Solving the Deadlock Problem with Deep Reinforcement Learning Using Information from Multiple Vehicles

Tsuyoshi Goto[1,*], Hidenori Itaya[1], Tsubasa Hirakawa[1], Takayoshi Yamashita[1], Hironobu Fujiyoshi[1]

Abstract— **Autonomous driving system controls a vehicle using path planning. Path planning for automated vehicles observes a vehicle and the surrounding information and plans a trajectory on the basis of rule-based approach. However, the rule-based path planning cannot generate an appropriate trajectory for complex scenes, such as two vehicles passes each other at an intersection without traffic lights. Such complex scene is called deadlock. For avoiding the deadlock, it is very costly to create rules manually. In this paper, we propose a multi-agent deep reinforcement learning method to generate appropriate trajectories at the deadlock scenes. The proposed method consists of a single feature extractor and actor-critic branches. Moreover, we introduce a mask-attention mechanism for visual explanation. By taking a look at the obtained attention maps, we can confirm the obtained agent and the reason of the behavior. For evaluating our method, we develop a simulator environment of autonomous driving that produces a certain deadlock scene. The experimental results with the developed environment show that the proposed method can generate trajectories avoiding deadlocks.**

I. INTRODUCTION

Autonomous driving system determines a vehicle trajectory using path planning. Path planning for automated vehicles observes a vehicle and the surrounding information and plans a trajectory on the basis of rule-based approach [1]. However, the rule-based path planning cannot generate an appropriate trajectory for complex scenes, such as two vehicles passes each other at an intersection without traffic lights. Such complex scene is called deadlock. For avoiding the deadlock, it is very costly to create rules manually. By solving this problem, we can reduce the cost of manual labor for developing rules.

One approach to acquire deadlock-avoiding paths is a deep reinforcement learning. In the deep reinforcement learning framework, we train an agent to find a policy that maximizes its value through trial and error. Unlike supervised learning framework, the agent is not given correct labels in advance, but rather finds the optimal action on the basis of whether its actions are judged to be good or bad behaviors. Moreover, deep reinforcement learning, which combines reinforcement learning with deep learning, has become a mainstream method [2]. Deep reinforcement learning has been used in a variety of tasks, such as robot control [3], Go [4], and playing a video-game [5], where it is difficult to create training data. Also, the deep reinforcement learning is applied for autonomous vehicle control [6], [7].

Among reinforcement learning, multi-agent reinforcement learning have been proposed. The multi-agent reinforcement learning deals with multiple agents that interacts in a single environment. Each agent finds the optimal actions considering interaction with the other agent actions. By training multiple agents simultaneously, we can acquire optimal policies for group behaviors.

In this paper, we introduce multi-agent deep reinforcement learning method for deadlock problem on an autonomous driving task. The proposed method consists of a single feature extractor across all agents and multiple branches for each agent. The feature extractor extracts common features from the states of each agent. Each branch consists of actor-critic, that is, an agent outputs value and policy. By using the branch structure, we can obtain different policies and avoid a deadlock problem. Meanwhile, the behaviors of the multi-agent deep reinforcement learning agents are more complex and more difficult to understand than that of a single-agent reinforcement learning. For analysing the trained agent behaviors, we further introduce a mask-attention mechanism [8] that visualizes the focused region of an input. For evaluating the proposed method, we develop an environment of autonomous driving that produces deadlock scenes. The experimental results with the developed environment show that the proposed method can generate trajectories avoiding deadlocks. And, we analyse the agent behavior using attention maps.

II. RELATED WORK

A. Deep reinforcement learning and multi-agent reinforcement learning

Over the last decade, many deep reinforcement learning methods have been proposed [2], [9], [10], [11], [12]. Among them, asynchronous advantage actor-critic (A3C) [9] is one of the most popular method. A3C is based on the actor-critic method [13] that combines a speedup by building multiple environments and collecting experience in parallel, asynchronous parameter updates for each model, and a method for considering rewards a few steps in advance. The mechanism that collects experiences in parallel is called a worker; each worker has its own local network and all workers share a global network. A3C reduced training time and achieved higher performance on Atari2600 benchmark.

