function. A SGD algorithm is different from the general gradient-descent algorithm in that it repeatedly updates parameters of the objective function by using a gradient that is computed based on either a single example or a subset of samples (=mini-batch). It is well known that repeating the update of weight parameters based on a gradient derived from a single training example (or mini-batch sample) guarantees reaching the global minimum if a large amount of training examples is available and they are reshuffled for every epoch. A SGD algorithm is the best fit to minimize the loss function of a MDP problem with an incumbent pair $\gamma\max Q(s_{t+1},a`|w)$ at each time interval, regarding the pair as a single example of both feature and label. That is, the algorithm made it unnecessary to evaluate the objective function [Eq. (2)], which requires computing the average of ($r_{t+1}$+ymax Q($s_{t+1}$,a`|w) - Q(s_{t+1},a`|w))^2 across all possible combinations of < s_t ,a_t, r_{t+1},$s_{t=1}$ >.

Unfortunately, a Q-learning algorithm does not always guarantee convergence to a global minimum when the Q-function is approximated by a nonlinear function, and, instead, is more likely to be stuck in a local minimum. This instability stems from the fact that consecutive states of a MDP are likely to be correlated in nature, which might violate the condition of a SGD that requires a random order of data examples. An RL-based algorithm has the advantage of receiving training examples automatically as the algorithm is running. On the other hand, there is a drawback whereby the examples cannot be reshuffled in advance. Another cause of the instability is the correlations between the action-values and the target values. To tackle these problems, the present study adopted three effective measures proposed by Mnih et al. (2015).

First, the target Q-function was separated from the main Q-function to be updated. Weight parameters of the target Q-function were not updated at every time interval, but were fixed until they could be updated on a long-term basis. When updating the target Q-function, the weights were set identical to those of the main Q-function at that time interval. Thus, in Eq. (3), weight parameters of the target Q-function were fixed values rather than determinant variables:

$$L(w)=E[(r_{t+1}+y\max Q(s_{t+1},a`|w) - Q(s_t,a_t|w))^2] \quad (3)$$

where w_ is the weight parameter of the target Q-function to be updated on a long-term basis.

Second, rather than immediately utilizing the state-transition data < s_t ,a_t, r_{t+1},$s_{t=1}$ >. acquired at each time interval, the data were added to a replay memory with a predefined size. At every time interval, a certain number of state-transition examples were chosen randomly from the replay memory to update the weights. Therefore, the algorithm could avert a correlation between consecutive environmental states. The last tip for guaranteeing the convergence of the algorithm was to confine the reward value to either +1 or -1, which prevented the objective function value from either diverging or oscillating.

An RL algorithm with a deep CNN model will include several hyper-parameters that affect the control performance. These hyper-parameters include the number of convolution layers, the number of filters for each convolution layer, the filter and stride size for each convolution layer, the number of hidden layers next to the last convolution layer, and the number of nodes (=neurons) within each non-convolution layer. In the present study, the hyper-parameters of a deep CNN were not determined in a rigorous manner such as the use of a grid search, because of the burden of computing time. Instead, these were chosen on a trial-and-error basis. The starting point to determine the hyper-parameters was to refer to those used by Mnih et al. (2015). In addition, a useful trick was adopted for saving computation time. The output layer of the CNN had the same number of nodes as the number of available actions, so that a single feed-forward evaluation of a CNN simultaneously determined the Q-function values for all actions. In this way, there was no need to establish a separate CNN for each action

Reinforcement learning for traffic signal control

All previous trials to employ RL to control traffic lights overlooked the possibility of losing state information, when representing a traffic state with only a few traffic parameters such as traffic flow, delay, and queue length. Another problem arose when it became apparent that although these traffic parameters could sufficiently account for a traffic state, it was impossible to perfectly measure them using current surveillance systems. In the present study, these two problems necessitated the use of raw video shoots to represent a traffic state. A deep CNN was adopted to approximate the Q-function of an RL-based traffic signal control, which was based solely on the video images of an intersection.

Fig. 1 shows a MDP that was applied to the control of traffic lights at an intersection.

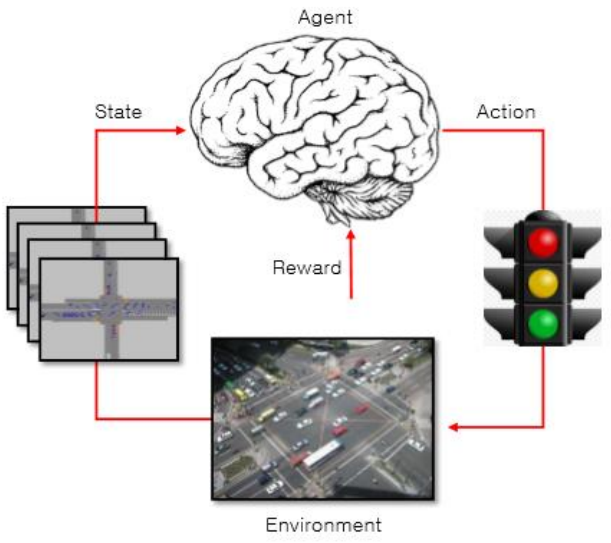


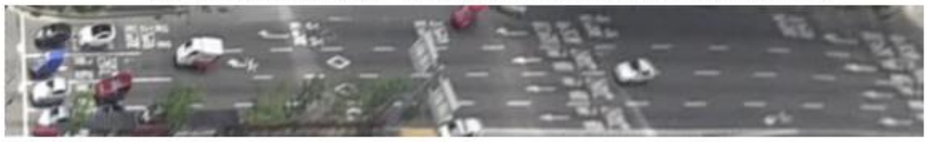
Figure 1. MDP for traffic signal control at an intersection

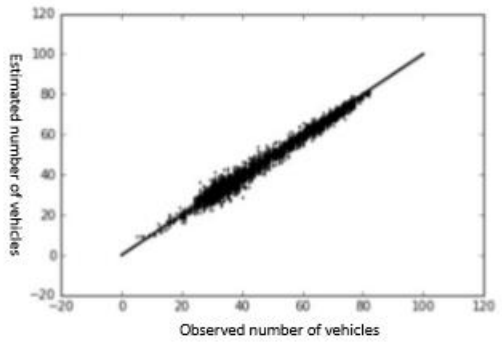
An artificial agent in charge of traffic signal control executed an action at the beginning of every time interval, interacting with the environment, so that the expectation of the discounted sum of future rewards would be maximized. In other words, the agent chose a signal phase at every time interval in an expectation that the intersection would be cleared in the future. The number of waiting and approaching vehicles (WAVEs) could be an appropriate index to derive an immediate reward where delay-related indices are not available. The best reward of an action (signal phase) taken would be a decrease in the maximum number of WAVEs across all lanes of an intersection, which would be a more comprehensive measure than a decrease in the maximum queue length since the latter considers only stopped vehicles. The present study adopted a decrease in the maximum number of WAVEs as the reward.

