2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Deep Merging: Vehicle Merging Controller Based on Deep Reinforcement Learning with Embedding Network

Ippei Nishitani[1], Hao Yang[2], Rui Guo[2], Shalini Keshavamurthy[2], and Kentaro Oguchi[2]

Abstract— **Vehicles at highway merging sections must make lane changes to join the highway. This lane change can generate congestion. To reduce congestion, vehicles should merge so as not to affect traffic flow as much as possible. In our study, we propose a vehicle controller called Deep Merging that uses deep reinforcement learning to improve the merging efficiency of vehicles while considering the impact on traffic flow. The system uses the images of a merging section as input to output the target vehicle speed. Moreover, an embedding network for estimating the controlled vehicle speed is introduced to the deep reinforcement learning network architecture to improve the learning efficiency. In order to show the effectiveness of the proposed method, the merging behavior and traffic conditions in several situations are verified by experiments using a traffic simulator. Through these experiments, it is confirmed that the proposed method enables controlled vehicles to effectively merge without adversely affecting to the traffic flow.**

I. Introduction

Automation of vehicle control on highway has been rapidly advancing over the years. Intelligent vehicles are expected to reduce traffic accidents and improve traffic efficiency. Traffic congestion on highway affects millions of people. Especially, vehicles at highway merging sections, such as on-ramp and lane-drop bottlenecks, have to make lane changes, which generate traffic oscillations and additional congestion [1]. Both the main-lane and on-ramp traffic are potentially congested due to the irregular lane change behaviors and unexpected brake maneuvers. To reduce congestion, controlled vehicles should effectively merge onto the main-lane at an appropriate timing and speed so as not to affect traffic flow as much as possible. Therefore, the purpose of our study is to develop the system that enables controlled vehicles to effectively merge onto the main-lane while minimizing the traffic impact on the main-lane and on-ramp.

Several rule-based algorithms for merging [2]-[4] have been proposed and their effectiveness are verified. However, they may not be able to adapt to the unexpected and complex situations where interactions with surrounding vehicles work in the real world. Vehicle control by machine learning is effective in such a complicated traffic environment [5]-[7].

In our study, we propose a vehicle controller called Deep Merging that uses deep reinforcement learning (RL) [14]-[16] to improve the merging efficiency of vehicles while considering the impact on traffic flow. The system uses the current and past images of a merging section as input to output the target speed of the controlled vehicle. It is assumed that these images are created from information collected by the roadside camera and / or onboard sensors. Furthermore, the merging behavior considering the impact on traffic flow is realized by setting the average speed of all vehicles after merging as a reward for deep RL. However, it is difficult to make a machine learning network perform appropriate feature extraction for output speed selection from image information. This may cause inadequate and slow network convergence. In order to perform appropriate feature extraction, several studies have been proposed to introduce an embedding network for an auxiliary task as prior knowledge, and their effectiveness has been confirmed [8][9]. Therefore, to improve the learning efficiency of deep RL, an embedding network for estimating the controlled vehicle speed is introduced to the deep RL network architecture. This embedding network enables the machine learning network to explicitly estimate the speed of the vehicle and to efficiently understand the state of the controlled vehicle. As a result, it is expected that it will be easier to make machine learning for vehicle control select appropriate behaviors, and reduce the number of trials required for training to achieve a good performance. The contributions of our study are listed as follows:

- This is the first paper to introduce an embedding network for estimating dynamic traffic conditions, such as a controlled vehicle speed, to vehicle merging controller. Appropriate feature for vehicle control can be extracted by introducing the embedding network. This boosts the learning efficiency of deep RL.
- We can achieve the merging behavior considering the impact on traffic flow by setting the average speed of all vehicles after merging as a reward for deep RL. This system enables controlled vehicles to effectively merge onto the main-lane while minimizing the traffic impact on the main-lane and on-ramp.

The remainder of this paper is organized as follows: First, the related works is described in Section II, followed by the algorithm for the proposed method in Section III. Various experiments of the proposed method are given in Section IV. Finally, our conclusions are presented in Section V.

II. Related Works

By incorporating deep learning and achieving high quality feature representation, deep RL has achieved significant triumph. From playing Atari games using Deep Q-Network (DQN) [14] to the overwhelming superiority in the game of Go [13], deep RL is shaping the way to understand the world in an unprecedented speed and help solve many real-world problems. Also, Deep RL has been applied to systems that control or navigate vehicles in several previous works.