Multi-agent reinforcement learning acquires agent policies of a group by training multiple agents in an environment simultaneously [14], [15], [16], [17], [18]. The multi-agent reinforcement learning can be categorized into two approaches. The first approach is introducing a mechanism to communicate with each agent. Each agent can find the optimal action considering the other agents by sending their internal states via the communication mechanism. Sainbayar et al. [14] introduced a communication mechanism. The introduced mechanism takes states and actions of every agents as vectors and compute average of these vectors. Then, the averaged vector is fed into every agents. This mechanism enables all agents to understand the states of all agents and to learn optimal behavior considering the other agents in a multi-agent environment. Jiechuan et al. [15] uses a graph convolution to take into consideration the information of other agents. The convolution is performed considering the importance of other agents only under certain conditions. This method consider the information excluding own agent, which facilitates the acquisition of cooperative actions.

The second approach is designing rewards considering the entire learning process of multi-agent reinforcement learning apart from the environmental rewards. This additional rewards can train agents while taking other agents into consideration. Natarajan et al. [16] used inverse reinforcement learning to find reward design taking interaction between agents in an environment into consideration. They achieved the collision avoidance between agents using the obtained reward. Natasha et al. [17] introduced a novel reward called social influence that represents the influence of an action for the other agents and three specific models. The first model is basic influence. The basic influence trains two agents: influencer that learns behaviors to affect others by social influence and influencee that learns behavior from environmental reward. The second model is influential communication, which introduces a communication mechanism and learns behaviors by social influence. The third model is modeling other agents that introduce a mechanism for predicting other agent's action and learn the prediction mechanism by social influence. Their method achieved the emergence of cooperative actions.

Our method is categorized into the first approach, i.e., sharing information between each agent. Unlike the above mentioned approach, the proposed method predicts every agent actions in a single network. The feature extractor extracts common features across all agents, which enables to consider the states of other agents.

B. Visual explanation in reinforcement learning

For understanding and analysing the decision-making of a reinforcement learning agent, visual explanation methods have been developed [8], [18]. Visual explanation generates an attention map that highlights region where an agent focused in taking an action. Itaya et al. [8] proposed mask-attention A3C (Mask A3C) that introduces an attention mechanism to asynchronous advantage actor-critic (A3C) [2]. A mask-attention mechanism is incorporated in the output for policy and in the output of state value in A3C. This mechanism masks the feature map of the middle layer by using the attention map. By using the mask-attention mechanism, the attention map can be obtained during the prediction step.

In this paper, we introduce the mask-attention mechanism. By visualizing attention maps, we further analyse the obtained agent behaviors in addition to quantitative evaluations.

III. PROPOSED METHOD

In autonomous driving, deadlock problem occurs because different agents seeks their own rewards. When multiple independent network are given for each agent, there is no mechanism to influence the learning of other agents and each agent takes an action only considering their own state. Consequently, taking actions avoiding the deadlock problem is delayed.

To solve the deadlock problem, we propose a multi-agent reinforcement learning method that learns every agents' policy simultaneously in a single network. Figure 1 shows the network structure of the proposed method. The proposed method consists of a feature extractor and multiple branch structure. The feature extractor extracts common feature from the input states of every agents. Then, extracted feature is fed into multiple branches. Each branch is corresponding to each agent and outputs the agent's action. Moreover, we introduce a mask-attention mechanism for the actor-critic branches, which enables us to understand the reason of the agent's decision-making. Hereafter, we introduce the details of the proposed method.