An artificial agent in charge of traffic signal control was assumed to change signal phases according to the reward (=reduction in the maximum number of WAVEs) and the traffic state (=images of the intersection). The agent got the reward value of +1 if the maximum number of WAVEs across every movement at the beginning of the current time interval was smaller than that at the beginning of the previous time interval, whereas it was given a reward value of -1 if the maximum number of WAVEs at the beginning of the current time interval was larger than that at the beginning of the previous time interval. The reward value was set at 0 if there was no change in the maximum number of WAVEs. Although measuring the number of WAVEs or the queue lengths is not trivial in the real world, almost every study of RL-based traffic signal control has overlooked this difficulty. Of course, the queue length can be measured if the existing image processing tools use the background subtraction method (Milla et al., 2013), but it is not easy to measure the number of WAVEs using only simple images without predefined rules. A deep CNN can be re-adopted to estimate the WAVE number. Of course, this involved a large amount of intersection photos labeled with the number of WAVEs. It should be noted that since this supervised learning is conducted offline it does not place a burden on the main RL learning procedure. The CNN model to count vehicles was totally independent of the other CNN model that parametrized the Q-function within a RL algorithm for traffic signal control.

Prior to preparing the present study, the author was able to count vehicles on an urban road based solely on simple photos (Chung and Sohn, 2017). In this preceding work, labels were manually attached to photos of a 145 m-long approach to a real intersection located in Seoul. Fig. 2 shows two representative photos of the intersection approach. 4,632 photos were chosen for a test, and the remaining photos were used to train a deep CNN model. As a result of the test, the XY-plot of observed and estimated values seems very promising [see Fig. 2 (c)]. The mean absolute error (MAE) was only 1.57 vehicles across the test dataset, although there were many occlusions. In particular, it was surprising that vehicles behind trees or traffic signs were counted properly.

(a) Congested case (Actual vehicles: 77, Estimated: 77.4)

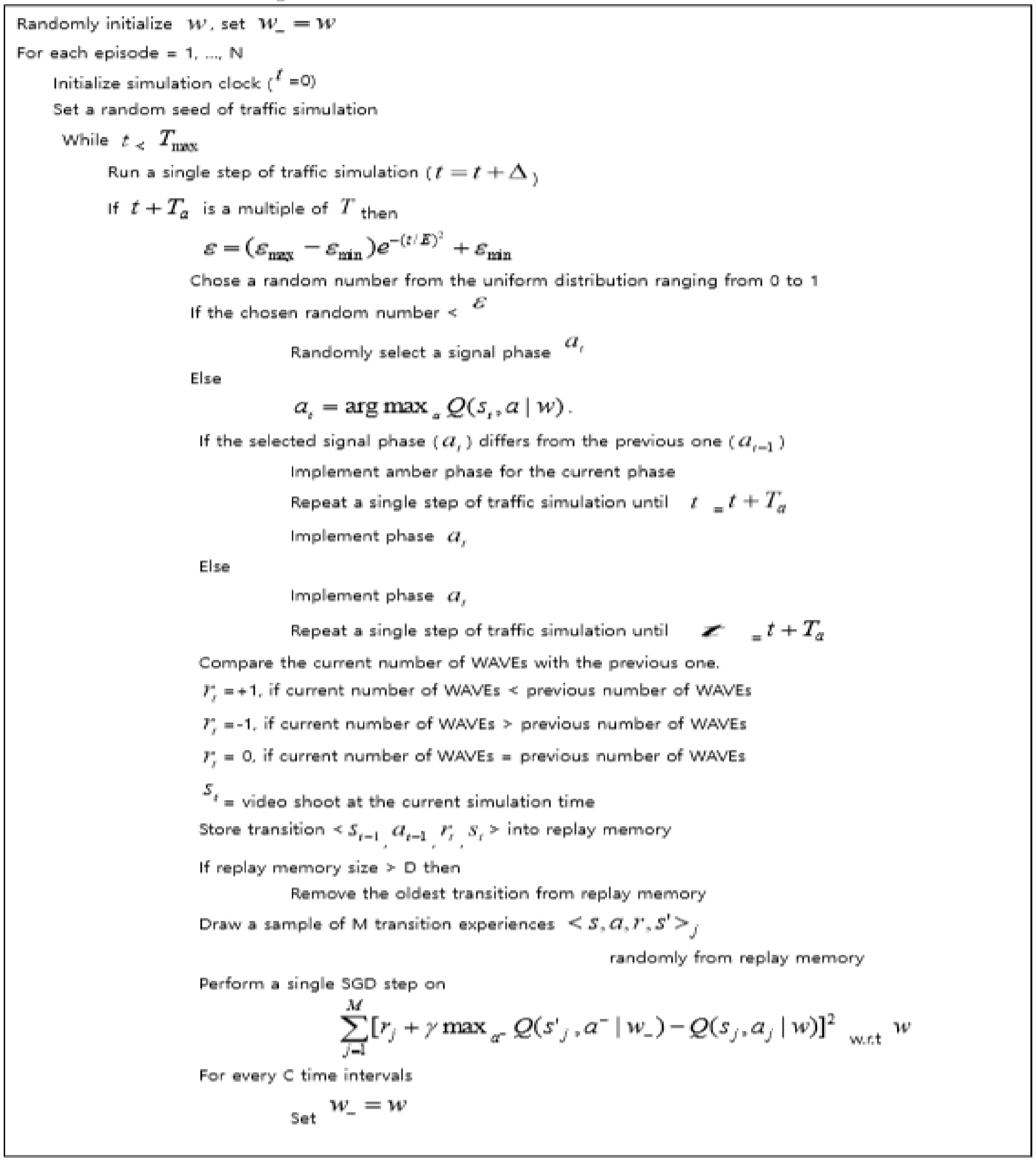

(b) Uncongested case (Actual vehicles: 11, Estimated: 11.2)