Schutera et al. [7] applied deep RL to train a car agent with the goal of achieving the highest average speed over a period

[1] Toyota Motor Corporation, Japan; ippei_nishitani@mail.toyota.co.jp
[2] Toyota Motor North America, USA; {hao.yang, rui.guo, shalini.keshavamurthy, kentaro.oguchi}@toyota.com

of time in MIT DeepTraffic simulation. The system enables control of acceleration and lane change behavior of multiple car agents in the main-lane. However, this paper does not cover scenes where the car agents merge from the ramp to the main-lane. As merging situation are more severe in adjusting relative speed and timing compared to lane changing situation, this system may not be able to achieve efficient merging behavior. Wang et al. [6] proposed to apply RL algorithm on the vehicle agent to find an optimal driving policy to assist vehicles merging onto the main-lane. The system needs to have the speed and position of the merging vehicle, the gap lag vehicle, and the gap front vehicle, as the input for the RL. Due to the dynamics of individual vehicles, it is not easy to identify them accurately, especially when the traffic volume on the main-lane is very high. And, the reward function is set to measure the safety, smoothness, and the promptness of the merging maneuver, and it is not intended to consider traffic flow. Mirowski et al. [9] proposed the method using deep RL allows agents to learn to navigate multiple cities and to traverse to target destinations that may be kilometers away in Google Street View. The auxiliary heading prediction task is applied to speed up learning by providing extra gradients as well as relevant information. However, this paper focuses on navigation in the city and does not take into account vehicle dynamics or other vehicle behavior. Therefore, this system is not focused on applying in complicated traffic environment where complex interactions between vehicles work.

The focus of our study is to improve the merging efficiency without adversely affecting to the traffic flow by instructing the target speed to the ego vehicle in a merging scenario using deep RL. The description of the proposed method is given in the next section.

III. Deep Merging

In this section, the methodologies of the vehicle control that include a speed controller using deep RL is described. We define the controlled vehicle whose speed is instructed by Deep Merging as the ego vehicle. We first describe vehicle control, then deep RL for speed control, followed by the embedding network.

A. Vehicle Control

The traffic simulator used in this study [11][12] controls all vehicles in its simulation with safety regulations. That is, it ensures that vehicles do not collide during simulation and not violate any traffic rules, e.g. driving beyond speed limit and reverse driving. The ego vehicle behavior is also basically controlled by the vehicle controller of the traffic simulator, but only the target speed is instructed by Deep Merging as shown in Fig. 1. Therefore, the ego vehicle basically follows the instructed speed, but if the safety regulations are violated, the speed is corrected to avoid collisions. Since Deep Merging aims to adjust the time of merging and speed, the speed instruction is not given after merging. Deep Merging adjusts the ego-speed and achieves an effective merge while minimizing impact on the main-lane and on-ramp traffic.

B. Deep Reinforcement Learning

In a RL problem, an agent and the environment which is formulated as a Markov Decision Process (MDP) interact at discrete time steps. The agent observes state s and selects an

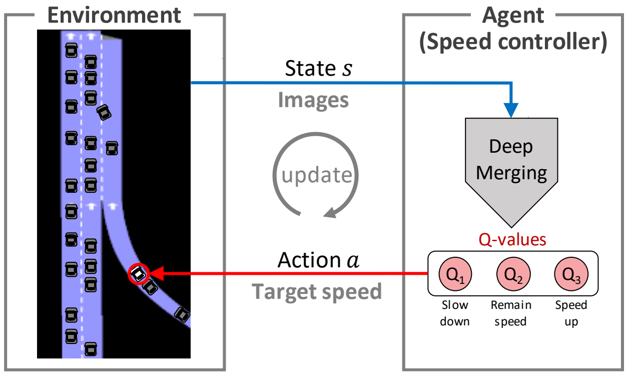


Figure 1. Overview of the speed controller for the controlled vehicle

action a according to its policy π. Then, the environment responds to the action and presents new state s' and the agent observes the reward r. In this research, traffic conditions are approximated as MDP by using a traffic simulator [6][17].

