Reinforcement Learning for Joint Control of Traffic Signals in a Transportation Network

Jincheol Lee, Jiyong Chung, and Keemin Sohn

***Abstract*—Reinforcement learning (RL) approaches have recently been spotlighted for use in adaptive traffic-signal control on an area-wide level. Most researchers have employed multi-agent reinforcement learning (MARL) algorithms wherein each agent shares a holistic traffic state and cooperates with other agents to reach a common goal. However, MARL algorithms cannot guarantee a global optimal solution unless the actions of all agents are fully coordinated. The present study employs a RL algorithm that recognizes an entire traffic state and jointly controls all the traffic signals of multiple intersections. With this approach, a deep Q-network (DQN) that depends solely on traffic images is extended to overcome the curse of dimensionality that is associated with a large state and action space. Several front layers in a deep convolutional neural network (CNN) to approximate the true Q-function are shared by each intersection approach. Weight parameters connecting the last hidden layer to the output layer are fixed. The proposed methodology outperforms a fixed-signal operation, a fully actuated signal operation, a multi-agent RL control without coordination, and a multi-agent RL control with partial coordination.**

***Index Terms*—Adaptive traffic signal control, Reinforcement learning, Deep Q-network.**

I. Introduction

REINFORCEMENT learning (RL) has recently been spotlighted for using in adaptive traffic control [1]–[6]. However, a problem arises when a RL algorithm is applied to jointly control signals at the network level. That is, the size of state-action space explodes, as the number of intersections to be jointly controlled increases, which is called the curse of dimensionality. A multi-agent RL algorithm (MARL) has been regarded as the solution to this problem. RL Agents for traffic light control are cooperative rather than competitive, which leads to a system-optimal traffic condition. A cooperative MARL problem usually has multiple optima, and thus cannot converge to an optimal solution without breaking ties among agents [7], even though each agent can deal with a global Q-function. The tie can be broken only when each agent knows all other agents' actions on a real-time basis. However, most MARL traffic signal controllers assume partial coordination wherein each agent communicates only with agents nearby [8]–[11].

The present study extended a deep Q-Network (DQN) that was developed by Mnih *et al*. [12] to jointly control traffic lights in a transportation network while avoiding the curse of dimensionality. We devised two simple remedies to circumvent the curse of dimensionality within the framework of the extended DQN. The first measure was to share a subset of weight parameters in a CNN to approximate the true Q-function. That is, the traffic state of each intersection approach in a transportation network was recognized through the common sub-architecture of CNN layers. All intersection approaches were processed with the same sub-architecture. Thus, the number of weight parameters could be a constant no matter how many intersection approaches were added. The sub-architecture allows different outcomes between intersection approaches, however, because each intersection approach receives different traffic states (= images). This scheme considerably reduced the number of weight parameters to be estimated. Otherwise, the number of weight parameters would increase proportionally to the number of intersections to be jointly controlled.

The other solution was a trick to deal with the large size of the discrete action space. According to the CNN structure proposed by Mnih *et al*. [12], the number of nodes in the output layer had to equal the dimension of action space. This scheme dramatically reduced computation time by simultaneously evaluating the Q-function values of all actions for a given state. Unfortunately, the scheme becomes infeasible as the size of the action space increases. The present study resolved the problem by assigning a value of 1 to every weight parameter connecting the last hidden layer into the output layer, which made the output layer redundant. The last hidden layer instead acted as a de facto output layer, which had a much smaller number of nodes. At most, the number of nodes in the last hidden layer was reduced to a product of the number of intersections to be jointly controlled and the number of feasible signal phases of an intersection. The subsequent simulation experiments showed that this simplification posed no harm while reaching an optimal policy.

Besides the use of these two measures to avert the curse of dimensionality, the present study made additional contributions to existing RL-based traffic signal controllers. First, representing a state by video images offered great flexibility. Neither preprocessing nor detecting traffic parameters was required to represent a traffic state. Second, the immediate reward based on a queue length or delay cannot be easily acquired in the field from the

Manuscript received June 13, 2019; revised August 19, 2019; accepted December 22, 2019. Date of publication December 27, 2019; date of current version February 12, 2020. This research was supported in part by the Chung-Ang University Research Scholarship Grants in 2019, and in part by the Korea Agency for Infrastructure Technology Advancement (KAIA) Grant funded by the Ministry of Land, Infrastructure and Transport Grant 19TLRP-B148659-02. The review of this article was coordinated by Dr. Z. Ma. *(Corresponding author: Keemin Sohn.)*

The authors are with the Laboratory of Big-Data Applications in Public Sectors, Department of Urban Engineering, Chung-Ang University, Seoul 06974, Korea (e-mail: traffic_control@naver.com; jiyong369@hanmail.net; kmsohn@cau.ac.kr).

Digital Object Identifier 10.1109/TVT.2019.2962514

conventional traffic surveillance schemes, even though it is the decisive factor to implement a RL algorithm. An image-based methodology was developed to derive the reward that can reflect traffic delay.

The next section will introduce literature regarding the contributions described in this introduction section. The third section will describe the modelling framework of a DQN. Details of the proposed traffic signal controller will be introduced in the fourth section. The fifth section will describe the design of simulation experiments that were implemented to confirm the performance of the proposed scheme. The results of simulation experiments and their comparisons with other methodologies are described in the sixth section. In the last section, conclusions are drawn, and further extensions are suggested to advance the proposed methodology.

II. Related Work

The MARL algorithm has been the universal choice to jointly control the traffic lights in multiple intersections. Some researchers have assumed that an independent agent controls each intersection without coordination, and that these agents are unaware of other agents' actions and states [13], [14]. This distributed-learning algorithm had the advantage of parallel computing, which dramatically reduced computing time. However, the MALR algorithm cannot guarantee a globally optimal solution without considering the coordination between agents [7]. Subsequent researchers extended the distributed MARL by including various types of partial coordination. Steingrover *et al*. [8] extended the MARL by providing additional information from neighboring agents. Kuyer *et al*. [9] also extended the work of Wiering by using a Max-Plus algorithm for coordination. The Max-Plus algorithm allowed partial coordination by passing messages between directly connected agents. Houli *et al*. [10] proposed a MARL algorithm that adopted a vehicular ad hoc network for information exchange among agents. El-Tantawy *et al*. [11] developed a partially coordinated MARL-based traffic signal controller that allowed each agent to communicate with its immediate neighbors and to partition the entire state space into subspaces consisting of a pair of agents. In fact, it is difficult to find a fully coordinated MARL model because of the curse of dimensionality [7], [15].

In addition to the curse of dimensionality, a more basic problem is how to represent the traffic state and the reward when implementing an RL-based traffic signal control. Most RL studies have taken it for granted that traffic parameters meant to represent the traffic state and the reward can be easily measured on-site. Conventional traffic parameters such as delays and queue lengths have been utilized to constitute the traffic state and reward [1], [11], [16]–[21]. However, the delays cannot be accurately measured on-site in real time under existing traffic surveillance systems, whereas the queue lengths could be measured by camera-based detectors. Some researchers have employed a cellular automata simulation to represent the traffic states and rewards of RL environments assuming that vehicular positions could easily be obtained [8], [9], [14], [22]. Genders and Razavi [23] and Liang *et al*. [24] chose a Boolean-valued matrix to indicate the vehicle presence on a road segment and used a CNN to map the matrix into an action-value. However, the location of an individual car cannot be accurately traced in the current stage of development. It will be a long time before all vehicles are equipped with on-board units to transmit all mobile information to a traffic controller.

Only a few recent studies [25], [26] have explicitly recognized this limitation and suggested methods to overcome it. Some researchers adopted traffic images to represent traffic states. Consecutive traffic images contain ample information about the traffic state. Humans recognize the traffic state of a road segment via their visual senses. A convolutional neural network (CNN) is expected to recognize a traffic state as humans do. Our previous work [25] directly adopted a raw image without the need of hand-crafted engineering to convert the image into a Boolean-valued matrix, and the rewards were also computed based solely on raw images. Casas [26] assumed that traffic states and rewards could be represented by raw data collected directly from loop detectors. No post-process was required to construct a higher-level traffic parameter from the raw data measured by detectors.