A. Feature extractor

As shown in Fig. 1, the proposed method first extracts a common feature for every agents. Here, given n agents in an environment, we denote agents as $i \in \{1, \ldots, n\}$ and the state and action of an agent at time step t as s_t^i and a_t^i, respectively. The proposed method concatenates the states of every agents as $S_t = [s_t^1, \ldots, s_t^n]$. Then, we extract the common feature x_t from S_t as $x_t = f(S_t; \theta_f)$, where θ_f is the parameter of the feature extractor.

The common feature x_t represents states of every agents. This enables each agent consider every states of the other agents.

B. Actor-Critic Branch

The common feature x_t is used for multi-branch structure. Each branch consists of actor-critic network, which is assigned for each agent. The actor-critic branch is composed of policy and state value outputs. Moreover, we add a mask-attention mechanism (details are described in Sec. III-C). Unlike the feature extractor, the actor-critic branch is independent and outputs the actions of the assigned agent.

The policy output in an actor-critic branch takes the common feature x_t as an input and predicts a policy. The output is a probability distribution, and the policy branch selects an action in the current state according to the probability distribution. We define the policy of an agent i as $\pi(a_t^i | x_t; \theta_p^i)$, where θ_p^i is the parameter for policy output. The value output also takes x_t as an input and predicts a state value. The output is the expected value of the reward in a state, and it represents the value of being in a certain state. We define the state value of an agent i as $V(x_t^i; \theta_v^i)$, where θ_v^i is the parameter for value output. By using the common feature x_t as input for actor-critic, each agent can take an action to avoid deadlock considering the other agents' behaviors.

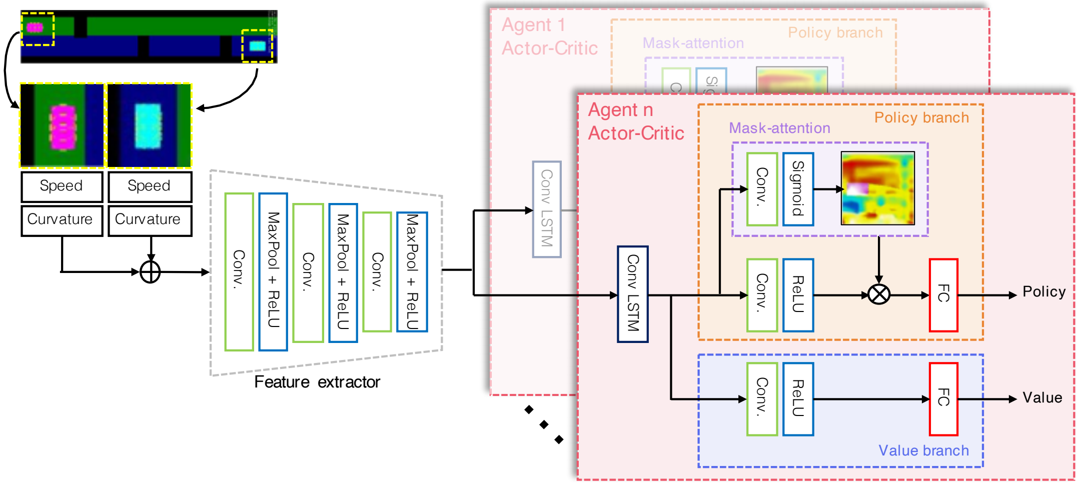


Fig. 1. The proposed network structure.