(c) Test result (MAE: 1.57)

Figure 2. Counting vehicles in urban arterial (Chung and Sohn, 2017)

It is apparent that the proposed vehicle counting model can be successfully incorporated with RL-based traffic control models in the real world. However, for brevity the WAVE number was directly extracted from a traffic simulator to compute the immediate reward in this study. A RLbased learning algorithm for traffic control takes a long time to reach a stable condition. While training the model in the real world, drivers might experience a very long queue due to random exploration. Apparently, they do not expect this type of imposition. It should be noted that this unexpected delay occurs not only for the present model, but also for every traffic



```
Randomly initialize $w$, set $w_- = w$
For each episode = 1, ..., N
    Initialize simulation clock ($t$ =0)
    Set a random seed of traffic simulation
    While $t < T_{max}$
        Run a single step of traffic simulation ($t = t + \Delta$)
        If $t + T_a$ is a multiple of $T$ then
            $\varepsilon = (\varepsilon_{max} - \varepsilon_{min}) e^{-(t/E)^2} + \varepsilon_{min}$
            Chose a random number from the uniform distribution ranging from 0 to 1
            If the chosen random number < $\varepsilon$
                Randomly select a signal phase $a_t$
            Else
                $a_t = \arg\max_a Q(s_t, a \mid w)$.
            If the selected signal phase ($a_t$) differs from the previous one ($a_{t-1}$)
                Implement amber phase for the current phase
                Repeat a single step of traffic simulation until $t = t + T_a$
                Implement phase $a_t$
            Else
                Implement phase $a_t$
                Repeat a single step of traffic simulation until $t = t + T_a$
            Compare the current number of WAVEs with the previous one.
            $r_t$ =+1, if current number of WAVEs < previous number of WAVEs
            $r_t$ =-1, if current number of WAVEs > previous number of WAVEs
            $r_t$ = 0, if current number of WAVEs = previous number of WAVEs
            $s_t$ = video shoot at the current simulation time
            Store transition < $s_{t-1}$, $a_{t-1}$, $r_t$, $s_t$ > into replay memory
            If replay memory size > D then
                Remove the oldest transition from replay memory
            Draw a sample of M transition experiences $<s, a, r, s'>_j$
                                        randomly from replay memory
            Perform a single SGD step on
                $\sum_{j=1}^{M} [r_j + \gamma \max_{a^-} Q(s'_j, a^- \mid w_-) - Q(s_j, a_j \mid w)]^2$ w.r.t $w$
            For every C time intervals
                Set $w_- = w$
```

signal control model that depends on a RL-based learning algorithm. A plausible way to circumvent this problem is to fine-tune a model using a reliable traffic simulator before installing it in a real intersection.

Fig8 shows the entire procedure of a RL-based traffic signal control based on simulation. Details of the simulation environment and the hyper-parameters will be described in the next section. When selecting the next signal phase (=action), exploration is inevitable to allow a model to self-learn through experience. A phase was chosen randomly with a predefined probability (= e). Otherwise, the phase that had the maximum Q-function value was selected with the probability of 1e . This is called the e -greedy choice of an action. For the present study, the exploration probability varied as time passed. At the initial stage, the probability was set as a value close to 1 so that a sufficient number of trial-and-error trials could be explored, while the probability converged to 0 after a sufficient number of iterations [see Table 1 (a)].

Table 1. Model specifications

(a) Hyper-parameters of the RL model

Hyper-parameter	Description	Applied value
N	Number of episodes	50
	Simulation time step	0.2 second
Tmax	Simulation period for each episode	20,000 seconds
T	Time interval	20 seconds

T a	Amber time	3 seconds
max	Initial probability of exploration	1.0
min	Final probability of exploration	0.1
E	Decaying parameter of exploration probability	400,000
D	Size of replay memory	2,000
M	Size of mini-batch sample	32
C	Cycle for updating target Q-function	20,000 seconds

(b) Hyper-parameters of deep CNN

Layer	Hyper-parameters	Applied value
Input layer	Resolution of input image	168 168 (Black-and-white)

	Number of input images for each time interval	1
1st hidden layer	Filter size	8 8
	Number of filters	16
	Stride size	(4, 4)
	Activation function	ReLu
2nd hidden layer	Filter size	4 4
	Number of filters	32
	Stride size	(4, 4)
	Activation function	ReLu

r hidden 3d layer	Filter size	3 3
	Number of filters	6 4
	Stride size	(1, 1)
	Activation function	ReL u
4t hidden h layer	Number of nodes	51 2
	Activation function	ReL u
Output layer	Number of nodes	4
	Activation function	Linear

A deep CNN was adopted as an approximation function for the original Q-function of the proposed RL-based traffic control model. Each layer of the CNN was rearranged with overlapping tiles to abstract localized characteristics of an image (LeCun et al., 2015), which played a crucial role in converting images of traffic states into an action-value. The network structure and application skill was dealt with in this paper, as described in the next section.

Simulation design

The proposed architecture for simulation experiments is shown in Fig. 3. The main logic for an RLbased traffic signal control was coded by Python 2.7 with the library Numpy for handling arrays efficiently. Theano, another library that is used to facilitate computations with large-scale matrices on a graphical processing unit (GPU), was running in the background. Keras, a deep-learning library that depends on either Theano or Tensor-Flow, was adopted to train a CNN to approximate the Q-function based on animation images. Instead of a real traffic operation, the present study utilized Vissim, a commercial traffic simulator, as an environment. The library WinCom was employed for the main program written in Python to interface with a Vissim object model at the code level. Except for Vissim, the rest of the components were all open-source libraries. This architecture was invented during the course of the present study to conduct RL-based traffic control simulations.