A DQN approach developed by Mnih *et al*. [12] had remarkable success in playing classic video games, whereby they provided a RL agent only with video images and let the agent play video games as humans do. A DQN is an critic-only methodology dealing with only a Q-function for solving a RL problem. On the other hand, several researchers adopted an actor-critic model wherein both an actor (= policy function) and a critic (= Q-function) are utilized to obtain an optimal policy [27]–[29]. Qi *et al*. [28] also utilized an asynchronous advantage actor critic (A3C) algorithm for vehicular edge computing, which were developed by Mnih *et al*. [30], wherein each actor-learner has a copy of a RL environment and implements actions in a separate thread. This could drastically reduce the computing time for convergence at the expense of large computer memory. However, it was not applicable for the proposed setting of traffic light control, since the required action space is too large to become the output of a policy function. A DQN was employed as a baseline model and was extended to solve the proposed joint traffic signal control problem.

A DQN approach approximates a true action-value function (= Q-function) with a CNN. The CNN is the only function that has been able to map consecutive traffic images into an action-value. Most RL-based traffic signal controllers have adopted a tabular update of the Q-function [1], [11], [19], [20], [21]. Image-based learning cannot be applied to this tabular methodology, since image-based states cannot be discretized. Prior to the DQN's emergence, some researchers have already employed the function approximation method [17], [18], [19], [31], [32]. However, such model-free RL methodologies utilized a naïve feed-forward neural network that cannot efficiently process image-based traffic states. Our previous study [25] confirmed that a DQN worked well for the traffic signal control of a single independent intersection. The present study extended the former approach to jointly control the traffic signals of multiple intersections in an urban transportation network.

III. Modeling Framework

An original DQN adopts video images as input for a CNN that can approximate a true Q-function. A deep CNN maps video images into action-values. In principle, when the state is video images and the action space is discrete, a parameterized CNN can replace the Q-function, as follows.

$$Q(s,a|W) = Q^{\pi^*}(s,a) \tag{1}$$

where, $Q^{\pi^*}(s,a)$ is the true state-action function when actions will be taken following the optimal policy π^*, $Q(s,a|W)$ is a CNN that approximates the true Q-function, s represents the initial state of the environment, a denotes an action that is taken for the given state, and W is a set of weight parameters of the CNN.

The loss function of a DQN is the mean square error between the target and the Q-function values. A parameterized Q-function (= a CNN) that minimizes the loss function will satisfy the Bellman optimality equation. Thus, minimizing (2) with respect to weight parameters (W) is identical to solving a RL problem.

$$\mathcal{L}(W) = E_{s,a,s',r}\left[\left(\underbrace{r+\gamma \max_{a'} Q(s',a'|W)}_{Target} - Q(s,a|W)\right)^2\right] \tag{2}$$

In principle, the expectation of the squared error in (2) should be taken across all possible combinations of the initial state and action and the resultant state and reward (s,a,s',r). However, it is impossible to obtain the data for all possible combinations (s,a,s',r) through experiments. A stochastic gradient descent (SGD) algorithm is the best fit for minimizing the loss function, wherein the loss function is iteratively minimized based on a small sample drawn from a replay set of (s,a,s',r) combinations. While training a DQN, a tuple of the current state and action and the resultant state and reward $[=(s_t,a_t,s_{t+1},r_{t+1})]$ observed in an iteration is not used directly to update the DQN, but is stored in a replay set.

If the replay set size exceeds the predefined threshold, the oldest tuple is discarded in a first-in-first-out fashion. In the current iteration, weight parameters of the Q-network (= a CNN) are updated based on a certain number of examples drawn randomly from the replay set. This approach avoids correlations between successive samples that can cause a solution to get stuck in a local optimum or can cause the loss function to diverge. As in the simple temporal difference (TD) update method, the DQN requires some amount of exploration in order to learn on a trial-and-error basis.

Directly minimizing (2) might show a large fluctuation in policy with slight changes to the parameters of the Q-function. To circumvent this problem, Mnih *et al*. [12] froze the target Q-function so that correlations between a learning Q-function and a target Q-function could be broken [see (3)]. Parameters of the target Q-function are fixed for the time being and are updated periodically on a long-term basis. When the target Q-function is updated, its parameters are set identical to those of an incumbent learning Q-function.

$$\mathcal{L}(W) = E_{s,a,s',r}\left[\left(r+\gamma \max_{a'} \underbrace{Q(s',a'|W^-)}_{Target\ Q-function} - Q(s,a|W)\right)^2\right] \tag{3}$$

where, W^- is a parameter set of the target Q-network.

In the DQN developed by Mnih *et al*. [12], a special structure was devised to accommodate a discrete action space. A CNN within the DQN did not use actions as input, and instead made them the output. The final layer had the same number of output nodes as the dimension of the action space. Thus, the CNN yielded multiple output values, each of which corresponded to a specific action. This treatment reduced the computation time to evaluate Q-function values, which also allowed for the sharing of weight parameters. The Q-function values for different actions were evaluated simultaneously from the same state input. However, even though actions are discrete, it is impossible to use this scheme if the action space is large. This problem is typical when jointly controlling traffic signals for multiple intersections. In the next section, an innovative way to tackle the problem will be suggested.

IV. Extending a DQN to Jointly Control Traffic Signals

A. The Definition of State

Consecutive video images were defined as a traffic state that reflects the dynamic and static characteristics of the traffic to be controlled. The Vissim, a commercial traffic simulator, provides high-quality animation for traffic states. Animation was used to create images to represent traffic states in the present study under the assumption that it would be a good replacement for video shoots depicting real traffic conditions on-site. This distinguishes the proposed traffic signal controller from other RL-based traffic signal controllers wherein a limited number of traffic parameters were adopted to represent traffic states. It is apparent that consecutive images contain much more information on the state of traffic than what the conventional traffic parameters offer.

B. The Definition of Action

There are two typical types of actions in RL-based traffic signal control. First, some researchers choose signal phases as discrete actions whereby the indication order of the signal phases could vary with the duration of each phase fixed [11], [14], [19], [25]. Second, either the green time duration or its ratio out of the cycle length can be defined as continuous actions with the indication order of signal phases fixed [16], [26], [33], [34]. Among the latter studies, the work of Casas [26] used a deep deterministic policy gradient (DDPG) method to deal directly with continuous actions, whereas the remaining studies discretized continuous actions to be adapted for a tabular Q-learning algorithm.

The present study employed the former discrete actions. Consequently, the duration of a signal phase could vary only by a multiple of the time interval of RL. There are two obstacles regarding the adoption of continuous actions. First, a DDPG method cannot handle a large number of actions (= joint phases), each of which has a continuous duration. Second, the additional discretization of signal phase durations exacerbates the curse of dimensionality. Recently, Dulac-Arnold *et al*. [15] developed a DDPG algorithm that can deal with a large discrete action space by approximating the procedure to choose an optimal action, but there has been no empirical evidence that it can solve real-world problems.

C. The Definition of Reward

One of the key requirements for a RL-based signal controller is to ensure that immediate rewards are easily recognized. Immediate rewards should be measured using existing surveillance systems such as a loop detector and a camera detector. In this context, delay-based indices cannot be used as an immediate reward in a real-world RL problem. The present traffic signal controller adopted a reward that could be computed from the traffic density and speed. Chung and Sohn [35] confirmed that a CNN can accurately count vehicles on a road segment or lane, based only on a video shoot. Furthermore, Joen *et al*. [25] successfully derived an immediate reward for their DQN from vehicle counts that were measured using the method developed by Chung and Sohn [35]. The reward was set at 1 if the current maximum number of waiting and approaching vehicles (WAVEs) in all lane groups was smaller than that measured at the previous time interval — otherwise, it was set at -1.

In the present RL-based traffic signal controller for multiple intersections, a global reward was computed in a manner equivalent to that described above based on the sum of each intersection's maximum number of WAVEs. The maximum number was chosen across all different lane groups of an intersection. An RL method based on this reward was simple to implement but had the drawback of not being able to consider vehicle delays. If a lane group had a small traffic volume, the signal phase for the lane group could not be initiated until the corresponding queue became sufficiently longer than queues for other lane groups with a heavy traffic volume.