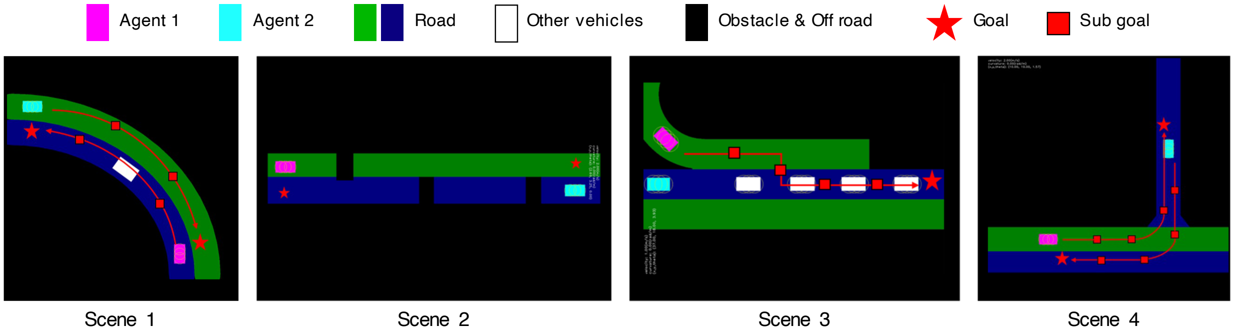


Fig. 2. Examples of the developed environment (scenes).

For training the proposed network, we define the loss functions for policy and value outputs, respectively. The loss function for policy output of i-th agent L_p^i is defined by

$$L_p^i = -\log\left(\pi(a|s;\theta_p^i)\right) A(s,a) - \beta H(\pi), \tag{1}$$

where $A(s,a)$ is the advantage function that considers a few steps in advance. Here, we introduce an entropy $H(\pi)$ as an regularization term, which helps to avoid falling into local minima. β is a scale parameter for $H(\pi)$. The loss function for value output L_v^i is defined by

$$L_v^i = \left(r + \gamma V(x_{t+1}^i;\theta_v^i) - V(x_t^i;\theta_v^i)\right)^2. \tag{2}$$

C. Mask-attention mechanism

The mask-attention mechanism is a method for visual explanation of deep reinforcement learning. It generates an attention map and use and use it for masking feature map in an intermediate feature map. By learning with this mechanism, the attention map can be acquired during the prediction step.

In this study, we introduce the mask-attention mechanism for policy output to provide a visual explanation of agent's action as shown in Fig. 1. Mask-attention is obtained from the feature map by using 1×1 convolution and sigmoid function. The obtained attention map is then used for attention mechanism with feature map $F(x_t^i)$ as

$$F'(x_t^i) = F(x_t^i) \cdot M(x_t^i), \tag{3}$$

where $M(x_t^i)$ is the mask-attention. By visualizing the attention map, we can confirm the attention area of the policy.

TABLE I
REWARD RULES

	Reward	
	Scenes 1, 3, 4	Scene 2
Goal	+10	+10
Collision	-10	-10
Following distance penalty	-3	-3
Sub goal	+2	
Lane departure	-10	
Forward		+0.5
Keep left lane		+1.5
Speed penalty		-0.5

IV. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed method. In our experiment, we developed autonomous driving environments in which deadlocks happen. In this paper, we call these environments as *scenes*. We tried to determine whether the proposed method could acquire avoidance actions against deadlocks that happen in each scene.

TABLE II

EPISODE REWARDS OBTAINED FOR EACH SCENE (AVERAGE OVER 100 EPISODES)

	Scene 1		Scene 2		Scene 3		Scene 4	
	Agent 1	Agent 2	Agent 1	Agent 2	Agent 1	Agent 2	Agent 1	Agent 2
A3C [2]	324.7	275.3	76.1	84.8	195.0	140.8	134.8	137.1
Ours	**961.5**	**580.2**	**997.0**	**994.4**	**785.0**	**585.3**	**773.4**	**589.6**

A. Environment

We prepared four scenes in which deadlocks happen. Figure 2 shows the developed scenes and Table I shows the reward design for each scenes. All environments are developed so that the agent drives correctly in the driving lane and that an episode fails if it collides with other vehicles. The maximum speed was 3 m/s, the minimum speed was -1 m/s, the maximum curvature was 0.25, and the minimum curvature was -0.25. In the following, we describe the details of each scene.

1) Scene 1: Scene 1 is a single lane road with bad visibility and a parked vehicle on the road. Two vehicles are placed on a one-lane curved road, and the scene simulates an environment with illegally parked vehicles.