A real intersection located in Seoul, Korea was chosen as a test-bed for the simulation. Fig. 4 shows both a real photo and an animation image of the test-bed.

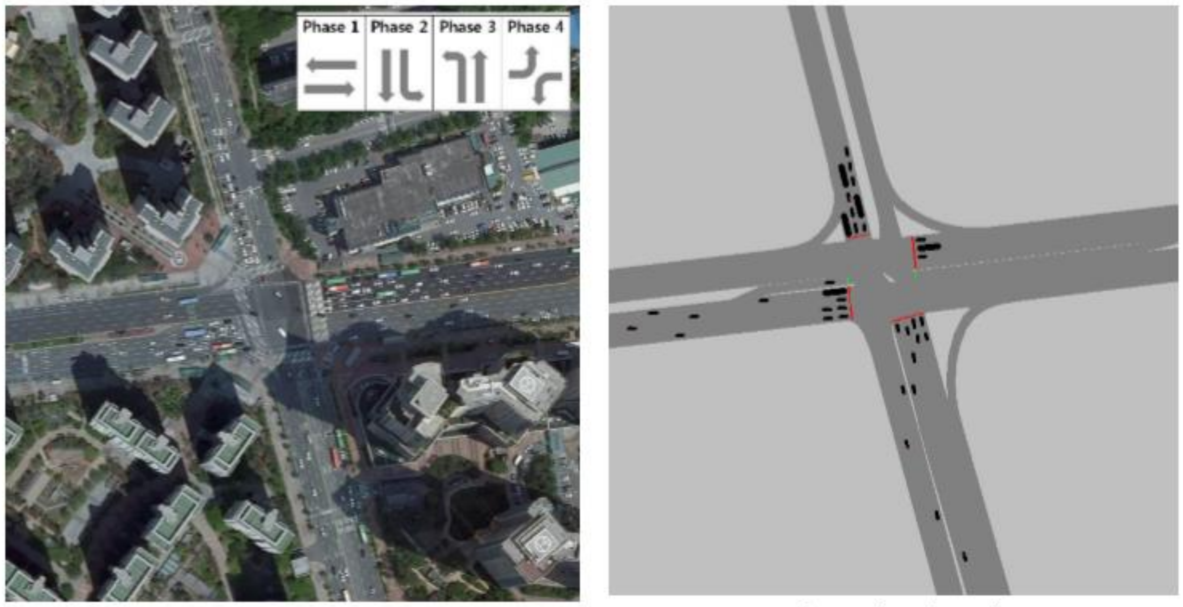


Figure 4. Test-bed images

Complex backgrounds should be eliminated from real photos prior to being inputted for the CNN model. Simple imageprocessing skills can accommodate this process when adopting real photos in the future. Unfortunately, there is no traffic simulation tool that provides an animation video in virtual reality. Most commercial simulation tools provide simple animation, such as that seen in classic video games. In the present study, a single image at the end of each time interval was used as a traffic state for the interval. The actual 4-phase signal was applied to the simulation, but the order of the phases was varied by the proposed algorithm. Of course, an amber time was offered when signal phases were switched. If the same action is chosen consecutively, the previous signal phase lasts until the next action is taken without the amber time. In order to construct input traffic volumes, observed peak-hour traffic volumes were altered by up to ± 30% to consider both non-peak traffic conditions and non-recurrent perturbations. Input traffic volumes were changed every 2,000 seconds during an episode. Under the simulation conditions, the proposed RL-based control algorithm changed the order and the duration of the signal phases. Table 1 shows the hyper-parameters adopted in the simulation experiment for both Qlearning and the CNN. The algorithm was trained for 20,000 simulation seconds (= about 5 hours 30 minutes) for each episode. There were 50 episodes simulated for training the model. Thus, the total simulation time was tantamount to 1,000,000 seconds. During the first 900,000 seconds, the exploration rate was gradually reduced from 100 to 10%. The 10% rate was maintained during the remaining 100,000 seconds. Performance measures for the intersection operation were collected during the last 100,000 seconds. The phase was selected every 20 seconds. The amber time was applied whenever a signal phase changed, which was consistent with the actual operation (=3 seconds) on the testbed. The mini-batch size was set at 32, which equated to the number of transition examples from which a gradient was computed at each time interval. Weight parameters of the main Q-function were updated at the beginning of each episode. The random seed of traffic simulation was also altered at the beginning of each episode.

The principle that a deep CNN approximates the main Q-function cannot be intuitively described, but can be partially accounted for by introducing how it extracts features from images and links them into the main Q-function value that represents the mean cumulative reward in the future for each action. First, it was necessary to set up the specifications of the deep CNN adopted in the present RL model. The input layer corresponded to a 168*168 input image. The output layer had 4 nodes, each of which was required to correspond to each phase (=action). Besides an input layer and an output layer, three convolutional hidden layers were included, as listed in Table 1 (b). The first hidden layer had 16 8*8 filters. Each filter of the first hidden layer extracted a specific feature while sliding through an input image with a stride value of 4 in both horizontal and vertical directions. Basically, it is not trivial to recognize which feature can be extracted by a specific filter. However, the following example is plausible. The first filter would derive a rectangular feature that represents a vehicle, regardless of where the vehicle resides in the image, and the second filter would elicit a half-circle feature that represents a vehicle’s front and end parts, and so on. The second hidden layer has 32 4*4*16 filters. Each filter of the second hidden layer could extract more complex features form 16 different feature matrices that filters of the first hidden layer have created. Continuously, filters of the second hidden layer slide through feature matrices of the first hidden layer with the predefined stride value. The third hidden layer has 64 3*3*32 filters, each of which also extract a higher-level feature from the second hidden layer. The stride value that filters of the third layer use is 1.