To balance the signal phases between different lane groups, another reward specification was devised. For each lane group, the traffic density measured periodically in a time interval was inversely weighted by space mean speeds. Lee *et al*., [36] proved that a CNN can measure the space mean speed using two consecutive photos of a road segment. During the time interval in which an action was taken, vehicle counts and space mean speeds were measured several times (N_f), and the space mean speeds were used as weights to average the vehicle counts [see (4)]. The weighted vehicle count was then used as an impedance index to determine the immediate reward for the time interval. In terms of (4), a lane group with stopped traffic has an impedance index that is larger than that with moving traffic, even though both lane groups have the same number of vehicle counts. Although the impedance index could not replace an exact measure for delay, the index is a robust surrogate that approximates it. The impedance-based reward was set at 1 if the sum of the maximum impedance of each intersection across all lane groups was smaller than that measured at the previous time interval — otherwise, it was set at -1.

$$I_t(v) = \sum_{i=1}^{N_f} \left[C_{it}(v) \times e^{-\beta V_{it}(v)} \right] / N_f \tag{4}$$

where, v represents a lane group $I_t(v)$ is the impedance index at time interval t, N_f is the frequency of measuring both vehicle counts and space mean speeds within a time interval, $C_{it}(v)$ is the i^{th} measure of vehicle count during the time interval t, $V_{it}(v)$ is the i^{th} measure of space mean speed during the time interval t, and β is the sensitivity parameter of speed and should be a positive value.

To obtain the impedance-based reward, two external CNNs are necessary separately from the CNN that approximated the action-value function within a DQN. The two external CNNs should be trained off-line before implementing the DQN algorithm for traffic signal control. Training and testing of the two CNNs was very successful [35], [36], but in the present study vehicle counts and space mean speeds to compute the reward were obtained directly from a traffic simulator, which saved a considerable amount of computer memory. If extra graphical processing units (GPUs) are available, the evaluation of the two external CNNs is possible in parallel with the implementation of a DQN.

D. The Solution for Large State and Action Spaces

A CNN within a DQN abstracts hidden features from video images and maps the features into action-values. Although consecutive video images can replace large dimensional state variables, it is impractical to use a large image to cover all intersections to be controlled. An exorbitant amount of time is required for a CNN filter to convolute the large image. When dealing with a single intersection, Jeon *et al*. [25] used a square image that included the entire intersection, which led to a CNN filter sliding through empty spaces with no intersection approach. To remove this inefficiency, only images covering an intersection approach were cropped and used as input for a CNN.

The cropped images were placed on a standardized rectangular image, the resolution of which matched the maximum size of the intersection approach in the testbed. Consequently, the space that remained from a smaller image was zero padded. Hence, no matter how wide the testbed was, the computation time to convolute the input images was linearly proportional to the number of approaches. In addition to this simple scheme, two more active solutions to avoid the dimensionality problem will be described in what follows.

First, several front layers of a CNN to extract hidden features of input images were shared for all intersection approaches. This considerably reduced the number of weight parameters of a CNN. It should be noted that this scheme did not result in the same outcome for every intersection approach. The feature map created by the shared sub-architecture varied for different intersection approaches, since the input traffic states differed. Owing

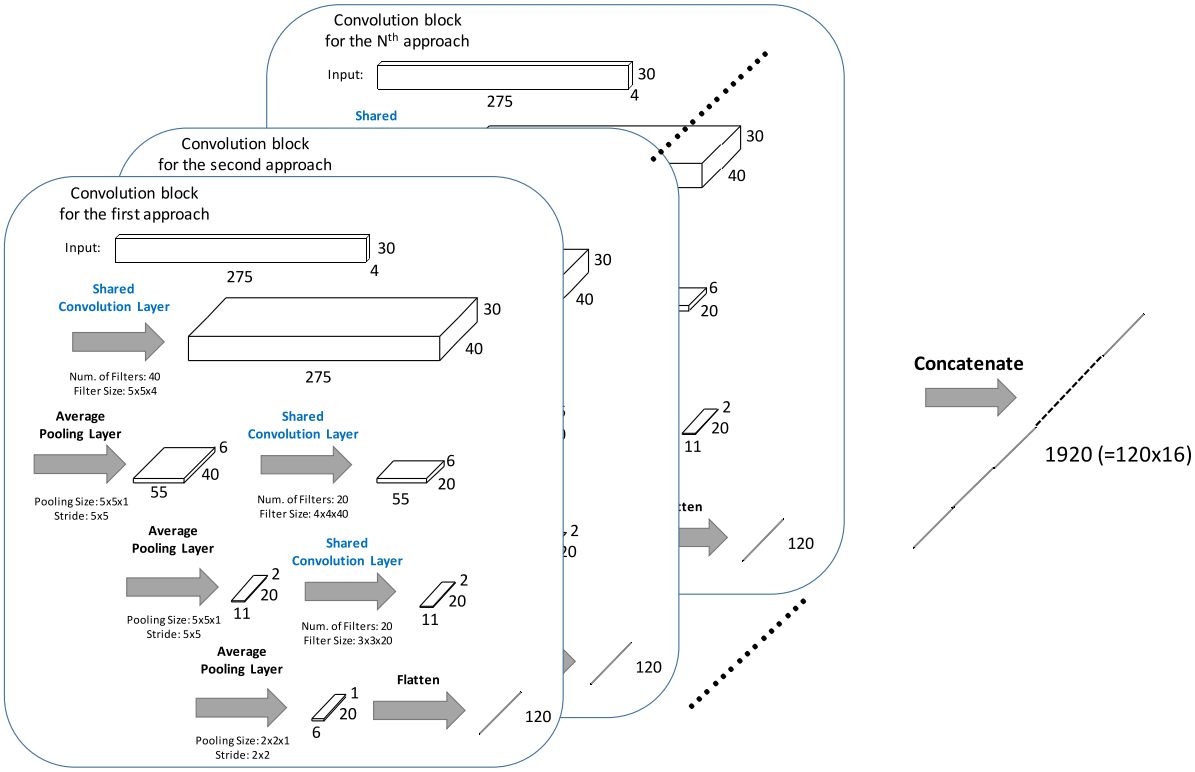

Fig. 1. Structure for processing input images based on the shared convolution blocks [the number of approaches (N) is 16 for a 2×2 transportation network with 4 intersections].

to the scheme, when abstracting input images, the number of weight parameters to be estimated did not increase, no matter how many intersections were included.

Fig. 1 shows the shared blocks that were applied to each intersection approach. How the shared blocks extracted hidden features from an intersection approach can be described as follows. The traffic state of an intersection approach was replaced with 4 consecutive rectangular gray-scale images, and their resolution was set at 30×275. These 4 input images were processed by 40 $5 \times 5 \times 4$ filters, and were then converted into 40 30×275 feature images in the first convolution layer. To compute the cell values of the feature maps, a linear combination of a filter's weight parameters and the cell values of the previous image covered by the filter was computed first, and then the computed value was activated by a rectified linear unit (ReLu). In Fig. 1, for brevity, the ReLu layer was not indicated explicitly but was regarded as included in the convolution layer. The size of new feature images after convolution was kept the same as the previous image via the use of zero padding. That is, prior to convoluting filters, the previous feature images were padded with null columns and rows to avoid a layer-by-layer dimension reduction.

After the convolution, a new feature image was created by pooling each of the 5×5 cells of the previously convoluted layer with their average values, which smoothed the feature images. After average pooling with a 5×5 sliding window, the dimension of the feature image dwindled to $55 \times 6 \times 40$. The second convolution layer was created by convoluting 20 $4 \times 4 \times 40$ filters on the pooled layer. Each filter created a feature image of the pooled layer that had the same size as that of the second convolution layer $(= 55 \times 6)$, by again using zero padding. Average pooling for the second convolution layer yielded the feature map with dimensions of $11 \times 2 \times 20$. The third convolution layer was created by convoluting 20 $3 \times 3 \times 20$ filters on the previous pooled layer. After activating and pooling, the last feature map with dimensions of $6 \times 1 \times 20$ was created. The last feature map was flattened and concatenated for linkage with the next fully connected layer.

The procedures thus far were commonly applied to every intersection approach in the testbed network. Sharing the weight parameters made the proposed traffic signal control algorithm advantageous in saving computer memory, so that an additional intersection would not increase the weight parameters of a CNN to approximate the Q-function. The weight sharing also provided a great flexibility to distribute the computing into multiple machines. The distributed scheme is expected to considerably save computation time, and to be resistant against failure.