2) Scene 2: Scene 2 contains an obstacle in the lane and the driver has to avoid the oncoming vehicle. Two vehicles are placed to create an environment with obstacles. The obstacles are placed randomly at intervals that do not allow more than two vehicles to pass.

3) Scene 3: Scene 3 is one of merging onto a main road. Multiple vehicles are placed in the scene, and one of the vehicles is merging. One vehicle on the merging side and one vehicle on the merging side are controlled. The other vehicles run at a constant speed.

4) Scene 4: Scene 4 contains a narrow exit and entrance. Two vehicles are placed on a narrow road, wide enough for no more than two vehicles to pass each other.

B. Training settings

The numbers of training times are 2.0×10^7 steps for scene 1, 2.0×10^7 steps for scene 2, 5.0×10^7 steps for scene 3, and 2.0×10^6 steps for scene 4. In all environments, the episode ended when the agent collided with an off-road object, an obstacle, or other vehicle, or when 1.0×10^5 steps had elapsed.

The proposed method was evaluated in the following two aspects:

- Comparison by episode reward, and
- Visualization of action and attention maps of trained models.

In the comparison with episodic rewards, we compare the proposed method with the A3C reinforcement learning method.

C. Comparison by episode reward

Table II shows the average cumulative reward for each scene with 100 episodes of evaluation. We can see that the agent using the proposed method obtained the highest reward in all scenes and there is a difference of more than 300 points compared with A3C. This results show that the proposed method successfully increases the reward by avoiding the deadlock. On the other hand, A3C could not avoid the deadlock situation and fell into a local minima, which results in a lower reward. These results confirm the effectiveness of the proposed method.

D. Visualization of attention map for policy

We show attention maps obtained from the proposed method in each scene. In the following results, the red vehicle is agent 1, and the blue vehicle is agent 2.

1) Scene 1: Figure 3 shows the actions of the agents in scene 1. From Fig. 3, agent 2 waited from t+1 to t+3 until agent 1 passed by and then started moving. This shows that agent 2 can acquire the action of avoiding deadlock by considering the other agent, which is to wait for agent 1 to pass by. Figure 4 shows attention maps of each agent at t+3. We can confirm that agent 1 was looking at the road around and ahead, while agent 2 was looking at itself and the other vehicle. This is probably because agent 1 learned to be an agent that does not care about the other agents and goes on the road ahead, while agent 2 learned to be an agent that looks at the other agents as well as itself and drives safely. These results confirm that the proposed method can be used to acquire an agent that avoids collisions by deadlock in consideration of the opponent.

2) Scene 2: Figure 5 shows the actions of the agents in Scene 2. Here, agent 1 stopped from t+1 to t+3, and agent 2 started moving after passing the obstacle. Agent 1 acquired the action of waiting for agent 2 to pass by to avoid the deadlock. Figure 6 shows attention maps of each agent at t+2. We can confirm that agent 1 was looking at the other vehicle and the obstacle, while agent 2 was looking at the road ahead and not at the other vehicle. This is because agent 1 learned to look at the opponent and act in accordance with the other agent, while agent 2 learned to go on its way without caring about the other agent. These results show that the proposed method can be used to acquire an agent that avoids collisions by deadlock in consideration of the opponent.

3) Scene 3: Figure 7 shows the actions of the agents in scene 3. Here, agent 2 moved to a different lane between t+1 and t+3, and agent 1 was able to merge into the empty lane. Agent 2 acquired the action of giving way for agent 1 to merge in order to avoid deadlock. Figure 8 shows attention maps of each agent at t+1. We can confirm that agent 1 was looking at the other vehicle and the vehicle ahead, while agent 2 was looking at the space between the other vehicle and the vehicle ahead. This is because agent 1 learned to look at other vehicles and tried to merge safely, and agent 2