The last hidden layer with 512 nodes was fully connected from the last convolutional hidden layer, which was then linked to the output layer with 4 nodes. The nodes within each convolutional hidden layer were activated by a rectified linear unit

(ReLu), while the last output nodes were activated linearly. The activation based on ReLu was a great breakthrough for deep learning (Nair and Hinton, 2010). Fig. 5 depicts the structure of the proposed deep CNN. An "RMSprop" algorithm was adopted to train the CNN with a learning rate of 0.00025. The details of this algorithm are outlined by Dauphin et.

A CNN plays a key role in representing a traffic state as an input for the main Q-function, and its hyper-parameters affect the overall performance of the overall RL learning. The hyperparameters listed in Table 1 (b) were chosen on a trial-and-error basis. The final hyper-parameters were determined after testing as many plausible combinations of hyper-parameters as possible. The initial reference of the hyper-

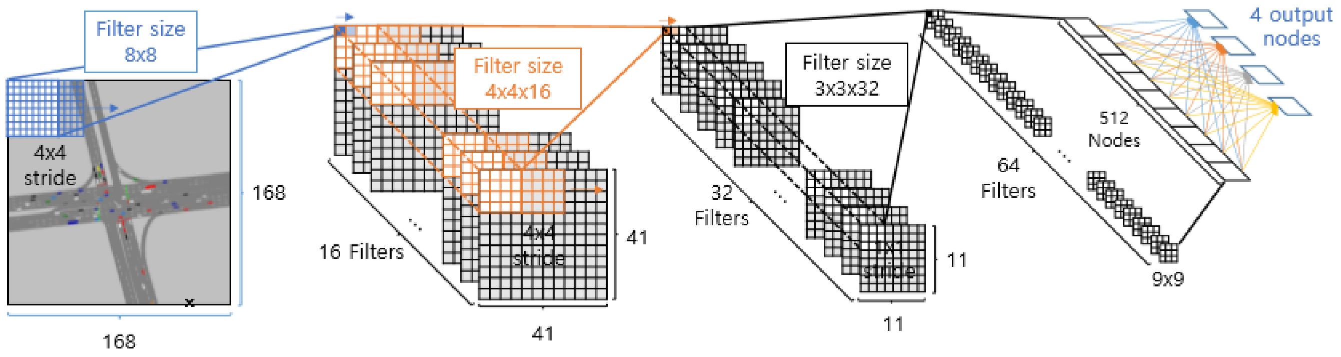


Figure 5. The structure of deep CNN for Q-function approximation

parameters was selected from those adopted by Minh et al. (2015).

The computing environment of the present simulation was as follows. The computer main memory was 128 GB. Python main logic and traffic simulation were implemented on two CPUs with the following specifications: Intel Xeon(R) CPU E5-2697 v2 @ 2.7GHz. There were 48 available CPU cores, which exceeded the maximum number of cores (=32) that Vissim allows. The deep CNN was trained on a single GPU. The GPU was a NVIDIA Quadro M6000 with a 12 GB GDDR5 memory. Ironically, a graphic accelerator was used to train the deep net, whereas the animation was run on CPU cores. It should be noted that, without a GPU, training a large-scale deep net cannot be conducted within a realistic window of computing time. Under this computing environment, it took, on average, 48.36 minutes to run a single episode. The total computation time for 50 episodes was tantamount to 40 hours and 20 minutes.

Simulation results

Simulation results were collected during the last 100,000 seconds with the probability of exploration fixed at 10%. As shown in Fig. 6, the average Q-function value across 4 phases converged to a certain value, although the trend was not smooth. The solid line in Fig. 6 indicates the fitting result from the 6th order polynomial function.

Three performance indices were recorded for each of the 10 time intervals (200 seconds): delay, vehicle throughput, and the maximum queue length across all lanes. In addition to the convergence of the main Q-function value, Fig. 7 provides more

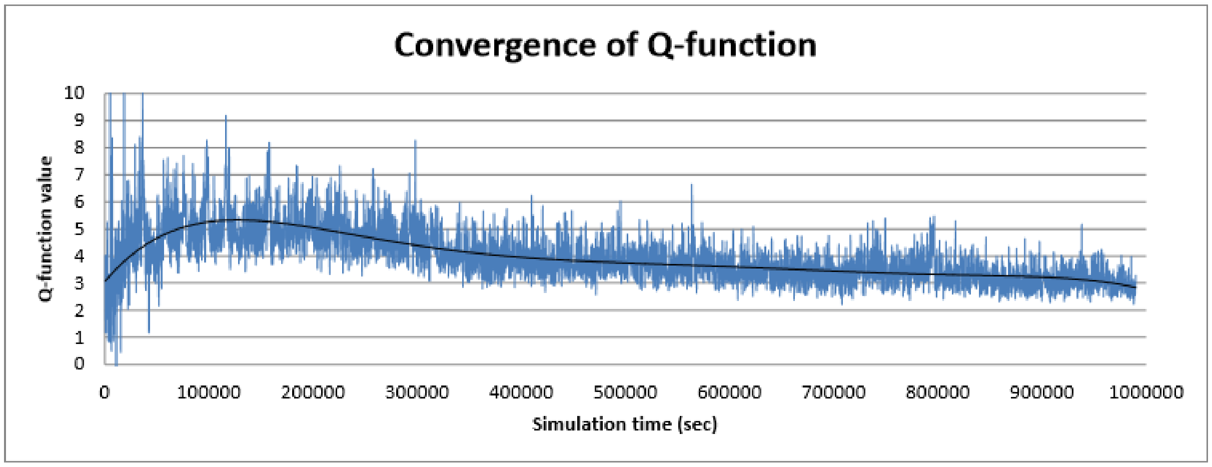


Figure 6. Trend of average Q-function values according to simulation time

intuitive evidence of the convergence with respect to the three chosen performance indices. Both the maximum number of WAVEs and the average delay converged as the simulation time passed. In the initial stage of learning, both indices fluctuated widely but became calm as the simulation time passed, which was the main reason that simulation experiments preceded the field test. Regarding the vehicle throughput, the variance was reduced considerably after learning, while the mean value did not change. The solid line in Fig. 7 indicates the moving average of 100 previous time intervals.