Second, the remedy for overcoming the large dimension of action space is as follows. When an agent makes a joint decision as to which combination of signal phases will be assigned to all intersections for every time interval, the number of possible phase combinations explodes as the number of intersections grows. For example, when 10 intersections, each of which has 4 distinct signal feasible phases, are jointly controlled, the size of the action space (= the number of phase combinations for the entire network) would be tantamount to more than 1 million

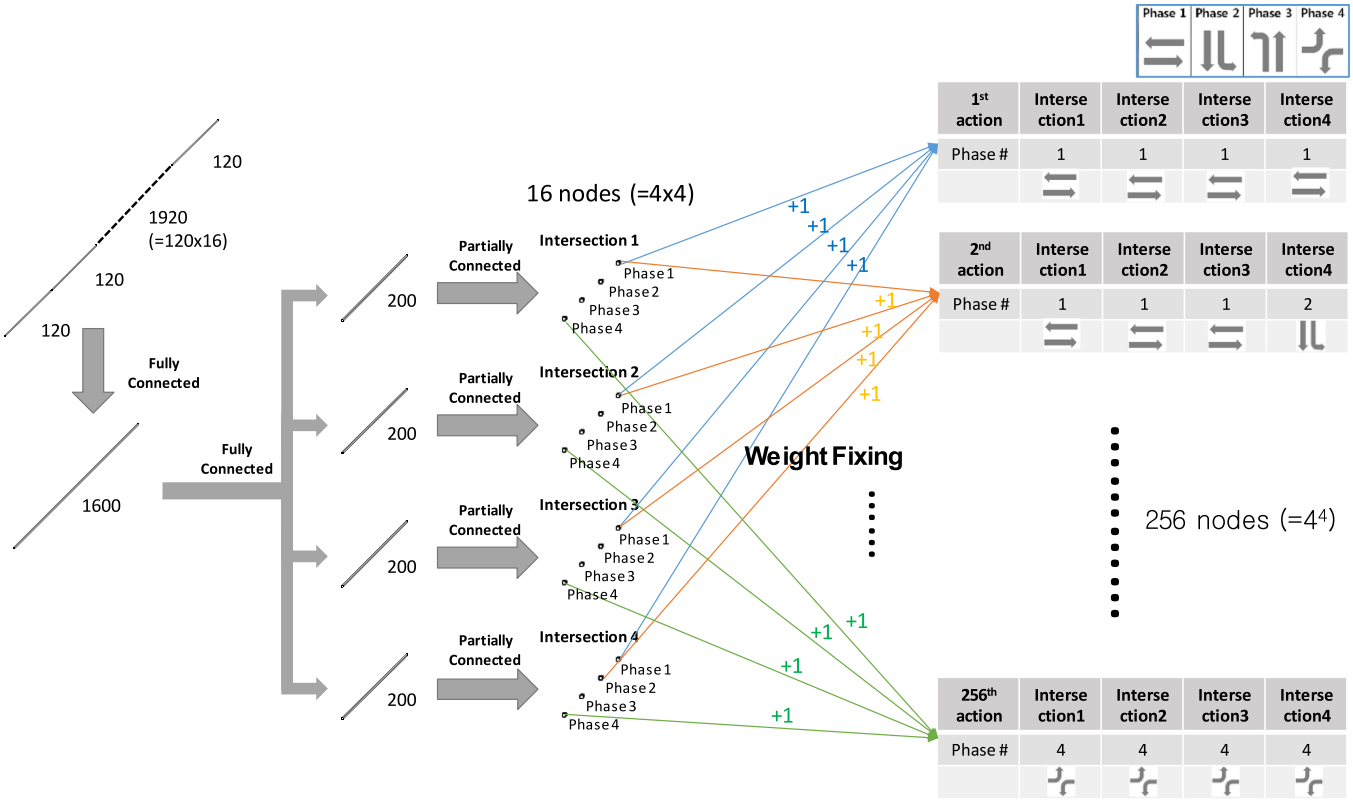

Fig. 2. Fixing weight parameters to reduce the action space (assuming a 2×2 transportation network with 4 intersections).

($\approx 4^{10}$). If the original DQN devised by Mnih *et al.* [12] was applied to this example, the number of output nodes would be more than 1 million.

To overcome the large dimension of action space, the weight parameters that connect the last hidden layer to the output layer were fixed at a constant (e.g., $+1$). Fig. 2 shows the remaining part of the CNN that approximated the true Q-function. The first layer in Fig. 2 indicates the concatenated fully connected (FC) layer in Fig. 1. That is, Figs. 1 and 2 together constitute the entire CNN. As shown in Fig. 2, two additional FC layers were attached to deepen the architecture. Then, 800 nodes in the last fully connected layer were divided into 4 groups such that each group included 200 nodes, respectively. Each group with 200 nodes in the last fully connected layer corresponded to a distinct intersection. Each node group was then connected to 4 nodes in the next partially connected layer, each of which represented a specific signal phase of the corresponding intersection.

In the present study, we designed a specific way to selectively connect the last hidden layer and the output layer. Each node of the output layer represented a unique combination of feasible signal phases for all intersections. An output node was fed by 4 different nodes in the last hidden layer, each of which came from a distinct intersection. This partial connection scheme with fixed weight parameters facilitated the training of the CNN, but never undermined the role of the proposed CNN to approximate the true Q-function. Basically, nobody knows the true form of a Q-function with respect to a continuous-state space. The Q-function has only been approximated by empirically proven artificial neural networks. The architecture for the artificial neural network used for approximation was chosen at the discretion of analysts. Only the performance of an arbitrarily chosen function could offer the rationale for the choice.

For the testbed where 4 intersections were included in a transportation network, each intersection had 4 distinct feasible signal phases, and the number of possible phase combinations of signals reached 256 ($= 4^4$). However, fixing the weight parameters at a constant that linked the last hidden layer to the output layer made the last hidden layer a de-facto output layer. The de-facto output layer had only $16 (= 4 \times 4)$ nodes. This validated the present approach as scalable, since the computation burden increased linearly proportional to the number of intersections to be jointly controlled.

The proposed CNN is fully qualified as an approximation function to yield a true action value. The CNN was able to accommodate the influence of the traffic states of other intersections when selecting a signal phase for a specific intersection. Furthermore, it could consider coordination between signal phases of different intersections when jointly choosing the best combination of signal phases. More concretely, correlations between joint actions could be recognized via hidden layers prior to the last FC hidden layer with 800 nodes within the CNN. If the weight parameters included in the layers change, many output node values are affected. Since the weight parameters are updated via many joint actions while training the CNN, correlations between the joint actions are stored by the weight parameters. Unfortunately, how the CNN recognizes correlations cannot be explicitly determined. This is a common handicap of deep learning models, as many researchers have noted.