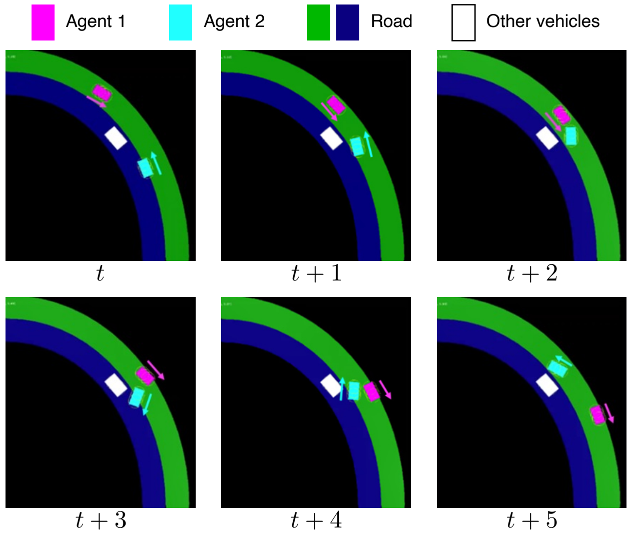


Fig. 3. Example of agents' trajectories in scene 1. The arrow shows the direction of movement of the vehicle.

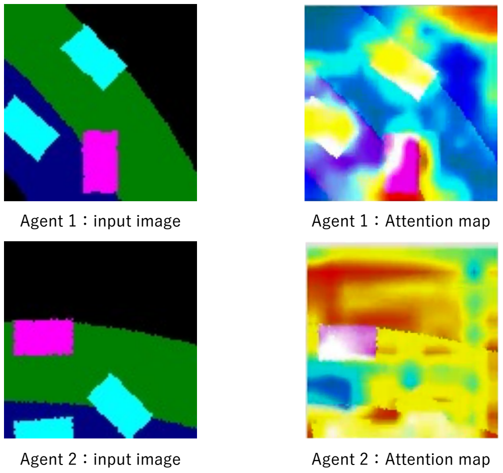


Fig. 4. Attention maps at deadlock in scene 1. Left side is the input image and the right side is the corresponding attention map.

learned to look at the distance from the merging vehicle and give way. These results confirm that the proposed method can be used to acquire an agent that avoids collisions by deadlock in consideration of the opponent.

4) Scene 4: Figure 9 shows the actions of the agents in scene 4. Here, agent 1 used backing and stopping from t+1 to t+3 to wait for agent 2 to come out of the narrow path, and then started moving after agent 2 passed by. Agent 1 acquired the action of waiting for agent 2 to pass by in order to avoid deadlock. Figure 10 shows attention maps of each agent at t+2. We can confirm that agent 1 was looking at the other vehicle and agent 2 was looking at the end of the exiting road. This is because agent 1 learned to watch the other agent and act in accordance with the other agent, while agent 2 learned to go on its way without caring about the other agent. These results confirm that the proposed method can be used to acquire an agent that avoids collisions by deadlock in consideration of the opponent.

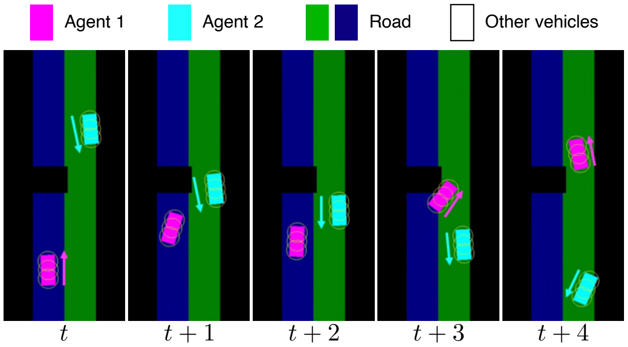


Fig. 5. Example of agents' trajectories in scene 2. The arrow shows the direction of movement of the vehicle.