RL agent aims to maximize expected discounted return, which is defined as:

$$R = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau \tag{1}$$

where r is a discount factor that trades off the importance of immediate and future rewards. Furthermore, consider a stochastic policy π, according to which the agent behaves, the value of the state-action pair Q^π and the value of the state V^π are defined as follows:

$$Q^\pi(s,a)=\boldsymbol{E}[R_t|s_t=s,a_t=a,\pi] \tag{2}$$

$$V^\pi(s)=\boldsymbol{E}_{a\sim\pi(s)}[Q^\pi(s,a)] \tag{3}$$

Using dynamic programming, the state-action value function, i.e., Q-function can be recursively computed and the optimal Q-function $Q^*(s,a)$ satisfies the Bellman equation:

$$Q^\pi(s,a)=\boldsymbol{E}_{s'}\left[r+\gamma\boldsymbol{E}_{a'\sim\pi(s')}\left[Q^\pi\left(s',a'\right)\right]|s,a,\pi\right] \tag{4}$$

$$Q^*(s,a)=\boldsymbol{E}_{s'}\left[r+\gamma\max_{a'} Q^*\left(s',a'\right)|s,a\right] \tag{5}$$

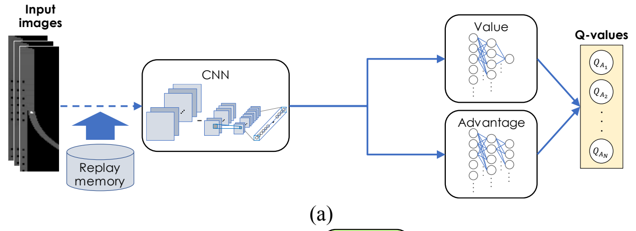
where s' is the new state and a' is the new action at new state.

Usually the value function is high dimensional, and it is difficult to manually formulate. Consequently, the value function is represented using a function approximator, such as a neural network. Deep Merging uses DQN [14] as a deep RL algorithm. In addition, we apply double DQN [16] and dueling DQN [15] architecture to Deep Merging from the viewpoint of learning stability and convergence speed. In DQN approach, the neural network is used for Q-value approximation. The network architecture of Deep Merging is shown in Fig 2 (a). The objective is to minimize the following loss function $L(\theta)$:

$$L(\theta)=\boldsymbol{E}_{s,a,r,s'}\left[\left(y^{\mathrm{DQN}}-Q(s,a;\theta)\right)^2\right] \tag{6}$$

$$y^{\mathrm{DQN}}=r+\gamma\max_{a'} Q\left(s',a';\theta^-\right) \tag{7}$$

where θ is the parameter of the network and θ^- is the parameter of the separate network, called target network. The target network parameter θ^- is synchronized to the network



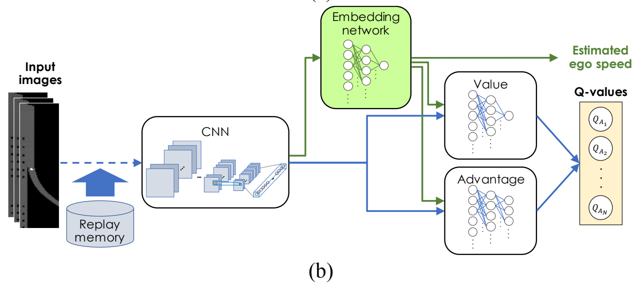


Figure 2. Network architecture; (a) Deep Merging, (b) Deep Merging with embedding network

parameter θ every synchronization cycle. In double DQN [16], to reduce the overestimation of Q-value, the y^{DQN} is given as:

$$y^{\mathrm{DQN}}=r+\gamma Q\left(s', \underset{a'}{\operatorname{argmax}} Q\left(s',a';\theta\right);\theta^{-}\right) \tag{8}$$

DQN with experience replay is used to successfully train the network. During Q-network training, instead of only using the current experience in standard temporal-difference learning (TD-learning), the network is trained by sampling mini-batches of experiences *s*, *a*, *r*, *s′* from the experience replay memory uniformly at random.

In order to focus on the importance of taking certain actions, dueling DQN [15] introduces the advantage function, relating to the value function and Q-functions:

$$A^{\pi}(s,a)=Q^{\pi}(s,a)-V^{\pi}(s) \tag{9}$$

$$\mathbf{E}_{a\sim\pi(s)}[A^{\pi}(s,a)]=0 \tag{10}$$

As a result, the Q-function can be expressed as:

$$Q(s,a;\theta)=V(s;\theta)+A(s,a;\theta) \tag{11}$$

However, this equation has the issue of identifiability. Therefore, the modified Q-function is expressed as:

$$Q(s,a;\theta)=V(s;\theta)+A(s,a;\theta)-\frac{1}{|A|}\sum_{a'}A\left(s,a';\theta\right) \tag{12}$$

The settings of state, action and reward for the deep RL are described as follows.