The actual operation of the test-bed was used as a reference for comparison. The test-bed was a 4-legged intersection operated by a fixed-signal plan. The cycle length was

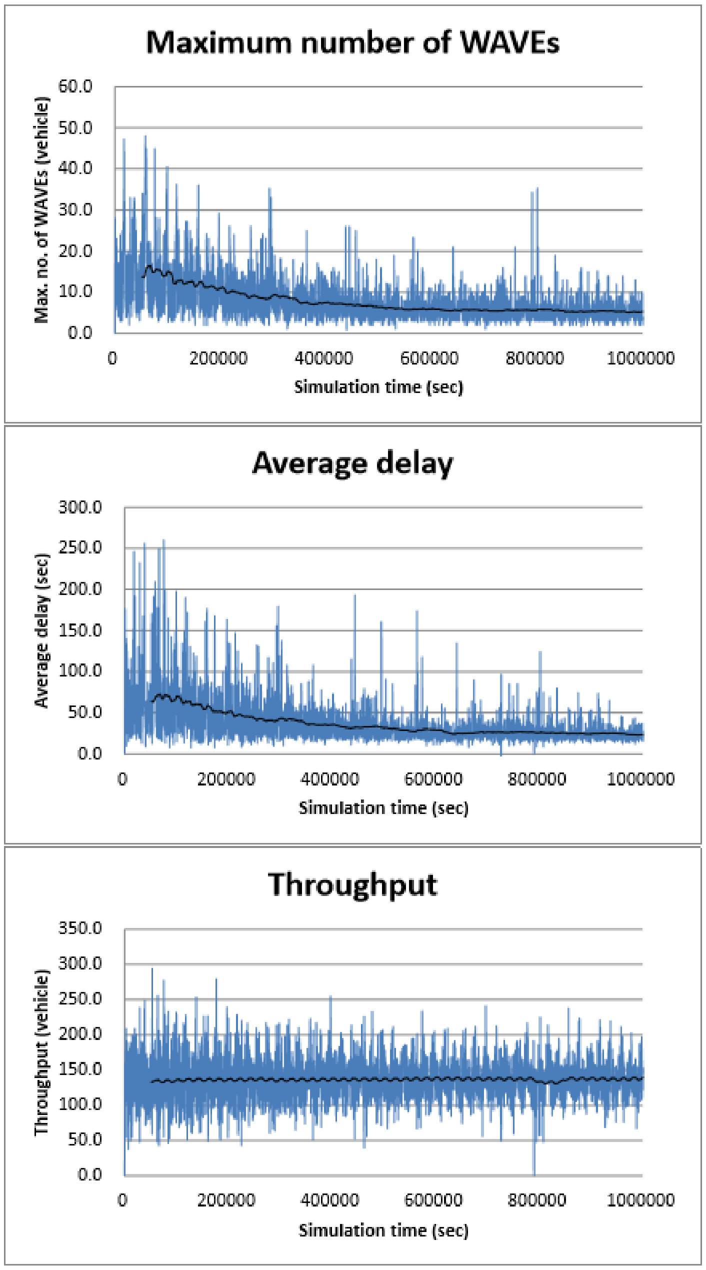


Figure 7. Convergence of the algorithm with respect to the chosen performance indices (the maximum number of WAVEs, the average delay, and the throughput)

140 seconds. Signal times were assigned to four different phases: 53(3), 23(3), 35(3), and 17(3) seconds. The figures within parenthesis indicate an amber time. These signal settings were determined based on observed traffic volumes according to the

capacity manual. A fully actuated signal operation scenario was also set up to be compared with the proposed RL-based algorithm. Vissim APIs fully supported establishing the scenario. The structure of the fully actuated signal scenario is summarized in Table 2. For both the fixed and fully actuated operations, the same traffic volumes were adopted as used in the RL-based simulation, which were altered by up to ±30% from the real peak traffic volumes.

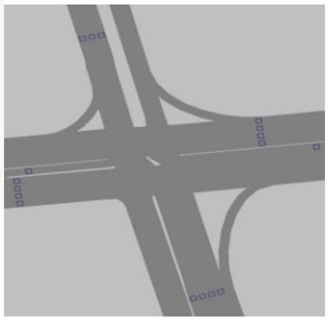
Table 2. Structure of the fully actuated signal
operation

(a) Applied parameters

Detector Length (m)	2
Loss time (sec)	4
Headway (sec)	2
Average vehicle length (m)	5.5

(b) Detector locations

	Direction	Distance from stop-line to detector
		45m

		50m
		30m
		40m
	↰↑ ↲	30m
	← ↓↳ ↱ →	35m

(c) Minimum and maximum green times

Phase	Minimum green (sec)	Max green (sec) (=Actual fixed signal)
(1)	18	53(3)
(2)	20	23(3)
(3)	22	35(3)