TABLE I
RL-BASED TRAFFIC SIGNAL CONTROL ALGORITHM FOR A TRANSPORTATION NETWORK

Randomly initialize W
set $W^- = W$
For each episode (1, ..., N)
  Initialize simulation clock (t = 0)
  Set a random seed of traffic simulation
  Initialize iteration number (n = 0)
  Randomly select a combination of signal phase $\mathbf{a}_0 = \langle a_0^{I1}, a_0^{I2}, a_0^{I3}, a_0^{I4} \rangle$
  Set initial consecutive video shoots $\mathbf{s}_0$
  While t < T_{max}
    Run a single step of traffic simulation (t = t + Δ)
    If t + T_a is a multiple of T then
      n = n +1
      $\mathbf{s}_n$= consecutive video shoots at the current simulation time
      Compare the current sum of max. numbers of WAVEs with the previous one
      $r_n = +1$, if the current sum of max. numbers of WAVEs < the previous sum of max. numbers of WAVEs
      $r_n = -1$, Otherwise
      Store transition <$\mathbf{s}_{n-1}, \mathbf{a}_{n-1}, r_n, \mathbf{s}_n$> into replay memory
      $\epsilon = (\varepsilon_{max} - \varepsilon_{min})e^{-(t/E)^2} + \varepsilon_{min}$
      Choose a random number from the uniform distribution ranging from 0 to 1
      If the chosen random number < ϵ
        Randomly select a combination of signal phase $\mathbf{a}_n = \langle a_n^{I1}, a_n^{I2}, a_n^{I3}, a_n^{I4} \rangle$
      Else
        $\mathbf{a}_n = \langle a_n^{I1}, a_n^{I2}, a_n^{I3}, a_n^{I4} \rangle = argmax_{\langle a^{I1}, a^{I2}, a^{I3}, a^{I4} \rangle} Q^G(\boldsymbol{s}_n, \langle a^{I1}, a^{I2}, a^{I3}, a^{I4} \rangle | W)$ [see Eq. (7)]
      For each intersection in $\langle I1, I2, I3, I4 \rangle$
        If the selected signal phase (a_n^{Ii}) differs from the previous one (a_{n-1}^{Ii})
          Implement amber phase for the current phase
        Else
          Implement phase a_n^{Ii}
      Repeat a single step of traffic simulation until t = t + T_a
      Implement $\mathbf{a}_n$ for all intersections
      If replay memory size > D then
        Remove the oldest transition from replay memory
      Draw a sample of M transition experiences $< \mathbf{s}, \mathbf{a}, r, \mathbf{s}' >_j$ randomly from replay memory
      Perform a single SGD step on min $\sum_{j=1}^{M} \left[r_j + \gamma \max_{\mathbf{a}'} Q^G(\boldsymbol{s}'_j, \mathbf{a}' | W^-) - Q^G(\boldsymbol{s}_j, \mathbf{a}_j | W) \right]^2$ w.r.t W
      For every C time intervals, set $W^- = W$

E. The Evaluation and Maximization of the Parametrized Q-Function

It is easy to evaluate the proposed CNN (= the parametrized Q-function) for a given pair of state and action. Once 16 node values of the last hidden layer are computed from a given traffic state (= 64 images: 4 consecutive images × 16 approaches), the Q-function value for a given action (= a signal phase combination for all intersections) can be obtained by summing the 4 node values in the de-facto output layer, each of which comes from a different intersection's signal phases. It also is easy to determine the action that will maximize the Q-function value for a given traffic state. The action selection is the most crucial part in the algorithm to train the extended DQN (see Table I), and it dominates the computing speed of the DQN. From each node group in the de-facto output layer, the phase with the maximum node value is chosen. In terms of the proposed architecture of the CNN, the best joint action is obtained simply by combining the 4 chosen phases.

The above description can be expressed mathematically as follows. The total action-value function can be divided into multiple partial action-value functions [see (5)], each of which has its own specific weight parameters, but also shares common weight parameters. This decomposition seems like the decentralized Q-function method for a MARL wherein partial Q-functions are independent with respect to actions. However, in terms of the formula, the actions of each partial Q-function cannot be stand-alone factors, because all partial Q-functions share some weight parameters (= w). The shared weight parameters can take the correlations between actions into account. However, if all weight parameters are given, according to the proposed CNN architecture, the total Q-function value can be computed by summing the four partial Q-function values [see (5)].

$$\begin{aligned} Q^G\left(s, \left\langle a^{I1}, a^{I2}, a^{I3}, a^{I4} \right\rangle | W\right) &= Q^{I1}\left(s, a^{I1} | \left\langle w, w^{I1} \right\rangle\right) \\ &+ Q^{I2}\left(s, a^{I2} | \left\langle w, w^{I2} \right\rangle\right) + Q^{I3}\left(s, a^{I3} | \left\langle w, w^{I3} \right\rangle\right) \\ &+ Q^{I4}(s, a^{I4} | \left\langle w, w^{I4} \right\rangle) \end{aligned} \tag{5}$$

where, $Q^G(\,)$ is the global Q-function, $Q^{Ii}(\,)$ is a partial Q-function for the i[th] intersection, a^{Ii} is a signal phase for the i[th] intersection, w is a set of shared weight parameters prior to the last FC hidden layer, w^{Ii} is a set of intersection-specific weight parameters directly connecting the last FC hidden layer to nodes for the i[th] intersection within the de-facto output layer, and, thus, W becomes the entire set of weight parameters (= $\langle w, w^{I1}, w^{I2}, w^{I3}, w^{I4} \rangle$).

The core part of Q-learning is to select an action to be taken such that its action-value is maximized [see (3)]. When all weight parameters are given, the maximum value of the total Q-function can be obtained by summing the maximum partial Q-function values, as shown in (6). This conditional independence holds only when all weight parameters of the Q-function (= a CNN) are fixed. It should be noted that partial Q-functions are not independent with respect to the weight parameters.

$$\begin{aligned}&\max_{\langle a^{I1},a^{I2},a^{I3},a^{I4}\rangle} Q^G\left(s,\left\langle a^{I1},a^{I2},a^{I3},a^{I4}\right\rangle|W\right)\\&=\max_{a^{I1}} Q^{I1}\left(s,a^{I1}|\left\langle w,w^{I1}\right\rangle\right)+\max_{a^{I2}} Q^{I2}\left(s,a^{I2}|\left\langle w,w^{I2}\right\rangle\right)\\&+\max_{a^{I3}} Q^{I3}\left(s,a^{I3}|\left\langle w,w^{I3}\right\rangle\right)+\max_{a^{I4}} Q^{I4}\left(s,a^{I4}|\left\langle w,w^{I4}\right\rangle\right).\end{aligned} \tag{6}$$

An action finding is also a trivial task. The signal-phase combination for the entire network when the total Q-function is maximized becomes the set of local signal phases, each of which maximizes its own partial Q-function given that all weight parameters are known [see (7)]. This property will be used for the proposed algorithm in the next subsection to train the proposed CNN within a DQN framework.

$$\begin{aligned}&argmax_{\langle a^{I1},a^{I2},a^{I3},a^{I4}\rangle}\ Q^G\left(s,\left\langle a^{I1},a^{I2},a^{I3},a^{I4}\right\rangle|W\right)\\&=<argmax_{a^{I1}}\ Q^{I1}\left(s,a^{I1}|\left\langle w,w^{I1}\right\rangle\right),\\&argmax_{a^{I2}}\ Q^{I2}\left(s,a^{I2}|\left\langle w,w^{I2}\right\rangle\right),\\&argmax_{a^{I3}}\ Q^{I3}\left(s,a^{I3}|\left\langle w,w^{I3}\right\rangle\right),\\&argmax_{a^{I4}}\ Q^{I4}\left(s,a^{I4}|\left\langle w,w^{I4}\right\rangle\right)>\end{aligned} \tag{7}$$

F. Solution Algorithm for the Extended DQN

The extended DQN was trained using the conventional Q-learning algorithm with a function approximation, and could be considered a model-free and off-policy RL algorithm. The proposed algorithm is listed in Table I. Although the algorithm was written according to the reward computed from the number of WAVEs, the reward defined from the impedance incorporating both WAVEs and space mean speeds could replace it.

Applying both exploration and exploitation is inevitable when training a DQN. The exploration (ϵ) rate determines how many action choices are made on a trial-and-error basis while implementing a RL algorithm. Accordingly, at the remaining rate ($1-\epsilon$), the action choice was made in terms of the incumbent Q-function. The latter case is referred to as an exploitation. The exploration rate started at a higher value (e.g., 1) in the initial stage, and gradually dwindled down to a sufficiently small value (e.g., 0.1) by the end of the iteration.

V. Establishing Simulation Experiment Conditions

A RL-based signal control algorithm cannot be implemented on a real transportation network in the field, since a RL agent must learn on a trial-and-error basis. No drivers can endure a long training period during which they must encounter unexpected traffic jams caused by exploration. Therefore, a plausible simulation environment was set up to test the developed traffic signal controller.

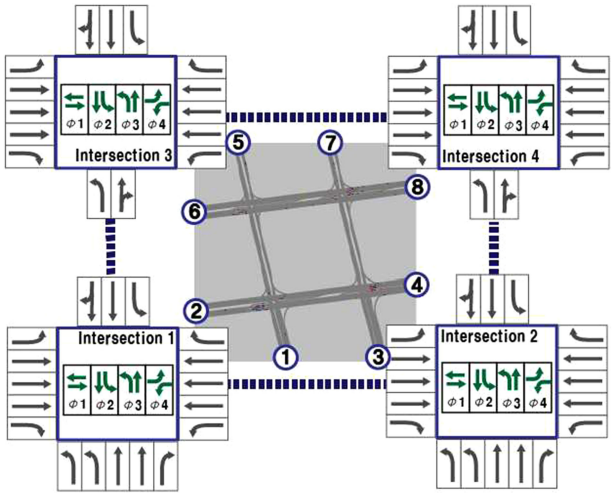

Fig. 3. Simulation environment.