1) State: To achieve effective merging onto the main-lane, the ego vehicle needs to know not only its own state but also the state of its surrounding vehicle [6]. Also, vehicle controlled by deep RL should be given information on road shape to make the ego vehicle aware of reachable area. Therefore, we use the images that contain information on the road shape, surrounding vehicles and the ego vehicle. Each road object is given a different color. It is assumed that the image of a merging section is created from information collected by the roadside camera and / or onboard sensors. The experiments in this paper are fully observed, that is, all vehicle positions and velocities in the section defined by the state are available. In addition, to understand the dynamics including acceleration of the ego vehicle and surrounding vehicles, Deep Merging uses past and current images as state representation. Increasing the number of images can provide more detailed dynamic information, but will take longer training convergence.

2) Action: This can be set as a target control command to instruct changes in the ego vehicle such as steering angle, speed, acceleration or control flag. In this paper, Deep Merging controls the target speed by giving the target acceleration. The target speed is obtained by adding the output acceleration to the current speed. Thus, Deep Merging can adjust the merging speed and timing of the ego vehicle. Increasing the number of acceleration options can provide smoother speed control, but will take longer to converge.

3) Reward: Deep Merging agent learns the action to maximize accumulated future reward. To achieve the effective merging onto the main-lane while minimizing the traffic impact, the impact of the merging on traffic flow should be considered in reward setting. In our study, rewards are given only when the merging is completed. That is, the reward is 0 when the ego vehicle is on the ramp. The merging completion reward is set as the average speed of all vehicles from merging to reaching the terminal area of the road. The merging completion reward r_{comp} is given as:

$$r_{\mathrm{comp}}=\frac{1}{T}\sum_{t=0}^{T-1}\left(\frac{1}{N_t}\sum_{n=0}^{N_t-1}\frac{v_n^t}{V_{\max}}\right)^2 \tag{13}$$

where T is the number of steps from merging to reaching the terminal area, N_t is the number of vehicles in target section at step t, v_n^t is the speed of each vehicle and $V_{\max}$ is the value to normalize the speed. With this reward setting, the ego vehicle adjusts its speed to increase the average speed after merging. Therefore, it is expected that smooth merging behavior, in which the ego vehicle avoids the behavior that causes slow traffic flow and local congestion, will be achieved. However, the number of experiences that the ego vehicle has completed the merging much less than the number of experiences that it is on the ramp. Thus, this reward is very sparse and in most experiences it will be 0. To solve this problem, a certain number of rewarded experiences is included in mini-batches of experiences to train the network with experience replay.

C. Embedding Network for Deep Reinforcement Learning

As shown in Fig. 2 (a), Deep Merging agent first uses the convolutional neural network (CNN) to extract features from input images, and then calculates Q-value of each action. In order to make Deep Merging choose the appropriate output speed according to the situation, it is important to understand the dynamic traffic conditions of the ego vehicle and surrounding vehicles. However, it is difficult to make a CNN perform appropriate feature extraction for output speed selection from sequential image information. This may cause inadequate and slow network convergence. In order to improve the learning efficiency of deep RL, an embedding network for estimating dynamic traffic conditions is introduced to the deep RL network architecture as shown in Fig. 2 (b). In this paper, the embedding network disentangles estimated ego-speed according to the reward and action settings. Estimated ego-speed from the embedding network is added as a feature for both the advantage function and the value function. The loss function for speed estimation and Q-value estimation is given as:

$$L(\theta)=\boldsymbol{E}_{s,a,r,s',v_e}\left[\left(y^{\mathrm{DQN}}-Q(s,a;\theta)\right)^2+w_e(v_e-V_e(s;\theta))^2\right] \tag{14}$$

where v_e is the actual ego-speed, w_e is a weight of the embedding network loss, and V_e is the function of the embedding network to estimate ego-speed from state s. A shared CNN for feature extraction is trained for two tasks: Q-value calculation and ego-speed estimation. Actual ego-speed information is needed for loss function calculations in the training phase, but not in the testing phase. This embedding network enables the machine learning network to explicitly estimate the ego-speed and to efficiently understand the state of the ego vehicle. Therefore, it is expected that this embedding network boosts the learning efficiency of deep RL.