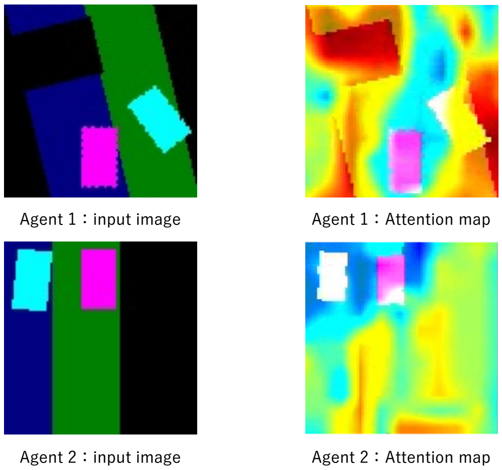


Fig. 6. Attention maps at deadlock in scene 2. Left side is the input image and the right side is the corresponding attention map.

V. CONCLUSIONS

In this paper, we proposed a multi-agent reinforcement learning method to avoid deadlocks by considering the view of other agents. The proposed method extracts a common feature from every agents' states and use multiple branches for taking actions for each agent. Our experimental results with the developed environments show that our method enabled agents to avoid deadlocks more easily than agents incorporating normal reinforcement learning. In addition, the visualization of the actions and attention maps of the trained model in an autonomous driving environment showed that the model acquired actions to avoid deadlocks in consideration of the opponent. These results indicate that the proposed method can be used to automatically obtain an optimal path in a deadlock situation. Our future work will include automatic acquisition of optimal paths in more varied environments and the implementation of a model that takes the other person into consideration and performs cooperative actions.

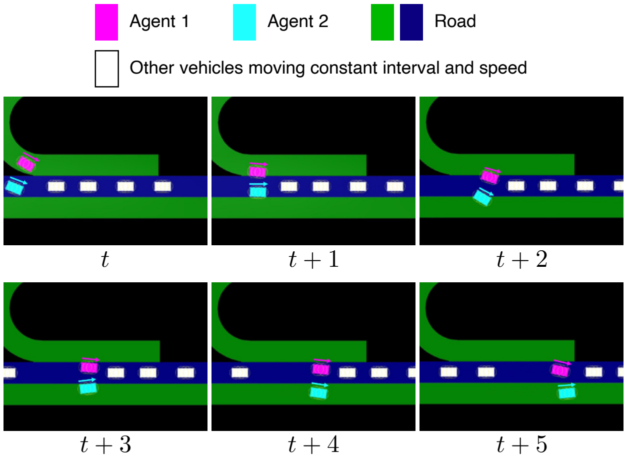
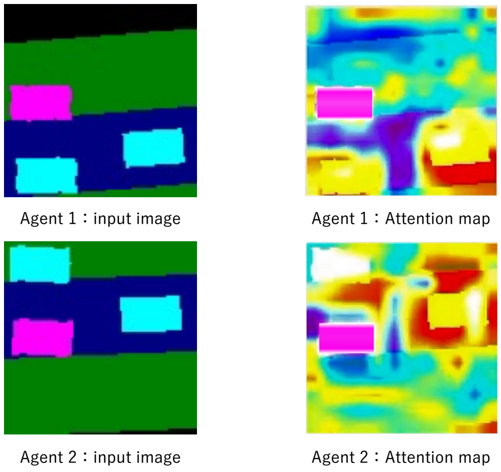


Fig. 7. Example of agents' trajectories in scene 3. The arrow shows the direction of movement of the vehicle.


Agent 1 : input image

Agent 1 : Attention map

Agent 2 : input image

Agent 2 : Attention map

Fig. 8. Attention maps at deadlock in scene3. Left side is the input image and the right side is the corresponding attention map.

Acknowledgement

Research and development for recognition technologies and other technologies necessary for automated driving technologies(levels 3 and 4).