A. Simulation Environment

An imaginary 2×2 urban transportation network was devised as a testbed for simulation experiments to test the proposed traffic signal controller [see Fig. 3]. Traffic lights of four intersections on the network were jointly controlled. Even though the network was imaginary, the operation prototype of each intersection was adopted from an actual intersection located in Seoul, Korea. For each intersection, 4 independent signal phases were commonly applied, which were excerpted from an actual intersection. Lanes were channeled to indicate their directions: straight ahead, left turn, and right turn.

To simulate the RL environment, commercial software for traffic simulation (Vissim) was used to provide complete traffic states and rewards to the proposed traffic signal controller. The simulator was assumed to behave as real traffic following the instructions from the proposed RL agent. The average traffic flows incoming to the network were synthesized bearing in mind that they could possibly be under-saturated. The traffic flows were altered within a range of $\pm 20\%$ around the average values to reflect random fluctuations under actual traffic conditions. The heavy-vehicle proportion of each inflow was randomly assigned a value around 7%. Table II shows average traffic flows applied to the simulation.

Moreover, the average shares of route choice were arbitrarily determined in the form of origin-destination flows, as shown in Table III, and randomly altered within a range of $\pm 10\%$ for each episode. The alterations for inflows and route choice rates meant that some directional flows were temporarily oversaturated while training the model.

Unlike our previous research that used an image covering the entire intersection [25], the proposed controller used only images covering intersection approaches. For the RL environment, the rectangular grey-scale images for 16 intersection approaches were cropped from the entire animation image, and each approach image was placed on the top-left corner of the

TABLE II
TRAFFIC VOLUMES APPLIED TO THE SIMULATION

Approach number	Average traffic flow (veh/hr)	Upper bound of variation	Lower bound of variation	Heavy vehicles ratio
①	964	+8%	-10%	7.06%
②	942	+10%	-11%	7.13%
③	942	+17%	-13%	6.63%
④	1,157	+10%	-13%	6.59%
⑤	580	+12%	-11%	6.90%
⑥	876	+19%	-9%	6.68%
⑦	566	+15%	-12%	7.17%
⑧	1,153	+13%	-13%	7.22%

TABLE III
AVERAGE ROUTE SHARES FOR TRAFFIC SIMULATION

From \ To	①	②	③	④	⑤	⑥	⑦	⑧
①	-	22.1%	7.2%	18.6%	7.9%	18.0%	8.0%	0.1%
②	8.9%	-	8.4%	21.8%	9.2%	21.0%	9.3%	8.9%
③	7.6%	22.0%	-	18.6%	7.8%	17.9%	7.9%	7.6%
④	8.6%	24.8%	8.1%	-	8.9%	20.2%	8.9%	8.6%
⑤	7.7%	22.2%	7.2%	18.7%	-	18.0%	8.0%	7.7%
⑥	8.5%	24.7%	8.0%	20.8%	8.8%	-	8.9%	8.5%
⑦	7.7%	22.2%	7.2%	18.7%	7.9%	18.0%	-	7.7%
⑧	0.0%	22.1%	7.2%	18.6%	7.9%	18.0%	8.0%	-

standardized rectangular image. The size of the standardized image ($= 275 \times 30$) represented the dimension of the largest approach out of the 16 approaches. When a road approach size was smaller than the standardized image, empty pixels of the image were zero-padded. In order to approximate the dynamics of actual traffic states, four consecutive images were captured for a time interval and fed into the model.

B. Establishing Hyper-Parameters

There were three types of hyper-parameters in the proposed RL-based traffic signal controller. First, hyper-parameters that determined the architecture of a CNN that approximated the true Q-function were chosen on a trial-and-error basis. The architecture in Figs. 1 and 2 were selected after testing 10 different combinations of hyper-parameters, each of which was randomly chosen within predefined ranges. According to Bergstra and Bengio [37], random searches of 8 trials matched or outperformed the grid searches of (on average) 100 trials. More recently, Bayesian optimization began to be applied to finding optimal hyperparameters of deep neural networks [38], [39]. This methodology would enhance the model performance in further studies.

Second, the hyper-parameters necessary to implement the proposed RL algorithm are listed in Table IV with a short description. Last, the hyper-parameters of the traffic simulation

TABLE IV
OPERATIONAL HYPER-PARAMETERS ADOPTED IN THE PROPOSED TRAFFIC SIGNAL CONTROL

Hyper-parameter	Description	Applied value
N	Number of episodes	200
Δ	Time step for traffic simulation	0.2 second
T_{max}	Simulation period for each episode	5,000 seconds
T	Time interval for RL action	20 seconds
T_a	Amber time	3 seconds
ε_{max}	Initial probability of exploration	1.0
ε_{min}	Final probability of exploration	0.1
E	Decaying parameter of exploration probability	400,000
D	Size of replay memory set	6,000
M	Size of mini-batch sample	16
C	Cycle for updating target Q-function	5,000 seconds
β	Parameter of the impedance-based reward	0.2
N_f	Frequency of measuring both vehicle counts and space mean speeds within a time interval (T)	4

depended on the default values provided by the Vissim (Vissim user's manual [41]). The time step for the traffic simulation is listed in Table IV. A random seed for the traffic simulation was altered for every episode.

C. Computing Environment

The computing environment for the simulation experiment was as follows. The main memory was 128 GB. Python main-logic and traffic-simulation software (Vissim) was implemented on two CPUs with the following specifications: Intel Xeon(R) CPU E5-2697 v2 @ 2.7 GHz. There were 48 available CPU cores, which exceeded the maximum number of cores ($= 32$) that the traffic simulator 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 an average of 14 minutes to run a single episode. The total computation time for 200 episodes was tantamount to about two days.

V. RESULTS AND COMPARISONS

A. Convergence of the Training Algorithm

The convergence of the proposed RL algorithm was confirmed by plotting the average reward while training the algorithm. For each episode, received rewards were averaged and plotted as a function of the episode number. As expected, the average reward tended to increase initially and then converged to a certain level, because the influence of the Q-function on choosing a signal phase increased as the exploration rate dwindled over

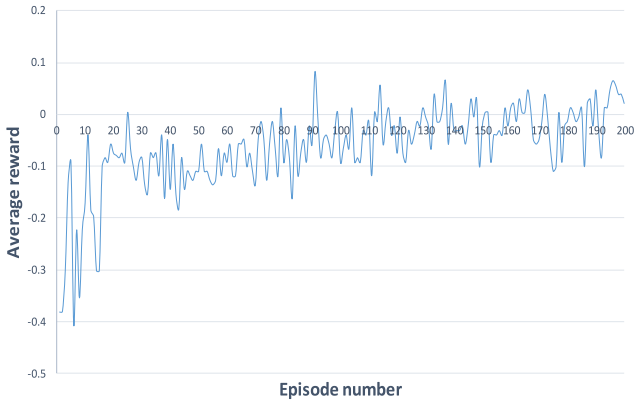

Fig. 4. Convergence of reward for a single-agent RL algorithm.

iterations. Unlike other RL applications wherein the final reward approaches +1, the final reward of the proposed RL algorithm ended up with a value close to 0. This was due to the intrinsic property of the proposed traffic signal control where the reward was defined according to the change in the number of WAVEs or the change in the impedance (= the number of WAVEs weighted by space mean speeds). Receiving a continuously positive reward (= +1) means the queue vanishes and the intersection is cleared, which never happens under real traffic conditions. A zero reward indicates that a certain level of queue length (or impedance value) is maintained in the network. Fig. 4 shows the varying trend of the average reward when executing the proposed RL algorithm that adopted the reward defined according to the number of WAVEs.