IV. Experiments and Analysis

To verify the effectiveness of Deep Merging, we assumed a situation in which the ego vehicle merges onto the main-lane from the on-ramp. We use a traffic simulator [11][12] as a training and testing environment. The traffic simulator used in the experiments models the states and actions at the level of individual vehicles, and have been broadly used for empirical evaluation of several complex automated tasks [10]. Traffic simulator and vehicle controller communicate by using API of the traffic simulator every action update interval. Deep Merging outputs the instructed speed according to the current state s to the traffic simulator. The traffic simulator receives the instructed speed for the ego vehicle as action a from vehicle controller, responds to the action a, presents new state s' and calculates the reward r. Deep Merging agent is implemented in python with TensorFlow [18].

A. Experimental Conditions

TABLE I. and TABLE II. show the experiment settings of the traffic simulator and Deep Merging, and Fig. 3 shows an overview of the system. Deep Merging is trained on a highway with two-lane main-lane and on-ramp using a traffic simulator. All vehicles on the main-lane and on-ramp are controlled by the vehicle controller of the traffic simulator based on the initial entering speed and the maximum desired speed setting. However, the target speed is instructed from Deep Merging only for the ego vehicle. It is assumed that there is always one ego vehicle in the environment, that is, when the current ego vehicle reaches the terminal area of the highway, the next ego vehicle enters from the on-ramp.

According to our simulation scenario settings, we set action as acceleration (+0.3 G), remaining speed (+0 G) and deceleration (-0.3 G). Also, we set state observation as 3 grayscale images of -1.0, -0.5 and 0 seconds. We use 80 x 256 pixel images, including on-ramp and first lane of main-lane, in which the ego vehicle is white, surrounding vehicles and inaccessible places are black and the road is gray.

The network of Deep Merging consists of convolutional layers followed by embedding layers, value function layers

TABLE I. *Parameters of Traffic Simulator*

Traffic simulator terms		**Value**
Simulation step		0.1 s
Action update cycle		0.5 s (5 steps)
Training duration		1,500,000 cycle
Testing duration		30,000 cycle
Traffic demand of main-lane		4,620 vehicles/h
Traffic demand of on-ramp		420 vehicles/h
Main-lane vehicle	Initial speed	80 km/h
	Maximum desired speed	80 km/h
On-ramp vehicle	Initial speed	50 km/h
	Maximum desired speed	80 km/h
Ego vehicle	Initial speed	50 km/h
	Maximum speed	120 km/h

TABLE II. *Parameters of Deep Merging*

Deep Merging terms	**Value**
Replay database size	50,000 set
Target network synchronization cycle	10,000 cycle
Learning rate	0.00025
Discount Factor γ	0.95
Mini batch size	32 set
Rewarded experience rate	25% (8 set)
Weight of the embedding network loss w_e	0.316
Value for reward normalization $V_{\max}$	80

TABLE III. *Network Architecture*

Network		**Value**		
		Kernel	***Stride***	***Feature maps / Units***
Convolutional layers	1st conv.	8x8	4x4	32
	2nd conv.	4x4	2x2	32
	3rd conv.	4x4	2x2	64
	4th conv.	4x4	2x2	128
	FC	-	-	3072
Embedding layers	FC	-	-	1024
	FC	-	-	1
Value function layers	FC	-	-	1024 (+ 1)
	FC	-	-	1
Advantage function layers	FC	-	-	1024 (+ 1)
	FC	-	-	3

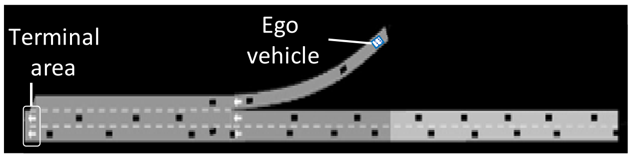


Figure 3. Overview of the system

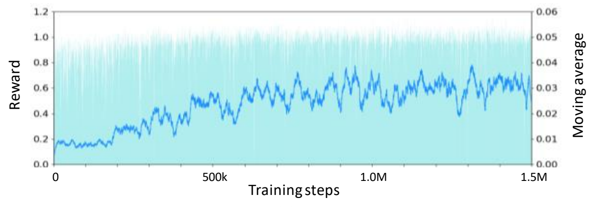
and advantage function layers as shown in Fig. 2 (b). The detail of the network architecture is shown in TABLE III. When the embedding network is introduced to the network, the output of embedding network is added to value function layer and advantage function layer as a feature.