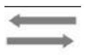
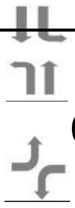
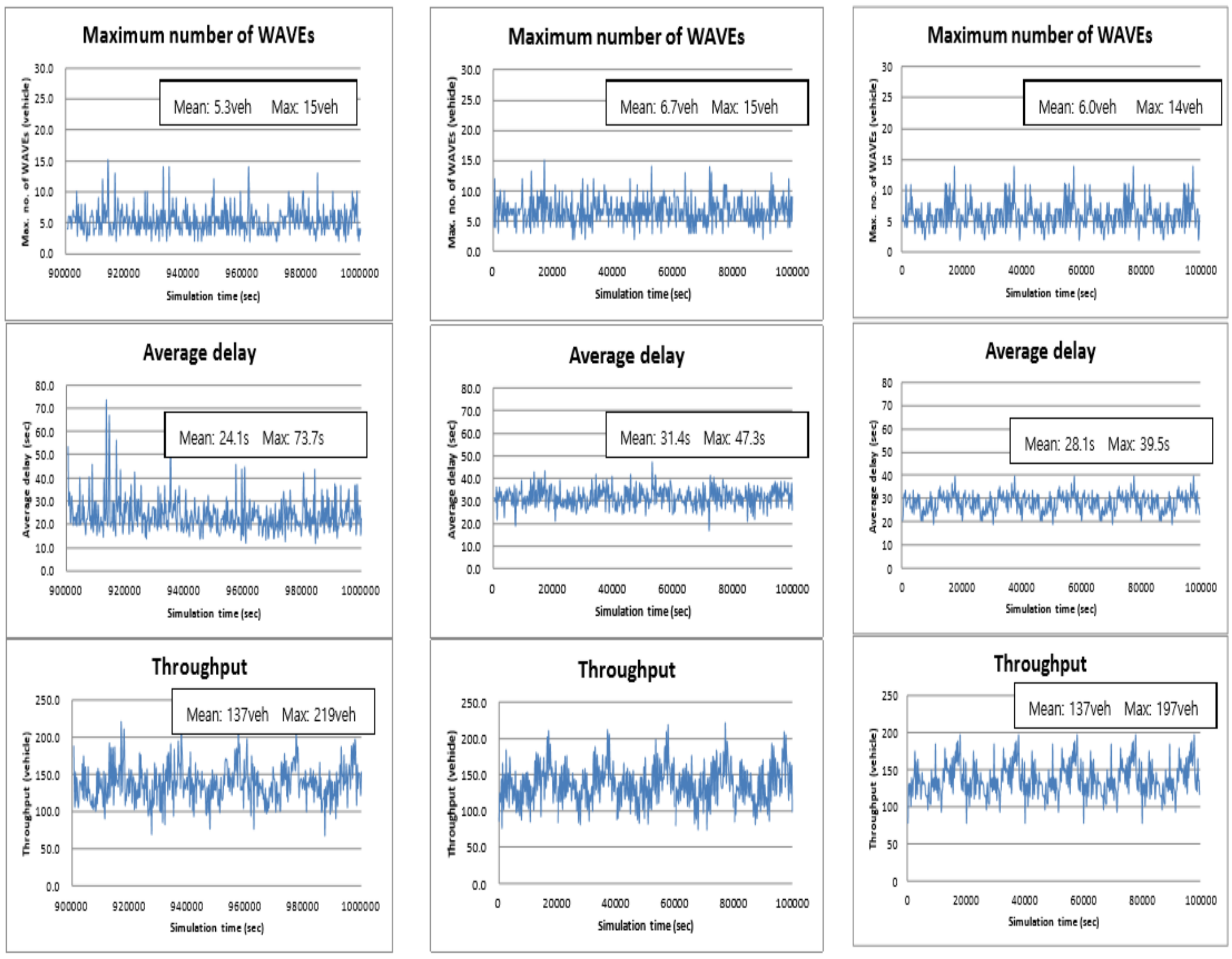
		
 (4)	24	17(3)

Fig. 8 shows the three performance indices for the three signal control schemes: the RL-based control, the actual fixed operation, and the fully actuated signal operation. The queue length was measured at the beginning of each time interval, while the remaining two indices were averaged across values during 10 time intervals. All



(a) RL-based learning algorithm (b) Actual operation (c) Fully-actuated signal control

Figure 8. Comparison of the RL-based learning algorithm with both the actual and fully-actuated operations w.r.t the three performance indices

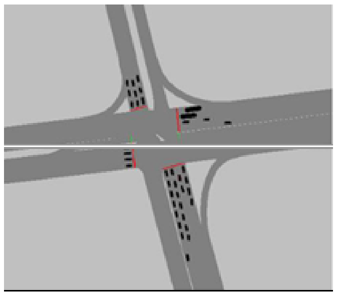
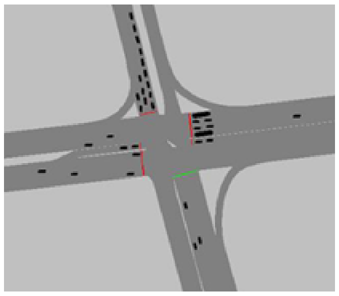
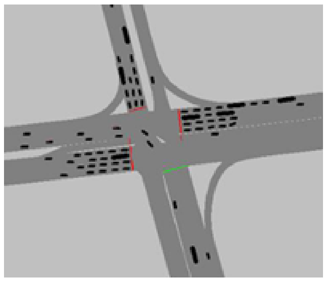
three mean values of the RL–based control during the last 100,000 seconds of simulation outperformed those from the actual operation and the fullyactuated operation. In particular, it was surprising that the average delay was reduced by more than 23% when compared with the actual operation. The mean of the maximum WAVE of the learning-based algorithm was also smaller than those of both the actual and fully actuated operations by about 21 and 12%, respectively. The levels of vehicle throughput during a particular time interval did not differ.

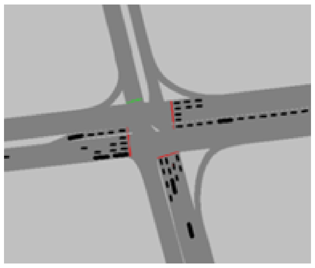
Contrary to the comparison of mean values, the maximum values in delay showed quite different results. The maximum delay of the RL-based control was 1.86-fold that of the maximum delay of the fully actuated operation and 1.55-fold that of the maximum delay of the actual operation. For the RL-based control, however, the frequency of encountering an extremely large delay was at most 3.4% when counting the number of occurrences where a delay from the RLbased control exceeded the maximum delay of fully-actuated operation (=39.5 sec). On the other hand, regarding the WAVEs, the maximum value (15 veh) from the RL-based algorithm was almost the same as those from both the actual and fully-actuated operations. The standard deviation (2.0 veh) in the maximum number of WAVEs for the RL-based learning algorithm was slightly smaller than that for the actual fixed operation (2.1 veh) and that for the fully actuated operation (2.2 veh).

The fluctuations in delay stemmed from the fact that the present study set the reward of the RL as reducing the maximum number of WAVEs instead of minimizing vehicle delay. The proposed algorithm depended upon no delay-related measure unlike many other previous RL studies that adopted the delay as a reward or a state. In reality, the delay is only an imaginary index because it could not be measured accurately in the field without an advanced communication-based surveillance system. As mentioned earlier, this kind of surveillance system cannot prevail in the foreseeable future.