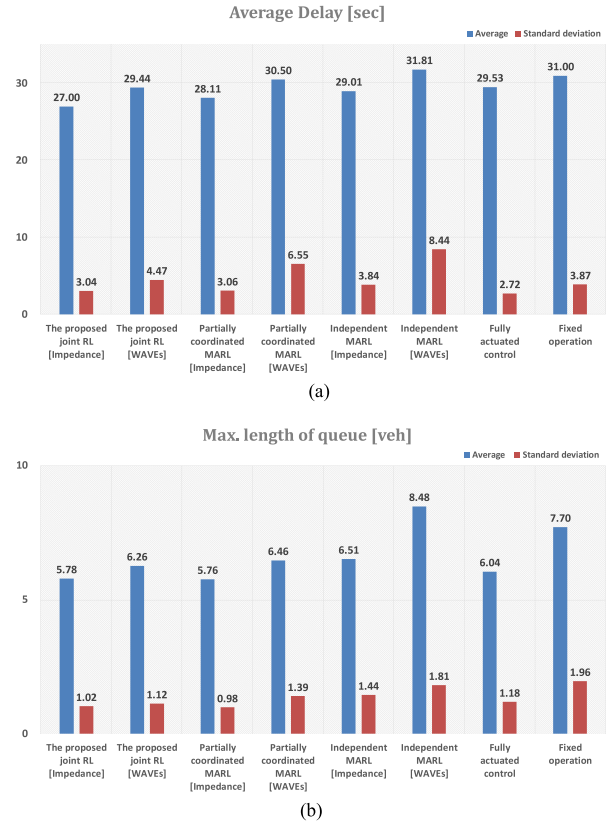

Fig. 5. Performance comparisons. (a) The comparison of average delays. (b) The comparison of maximum average queue lengths measured in the number of vehicles.

B. Establishing Scenarios

The performance of the proposed RL-based traffic signal controller was tested for simulation results collected during 100,000 seconds without exploration. Two performance indices were recorded for every 200 seconds during the test period (= 100,000 seconds): delay and the maximum queue length across all lane groups.

For comparison, a fixed signal operation was set as the baseline, and a fully actuated signal control was implemented as the second reference. For the fixed operation, a constant signal time was allotted for each movement according to the critical movement analysis introduced in the highway capacity manual (HCM). A signal offset was applied to the east-west direction that accommodated the largest amount of through traffic. The condition parameters of fully actuated control such as detector locations, loss time, average vehicle length, the minimum and maximum green times, etc., were set up in an identical manner for each intersection. The chosen parameter values were omitted here for brevity. For the details, readers are referred to Jeon *et al.* [25].

In addition, for comparison with the proposed RL-based traffic signal controller, two types of MARL algorithms were applied under the same simulation conditions. For the first type, agents were assumed to be independent of each other. Each RL agent oversaw the traffic lights in an intersection independently of other intersections, and the action taken depended solely on its own reward and state. The structure of each agent's CNN was set identical to the input unit of the CNN within the extended DQN (see Fig. 1). The output layer was composed of only 4 nodes, each of which represented a distinct signal phase for each intersection. For the second type, a partially coordinated MARL scheme was set up by grouping four intersections into two groups, and each group was controlled with a full coordination among its jurisdictional intersections. The upper and lower two intersections were separately controlled by two different CNNs. The two CNNs had the same structure and shared traffic states for the entire network as input. That is, a partially coordinated MARL model additively decomposed the global Q-function into two local Q-functions for coordination [41].

For each of the RL controls, two different kinds of rewards were attempted: a reward based on the maximum number of WAVEs, and a reward based on the impedance reflecting the impact of the space mean speed. In summary, 8 scenarios were tested and compared in the present study: the baseline (= fixed signal operation), a fully actuated operation, two joint RL algorithms, two independent MARL algorithms, and two partially coordinated MARL algorithms.

C. Test Results From Scenarios

Fig. 5 shows the performance comparisons for the 8 scenarios. The results from statistical testing for the pairwise comparisons are listed in Table V.

TABLE V
PERFORMANCE COMPARISONS BASED ON STATISTICAL TEST

Pairwise comparison		Delay		Queue length	
		t-value	p-value	t-value	p-value
The proposed joint RL [Impedance]	The proposed joint RL [WAVEs]	-10.09	6.38E-22	-7.09	4.76E-12
	Partially coordinated MARL [Impedance]	-5.75	1.52E-08	**0.32***	**7.52E-01***
	Partially coordinated MARL [WAVEs]	-10.84	1.01E-24	-8.82	1.93E-17
	Independent MARL [Impedance]	-9.18	1.17E-18	-9.25	6.55E-19
	Independent MARL [WAVEs]	-11.99	2.79E-29	-29.06	2.18E-109
	Fully actuated operation	-13.87	3.18E-37	-3.73	2.16E-04
	Fixed operation	-18.17	5.07E-57	-19.43	4.87E-63
The proposed joint RL [WAVEs]	Partially coordinated MARL [Impedance]	5.49	6.41E-08	7.51	2.70E-13
	Partially coordinated MARL [WAVEs]	-2.99	2.94E-03	-2.51	1.26E-02
	Independent MARL [Impedance]	1.63*	1.03E-01*	-3.06	2.30E-03
	Independent MARL [WAVEs]	-5.55	4.67E-08	-23.32	6.55E-82
	Fully actuated operation	-0.38*	7.01E-01*	**3.02**	**2.63E-03**
	Fixed operation	-5.90	6.72E-09	-14.26	5.82E-39
Partially coordinated MARL [Impedance]	Partially coordinated MARL [WAVEs]	-7.39	6.14E-13	-9.20	9.51E-19
	Independent MARL [Impedance]	-4.10	4.85E-05	-9.63	3.07E-20
	Independent MARL [WAVEs]	-9.22	8.62E-19	-29.55	1.10E-111
	Fully actuated operation	-7.76	4.99E-14	-4.08	5.20E-05
	Fixed operation	-13.10	6.68E-34	-19.80	8.44E-65
Partially coordinated MARL [WAVEs]	Independent MARL [Impedance]	4.39	1.40E-05	-0.56*	5.77E-01*
	Independent MARL [WAVEs]	-2.74	6.33E-03	-19.79	8.80E-65
	Fully actuated operation	**3.06**	**2.35E-03**	**5.15**	**3.74E-07**
	Fixed operation	-1.47*	1.42E-01*	-11.54	1.81E-27
Independent MARL [Impedance]	Independent MARL [WAVEs]	-6.75	4.06E-11	-19.05	3.48E-61
	Fully actuated operation	-2.47	1.38E-02	**5.65**	**2.77E-08**
	Fixed operation	-8.16	2.71E-15	-10.94	4.05E-25
Independent MARL [WAVEs]	Fully actuated operation	**5.75**	**1.56E-08**	**25.25**	**2.99E-91**
	Fixed operation	**1.95**	**5.17E-02**	**6.54**	**1.55E-10**
Fully actuated operation	Fixed operation	-6.95	1.16E-11	-16.22	7.52E-48

The negative sign represents that the approach in the first column outperforms that in the second column.
Figures with superscript $*$ indicate a statistical insignificance at the 0.05 level.
Bold figures indicate results that contradict prior expectations.

The proposed RL controller with an impedance-based reward that reflected the space mean speeds recorded the shortest delay. This result is noteworthy because neither the reward nor the state was computed directly from the exact delays that indicated the service level of the traffic state. It was confirmed that the state elicited from images and the reward approximated by WAVEs and space mean speeds could minimize the delay.

Regarding the maximum queue length, the proposed RL-algorithm with the impedance-based reward also outperformed other algorithms except for the partially coordinated MARL with the impedance-based reward. Although the performance of the proposed RL algorithm seems to be inferior to that of the partially coordinated MARL with the impedance-based reward, the difference is very small and the inferiority is not statistically significant.

The excellence of the proposed RL algorithm both in the delay and the maximum queue length implies that the scheme of sharing and fixing weight parameters worked properly and thus a solution closer to the global optimum was found. In addition, for both the proposed and MARL controls, the algorithm with the reward that was computed based solely on the maximum number of WAVEs was inferior to that with the impedance-based reward that reflected the space mean speeds. Even fully actuated operations outperformed the RL algorithm with the WAVE-based reward. This implies that the reward definition is the most important in guaranteeing the performance of a RL algorithm. In this regard, the newly devised impedance was qualified to ensure the success of the proposed RL algorithm. Furthermore, the impedance incorporating the space mean speed proved to be successful in approximating a true delay.