For comparison with the proposed method, we define the default vehicle controller in traffic simulator as method A and vehicle controller with Deep Merging without an embedding network as method B. For this experiment, method A, method B and the proposed method are applied to the ego vehicle, and we compared the traffic condition using three methods in experiment 1. In addition, the traffic volume of the highway changes over time in the real world. Therefore, to verify the robustness of each method, we simulated the vehicle behavior by changing the traffic demand of main-lane or on-ramp in experiment 2 and 3.

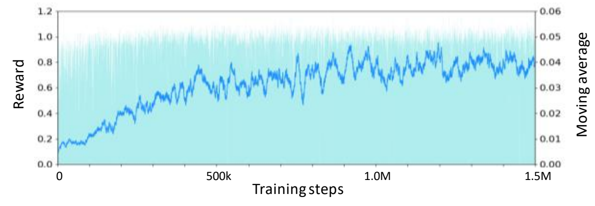
To verify the merging efficiency and the impact on traffic flow, the experimental results were evaluated in terms of traffic volume of ego vehicle, average ego-speed, average on-ramp speed and average main-lane speed. According to the experimental conditions, traffic volume of ego vehicle is equal to the number of merging successes of ego vehicle. Average ego-speed is the average speed of ego vehicle of all testing steps. Average on-ramp speed and average main-lane speed are the average speeds of all vehicles of all testing steps in each lane.

B. *Experimental Results and Discussion*

In training, we trained Deep Merging agent for 1,500,000 steps. Fig. 4 shows the training reward curves of method B and the proposed method. The moving average of reward of the proposed method raised more quickly than method B. Moreover, in method B, the moving average of reward converged to about 0.03, whereas in the proposed method, it converged to about 0.04. Thus, we confirmed that the embedding network boosts the learning efficiency and performance of deep RL in training phase.



(a)



(b)

Figure 4. Training reward curve; (a) Method B, (b) Proposed method

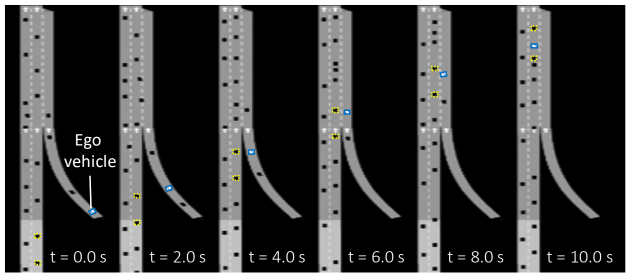


Figure 5. Merging behavior of the ego vehicle using proposed method

In experiment 1, we conducted a testing for 30,000 steps using a traffic simulator with the same parameters as during training. Fig. 5 shows the merging behavior of the ego vehicle using the proposed method in experiment 1. The ego vehicle was able to select an appropriate output speed according to the state including surrounding vehicle behavior. Therefore, the ego vehicle succeeded to merge onto the main-lane smoothly by adjusting its speed. In addition, we compared the experimental results of method A, method B and the proposed method, as given in TABLE IV. Bold face in TABLE IV. indicates best performance. Method B improved traffic volume of ego vehicle, average ego-speed and average on-ramp speed with less impact on average main-lane speed by using deep RL compared to method A. Moreover, the proposed method further improved these 3 terms by introducing the embedding network. Traffic volume of ego vehicle of the proposed method was improved by 31.5% compared to method A. On the other hand, average main-lane speed of the proposed method was 2.2% lower than that of method A. This is due to the fact that the vehicles on the main-lane does not have to slow down frequently if the traffic volume of ego vehicle is less. Therefore, it can be said that the proposed method improved the merging efficiency with less impact on average main-lane speed. Thus, the results of experiment 1 confirm that the proposed method is able to realize effective merging behavior while minimizing the traffic impact on the main-lane and on-ramp by using deep RL taking into account the ego vehicle speed and the average speed of all vehicles after merging.