Deep-learning models have been criticized by many researchers as resembling a black box. The present study largely depended on a deep CNN model, and will not escape this criticism. The following explanation is intended to be a plausible answer to such criticism. Although it is impossible to account for the intrinsic mechanism in which the proposed deep CNN model was operating, some evidence can be provided to show that the model recognized traffic states much as a human would. Table 3

Table 3. Traffic state vs. Q-function

Traffic state	Phase	Q-function
		3.00
		3.15
		4.24
		2.97
		2.98
		3.24
		2.92
		2.77
		**3.50**
		3.15
		2.99

		2.46
		2.69
		2.40
		2.74
		3.00

shows 4 typical cases of mapping between current traffic states and the corresponding Q-function values estimated by the proposed deep CNN model. According to the proposed model, the next phase was determined by the output node that possessed the maximum Q-function value. According to the figure, the next phase choice the Q-function made was consistent with human intuition.

Conclusions

This study is the first to adopt an image-based RL algorithm for traffic signal control. All other previous attempts to use a RL-based signal control algorithm depended upon the existing traffic parameters measured from detectors to represent a traffic state. Apparently, a RL model based solely on images outperformed the actual and fully actuated operations of an intersection in all three performance measures. The average delay was reduced by more than 23% when compared with the actual operation. The mean of the maximum WAVE of the learning-based algorithm was also smaller than those of both the actual and fully actuated operations by about 21 and 12%, respectively. The levels of vehicle throughput during a particular time interval did not differ.

Despite superiority, the average delay from the proposed model fluctuated more than those from the other two reference operations. Finding hyper-parameters that will guarantee stability in delays is inevitable with further study. After determining the hyper-parameters that will guarantee stability in delays, the model should be trained in the field by using real video over a test-bed intersection. Furthermore, the present study has a limitation whereby it could be applied only to a single independent intersection. New simulation experiments are currently under construction to jointly control closely spaced intersections. If a single agent were assumed, the present algorithm could be altered slightly to accommodate increases in state input and actions. A multi-agent RL algorithm would require multiple deep CNNs. The obstacle for both approaches would be the burden of computation time. Multiple GPUs are now being tuned in parallel to overcome this problem.

The ultimate goal is an image-based artificial intelligence that can exceed that of both humans and those of the existing advanced technologies. That goal is a distant one, but this study has taken the first step.

References

1. Abdulhai, B., Kattan, L.: (2003) Reinforcement learning: Introduction to theory and potential for transport applications. Canadian Journal of Civil Engineering 30(6), 981–991.

2. Abdoos, M., Mozayani, N., Bazzan, A.: (2011) Traffic light control in non-stationary environments based on multi agent q-learning. In: Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on, pp. 1580–1585

3. Abdoos, M., Mozayani, N., Bazzan, A.: (2014) Hierarchical control of traffic signals using q-learning with tile coding. Applied Intelligence 40(2), 201–213.

4. Abdulhai, B., Pringle, R., Karakoulas, G.: (2003) Reinforcement learning for true adaptive traffic signal control. Journal of Transportation Engineering 129(3), 278–285.

5. Arel, I., Liu, C., Urbanik, T., Kohls, A.: (2010) Reinforcement learning-based multi-agent system for network traffic signal control. Intelligent Transport Systems, IET 4(2), 128–135

6. Baird., L. (1995) Residual algorithms: Reinforcement learning with function approximation. ICML, pages 30–37

7. Bakker, B., Whiteson, S., Kester, L., Groen, F.: (2010) Traffic light control by multiagent reinforcement learning systems. In: R. Babuka, F. Groen (eds.) Interactive Collaborative Information Systems, Studies in Computational Intelligence, vol. 281, pp. 475–510. Springer Berlin Heidelberg

8. Bazzan, A.L.C., Klgl, F.: (2014) A review on agent-based technology for traffic and transportation. The Knowledge Engineering Review 29, 375–403.

9. Bazzan, A.L.C.: (2009) Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. Autonomous Agents and Multi-Agent Systems 18(3), 342–375.

10. Boureau, Y.L., Ponce, J. and LeCun, Y. (2010) A Theoretical Analysis of Feature Pooling in Visual Recognition. In International Conference on Machine Learning

11. Chung, J. and Sohn, K. (2017) Image-based learning to measure traffic density using a deep convolutional neural network (CNN), IEEE Transactions on Intelligent Transport

12. Ciresan, D., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J. (2011) Flexible, High Performance Convolutional Neural Networks for Image Classification, Proceedings of the TwentySecond international joint conference on Artificial Intelligence-Volume Volume 2: 1237–1242.

13. Corazza, M., & Sangalli, A. (2015). Q-Learning and SARSA: a comparison between two intelligent stochastic control approaches for financial trading.

University Ca'Foscari of Venice, Dept. of Economics Research Paper Series No, 15.

14. Dauphin, Y. N., de Vries, H., Chung, J., & Bengio, Y. (2015). RMSProp and equilibrated adaptive learning rates for non-convex optimization. arXiv preprint arXiv:1502.04390.

15. El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2014). Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. Journal of Intelligent Transportation Systems, 18(3), 227-245.

16. Isa, J., Kooij, J., Koppejan, R., Kuijer, L.: (2006) Reinforcement learning of traffic light controllers adapting to accidents. In: Design and Organisation of Autonomous Systems

17. Jin, J., & Ma, X. (2015). Adaptive group-based signal control by reinforcement learning. Transportation Research Procedia, 10, 207-216.

18. Khamis, M. A., & Gomaa, W. (2014). Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework. Engineering Applications of Artificial Intelligence, 29, 134-151.

19. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

20. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444. Liu, Z.: (2007) A survey of intelligence methods in urban traffic signal control. IJCSNS International Journal of Computer Science and Network Security 7(7)

21. Mannion, P., Duggan, J., & Howley, E. (2015). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. Autonomic Road Transport Support Systems.

22. Milla, J. M., Toral, S. L., Vargas, M., & Barrero, F. J. (2013). Dual-rate background subtraction approach for estimating traffic queue parameters in urban scenes. IET Intelligent Transport Systems, 7(1), 122-130.

23. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.,A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King H., D. Kumaran, D., Wierstra, D., Legg, S., and Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

24. Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10) (pp. 807-814).