Regarding the stability of algorithms, RL-based algorithms with the reward based on the maximum WAVEs recorded a larger standard deviation in delay than those of the two reference control algorithms (= fixed operation and fully actuated algorithm). This means the RL algorithms might be less stable than the existing control algorithms unless the reward was carefully defined. Casas [26] also pointed out that rewards fluctuated after convergence when a Q-learning was adopted in the traffic signal control.

In the future, the performance should be tested for fully over-saturated traffic conditions. However, at this stage it is difficult for a traffic simulator to properly cover all the traffic

conditions. A field test is inevitable for the proposed approach prior to application to real conditions.

VI. Conclusion

The present study succeeded in using a RL-based algorithm to jointly control the traffic signals in an urban transportation network. Unlike the previous efforts to control traffic signals on multiple intersections wherein partially coordinated MARL algorithms have been adopted, a scalable RL algorithm was developed to increase the possibility of reaching a global optimum.

The contributions of the present study are three-fold. First, two novel methodologies were devised to overcome the curse of dimensionality. The CNN parameters used to recognize traffic states were shared for all intersection approaches, and those connecting the last hidden layer and the output layer were fixed at 1. The former resolved the problem of a large state space, whereas the latter resolved the problem of an explosive action space.

Second, we devised a RL-based traffic signal controller that depended solely on video images. Traffic states were fed into the model as video images without the use of feature-engineering tools. Last, the reward that can be obtained from currently available sources (= images) was used, unlike previous RL-based traffic light controllers that adopted delay-based rewards that cannot be measured in the field. Putting the ideas together led to an extended DQN that can jointly control multiple signalized intersections in an urban transportation network.

The proposed RL-based traffic signal controller represents the first step toward the attainment of robust artificial intelligence. It was clear in the current stage that a CNN can recognize traffic states through video images. However, artificial intelligence cannot yet provide an exact measure of the on-site delay. The delay is a key variable for deriving immediate rewards and for allocating traffic signal phases. For further study, measuring the delay based solely on sensory data will be studied. Furthermore, the developed algorithm will be tested for an actual network in real time in an attempt to overcome the limitations of algorithms that can only test simulated environments.

References

[1] B. Abdulhai and L. Kattan, "Reinforcement learning: Introduction to theory and potential for transport applications," *Can. J. Civil Eng.*, vol. 30, no. 6, pp. 981–991, 2003.

[2] A. L. C. Bazzan and F. Klügl, "A review on agent-based technology for traffic and transportation," *Knowl. Eng. Rev.*, vol. 29, no. 3, pp. 375–403, 2014.

[3] A. L. C. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Auton. Agents Multi-Agent Syst.*, vol. 18, no. 3, pp. 342–375, 2009.

[4] Z. Liu, "A survey of intelligence methods in urban traffic signal control," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 7, no. 7, pp. 105–112, 2007.

[5] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," in *Autonomic Road Transport Support Systems*. Cham, Birkhäuser, 2016, pp. 47–66.

[6] X. Huang, Y. Tingting, Q. Guanhua, and R. Yizhi, "Deep reinforcement learning for multimedia traffic control in software defined networking," *IEEE Netw.*, vol. 32, no. 6, pp. 35–41, Nov/Dec. 2018.

[7] L. Busoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations Multi-Agent Syst. Appl.-1*, vol. 310, pp. 183–221, 2010.

[8] M. Steingrover *et al.*, "Reinforcement learning of traffic light controllers adapting to traffic congestion," in *BNAIC*, pp. 216–223, 2005.

[9] L. Kuyer *et al.*, "Multiagent reinforcement learning for urban traffic control using coordination graphs," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Berlin, Heidelberg, Springer, 2008.

[10] D. Houli, L. Zhiheng, and Z. Yi, "Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network," *EURASIP J. Advances Signal Process. 2010*, vol. 7, no. 724035, 2010.

[11] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, Sep. 2013.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–33, 2015.

[13] S. Mikami and Y. Kakazu, "Genetic reinforcement learning for cooperative traffic signal control," in *Proc. IEEE World Congress Comput. Intell., Proc. First IEEE Conf. Evol. Comput.,* 1994, pp. 223–228.

[14] M. A. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proc. Mach. Learn.: Proc. Seventeenth Int. Conf. (ICML'2000)*, 2000.

[15] G. Dulac-Arnold *et al.*, "Deep reinforcement learning in large discrete action spaces," *arXiv:1512.07679*, 2015.

[16] M Abdoos *et al.*, "Traffic light control in non-stationary environments based on multi agent Q-learning," in *Proc. 2011 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, 2011, pp. 1580–1585.

[17] M. Abdoos *et al.*, "Hierarchical control of traffic signals using Q-learning with tile coding," *Appl. Intell.*, vol. 40, no. 2, pp. 201–213, 2014.

[18] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 128–135, Jun. 2010.

[19] B. Bakker *et al.*, "Traffic light control by multiagent reinforcement learning systems," in *Interactive Collaborative Information Systems*. Berlin, Heidelberg, Springer, 2010, pp. 475–510.

[20] J. Jin, and X Ma., "Adaptive group-based signal control by reinforcement learning," *Transp. Res. Procedia*, vol. 10, pp. 207–216, 2015.

[21] J. Iša *et al.*, "Reinforcement learning of traffic light controllers adapting to accidents," *Des. Organisation Auton. Syst.*, pp. 1–14, 2006.

[22] M. A. Khamis and W. Gomaa, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Eng. Appl. Artif. Intell.*, vol. 29, pp. 134–151, 2014.

[23] W. Genders, and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv:1611.01142*, 2016.

[24] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1243–1253, Feb. 2019.

[25] H. Jeon, J. Lee, and K. Sohn, "Artificial intelligence for traffic signal control based solely on video images," *J. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 433–445, 2018.

[26] N. Casas, "Deep deterministic policy gradient for urban traffic light control," *arXiv:1703.09035*, 2017.

[27] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," *arXiv:1802.08757*, 2018.

[28] Q. Qi *et al.*, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, May 2019.

[29] L. Minne *et al.*, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. The World Wide Web Conf. ACM*, 2019. pp. 983–994.

[30] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[31] J. Glick, "Reinforcement learning for adaptive traffic signal control," Final Project, CS 229 (Machine Learning), Stanford University, 2015.

[32] L. Li, Y. Lv, and F. Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 10. Jul. 2016.

[33] P. G. Balaji *et al.*, "Urban traffic signal control using reinforcement learning agents," *IET Intell. Transport Syst.*, vol. 4, no. 3, pp. 177–188, Sep. 2010.

[34] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," *Eng. Appl. Artif. Intell.*, vol. 26, no. 5–6, pp. 1575–1587, 2013.
[35] J. Chung and K. Sohn, "Image-based learning to measure traffic density using a deep convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1670–1675, May 2018.
[36] J. Lee, S. Roh, J. Shin, and K. Sohn, "Image-based learning to measure the space mean speed on a stretch of road without the need to tag images with labels," *Sensors*, vol. 19, no. 5:1227, 2019.
[37] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, pp. 281–305, Feb. 2012.
[38] A. Klein *et al.*, "Fast Bayesian optimization of machine learning hyperparameters on large datasets," in *Artif. Intell. Statist., AISTATS*, 2017, pp. 528–536.
[39] J. Snoek *et al.*, "Scalable Bayesian optimization using deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2015.
[40] C. Guestrin, M. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," *ICML*, vol. 2, pp. 227–234, Jul. 2002.
[41] Vissim User's manual, [Online]. Available: http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/

Jincheol Lee received the graduate degree from Chung-Ang University, Seoul, South Korea. He is currently working toward the Ph.D. degree the theme of which covers applying artificial intelligence to traffic surveillance.

Jiyong Chung currently working toward the graduate degree from Chung-Ang University, Seoul, South Korea. His research interest includes artificial intelligence.

Keemin Sohn was born in Seoul, South Korea, in 1968. He educated from Seoul National University (SNU), Seoul, South Korea and the Ph.D. degree in transportation planning from SNU, in 2003. He is a Professor with the Department of Urban Engineering, Chung-Ang University, Seoul, South Korea. His research interests include applications of artificial intelligence to transportation engineering and planning.