In experiment 2, we changed the traffic demand of the main-lane of the traffic simulator from 4,620 vehicles/h to 4,200 vehicles/h and 5,040 vehicles/h, and conducted a testing for 30,000 steps using pre-trained model. Fig. 6 shows all results of experiment 2. As shown in Fig. 6 (a)-(c), the proposed method had the highest traffic volume of ego vehicle, average ego-speed and average on-ramp speed for all main-lane demand conditions. Of all the main-lane demand conditions, the condition of 4,200 vehicles/h generated the smoothest traffic flow, and the proposed method had 1,093 vehicles, 60.0 km/h, and 52.5 km/h for traffic volume of ego vehicle, average ego-speed and average on-ramp speed respectively. This may be due to the fact that the smaller the main-lane demand, the easier it is for the ego vehicle to merge smoothly, which results in less traffic congestion due to irregular lane change behaviors and unexpected braking. On the other hand, average main-lane speed had almost the same value in all methods and all conditions.

TABLE IV. EXPERIMENTAL RESULTS OF EXPERIMENT 1

Methods	**Traffic volume of ego vehicle**	**Ave. ego-speed [km/h]**	**Ave. on-ramp speed [km/h]**	**Ave. main-lane speed [km/h]**
Method A	709	38.8	37.6	**76.6**
Method B	844	46.2	39.4	75.2
Proposed method	**932**	**51.1**	**42.9**	74.9

In experiment 3, we changed the traffic demand of the on-ramp of the traffic simulator from 420 vehicles/h to 210 vehicles/h and 630 vehicles/h, and conducted a testing for 30,000 steps using pre-trained model. Fig. 7 shows all results of experiment 3. As shown in Fig. 7 (a)-(c), the proposed method had the highest traffic volume of ego vehicle, average ego-speed and average on-ramp speed for all on-ramp demand conditions. Of all the on-ramp demand conditions, the condition of 210 vehicles/h generated the smoothest traffic flow, and the proposed method had 942 vehicles, 51.6 km/h,

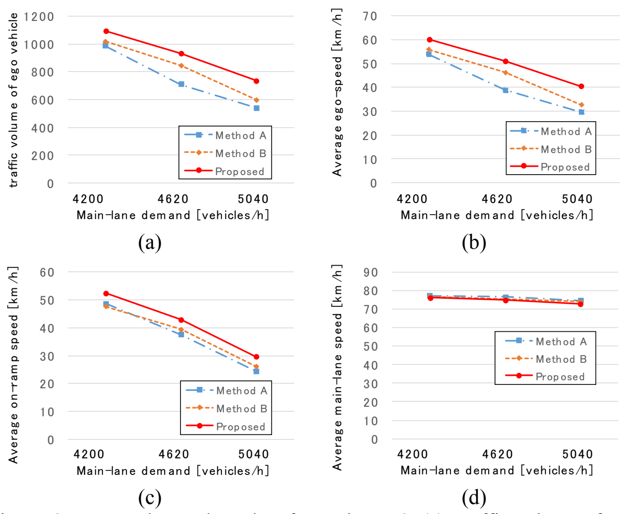


Figure 6. Experimental results of experiment 2; (a) Traffic volume of ego vehicle, (b) Average ego-speed, (c) Average on-ramp speed, (d) Average main-lane speed

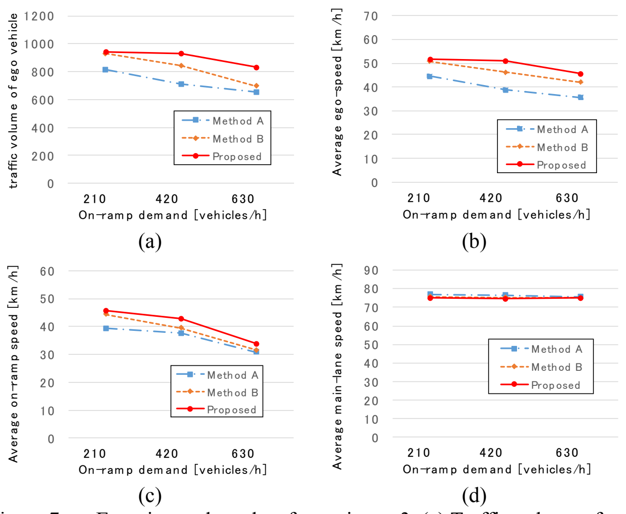


Figure 7. Experimental results of experiment 3; (a) Traffic volume of ego vehicle, (b) Average ego-speed, (c) Average on-ramp speed, (d) Average main-lane speed

and 45.6 km/h for traffic volume of ego vehicle, average ego-speed and average on-ramp speed respectively. This may be due to the fact that the smaller the on-ramp demand, the less the on-ramp vehicles are blocked from merging by other on-ramp vehicles. On the other hand, average main-lane speed had almost the same value in all methods and all conditions.