References

[1] Y. Keisuke, I. Toshiki, and S. Naoki, "Trajectory Optimization and State Transition for Urban Automated Driving," in *The 2nd International Symposium on Swarm Behavior and Bio-Inspired Robotics*, 2017, pp. 169–172.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3389–3396.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

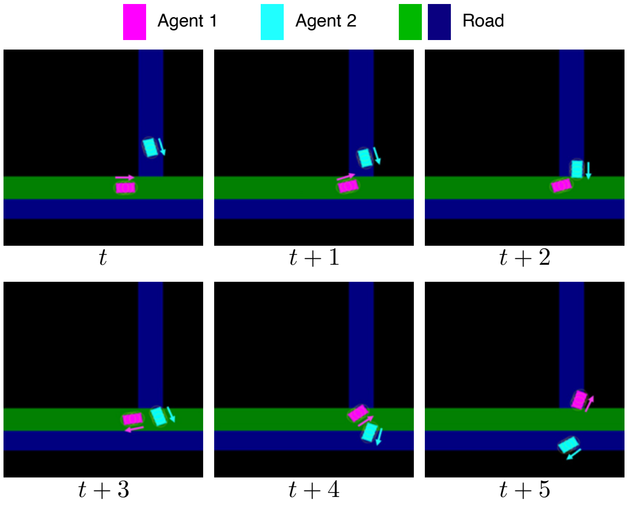
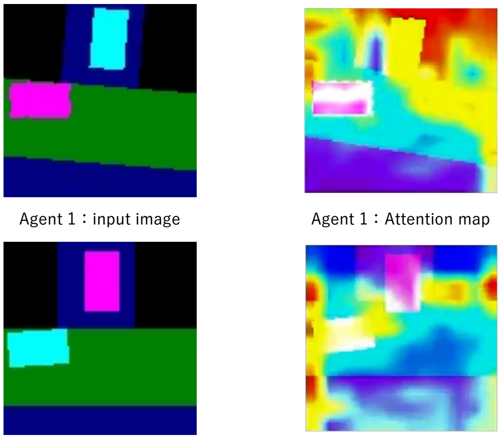


Fig. 9. Example of agents' trajectories in scene 4. The arrow shows the direction of movement of the vehicle.


Agent 1 : input image

Agent 1 : Attention map

Agent 2 : input image

Agent 2 : Attention map

Fig. 10. Attention maps at deadlock in scene4. Left side is the input image and the right side is the corresponding attention map.

[5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[6] A. Folkers, M. Rick, and C. Büskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2025–2031.

[7] M. Buechel and A. Knoll, "Deep reinforcement learning for predictive longitudinal control of automated vehicles," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2391–2397.

[8] H. Itaya, T. Hirakawa, T. Yamashita, H. Fujiyoshi, and K. Sugiura, "Visual Explanation using Attention Mechanism in Actor-Critic-based Deep Reinforcement Learning," in *The International Joint Conference on Neural Networks*, 2021.

[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[10] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint*

arXiv:1707.06347, 2017.

[12] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning,” in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[13] V. R. Konda and J. N. Tsitsiklis, “Actor-Critic Algorithms,” in *Proceeding of neural information processing systems*, 2000, pp. 1008–1014.

[14] S. Sukhbaatar, R. Fergus, *et al.*, “Learning multiagent communication with backpropagation,” *Advances in neural information processing systems*, vol. 29, pp. 2244–2252, 2016.

[15] J. Jiang, C. Dun, T. Huang, and Z. Lu, “Graph convolutional reinforcement learning,” *arXiv preprint arXiv:1810.09202*, 2018.

[16] S. Natarajan, G. Kunapuli, K. Judah, P. Tadepalli, K. Kersting, and J. Shavlik, “Multi-agent inverse reinforcement learning,” in *2010 Ninth International Conference on Machine Learning and Applications*. IEEE, 2010, pp. 395–400.

[17] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. De Freitas, “Social influence as intrinsic motivation for multi-agent deep reinforcement learning,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 3040–3049.

[18] I. Sorokin, A. Seleznev, M. Pavlov, A. Fedorov, and A. Ignateva, “Deep attention recurrent q-network,” *arXiv preprint arXiv:1512.01693*, 2015.