Thus, the results of experiment 2 and experiment 3 confirm that the proposed method can improve the merging efficiency without adversely affecting to the traffic flow even if the demand of main-lane or on-ramp is different from the training condition.

V. Conclusion

In this study, to realize the merging behavior considering the impact on traffic flow for controlled vehicles, we have proposed the vehicle merging controller based on the deep RL with an embedding network named Deep Merging.

The main contribution of this study is that the controlled vehicle can effectively merge onto the main-lane while minimizing the traffic impact on the main-lane and on-ramp. Moreover, the learning efficiency of deep RL is boosted by introducing an embedding network to estimate dynamic traffic conditions, such as a controlled vehicle speed.

One of the future works is to improve the robustness of Deep Merging for roads with different shapes, traffic demands and maximum speeds. In addition, it is also important to extend Deep Merging to multi-agent control to further improve traffic flow. To realize these, it is effective to change the network structure and training framework.

References

[1] M. Treiber and A. Kesting, "Evidence of convective instability in congested traffic flow: A systematic empirical and theoretical investigation," Transp. Res. Part B: Methodological, Vol. 45, No. 9, pp. 1362–1377, 2011.

[2] R. Scarinci, and B. Heydecker, "Control concepts for facilitating motorway On-ramp merging using intelligent vehicles," Transport Reviews, Vol. 34, No. 6, pp. 775–797, 2017.

[3] D. Marinescu, J. Curn, M. Bouroche and V. Cahill, "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach," Intell. Transp. Syst. Conf. (ITSC), pp. 900–906, 2012.

[4] J. Wei, J. M. Dolan and B. Litkouhi, "Autonomous vehicle social behavior for highway entrance ramp management," Intell. Veh. Symp. (IV), pp. 201–207, 2013.

[5] I. Sobh et al., "Exploring applications of deep reinforcement learning for real-world autonomous driving systems," Under review for the Int. Conf. on Comput. Vision Theory and Appl. (VISAPP), 2019.

[6] P. Wang and C. Chan, "Autonomous ramp merge maneuver based on reinforcement learning with continuous action space," Intell. Transp. Syst. Conf. (ITSC), pp. 1-6, 2017.

[7] M. Schutera, N. Goby, D. Neumann and M. Reischl, "Transfer learning versus multi-agent learning regarding distributed decision-making in highway traffic," Agents in Traffic and Transp. (ATT), Vol. 2129, pp. 57-62, 2018.

[8] M. Jaderberg et al., "Reinforcement learning with unsupervised auxiliary tasks," Int. Conf. on Learn. Representations (ICLR), pp1-14, 2017.

[9] P. Mirowski et al., "Learning to navigate in cities without a map," Advances in NIPS, pp. 2419-2430, 2018.

[10] E. Vinitsky et al., "Benchmarks for reinforcement learning inmixed-autonomy traffic," Conf. on Robot. Learn., pp. 399-409, 2018.

[11] J.Barcelo, E.Codina, J.Casas, J.L.Ferrer, and D. García, "Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems," J. Intell. Robot. Syst., Vol. 41, pp. 173-203, 2005.

[12] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, "Traffic simulation with aimsun," Fundamentals of traffic simulation, pp. 173–232, 2010.

[13] D. Silver et al., "Mastering the game of go without human knowledge," Nature, Vol. 550, No. 7676, pp. 354-359, 2017.

[14] V. Mnih et al., "Playing Atari with deep reinforcement learning," NIPS Deep Learning Workshop, pp. 1-9, 2013.

[15] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," Int. Conf. on Machine Learn. (ICML), Vol. 48, pp. 1995-2003, 2016.

[16] H. Van Hasselt, a. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," AAAI Conf. on Artif. Intell. (AAAI), pp. 2094-2100, 2016.

[17] C. Wu et al., "Framework for control and deep reinforcement learning in traffic," Intell. Transp. Syst. Conf. (ITSC), pp. 1-8, 2017.

[18] M. Abadi et al., "Tensorflow: A system for large-scale machine learning," USENIX Symp. on Oper. Syst. Des. and Implement. (OSDI), pp. 265-283, 2016.