Proceedings of the 2017 IEEE
International Conference on Robotics and Biomimetics
December 5-8, 2017, Macau SAR, China

Autonomous Overtaking Decision Making of Driverless Bus Based on Deep Q-learning Method

Lingli Yu, Xuanya Shao, Xiaoxin Yan
School of Information Science and Engineering,
Central South University,
Changsha, China
llyu@csu.edu.cn

***Abstract*—The autonomous overtaking maneuver is a valuable technology in unmanned vehicle field. However, overtaking is always perplexed by its security and time cost. Now, an autonomous overtaking decision making method based on deep Q-learning network is proposed in this paper, which employs a deep neural network(DNN) to learn Q function from action chosen to state transition. Based on the trained DNN, appropriate action is adopted in different environments for higher reward state. A series of experiments are performed to verify the effectiveness and robustness of our proposed approach for overtaking decision making based on deep Q-learning method. The results support that our approach achieves better security and lower time cost compared with traditional reinforcement learning methods.**

***Keywords*—deep learning; reinforcement learning; deep Q network; overtaking decision making**

I. Introduction

The driverless bus overtaking decision making is a complex and dynamic system. Not only the vehicle kinematics but the dynamic environment should be considered[1].

Currently, the major two difficulties are model building and time uncertainty of overtaking.

To deal with difficulty of modeling, Prakash Shashikala[2] builds a model with finite state machine (FSM) to solve active vibration control of a full scale aircraft wing. There are still some problems exist in FSM. As in [3], Kurt Arda use EFSM to deal with the model problem. The model by EFSM is more efficient. As to more complex situation, the FSM and EFSM states grow exponentially, so it is very difficult to build the model.

Another problem is that the time of overtaking is difficult to determine, because there are a variety of situations in the real life. In many cases, the autonomous vehicles using finite state machine may be too cautious, causing unnecessary delays[3]. As pointed out in [4] and [5], reinforcement learning is adopted for decision making of autonomous vehicles, in which the decision process is based on Markov. Reference [6] proposes that model based on the partially observable Markov decision process(POMDP) is adopted to achieve decision making.

There are also some other approaches to be used in autonomous vehicles decision making. As in [8], multiple criteria decision making is applied to divide the whole decision into two stages. Decision making method of fuzzy neural network is proposed as in [9].

In this paper, we focus on Q-learning for driverless bus overtaking decision making. The complex dynamic environment is implicitly modeled by Q-learning. However, the major difficulty of Q-learning for overtaking decision making of driverless bus lies in exponentially expanding complexity of states with the number of the vehicles and lanes[10-12]. Deep learning has achieved great success in pixel-level image processing[13-17]. Recently, Deep Q Network (DQN) is proposed to solve the modeling problems of complex systems, which is a new approach that combines deep learning and reinforcement learning[18-21]. In this paper, we study the effectiveness and feasibility of driverless bus overtaking decision making using DQN. Experiments show that DQN is a more convenient and powerful method compared with traditional reinforcement learning methods.

The remainder of this paper is organized as follows. In order to detail our new design, firstly, we present the new overtaking decision making approach using DQN in Section Ⅱ. Then, a series of simulation experiments are conducted to demonstrate the effectiveness of the proposed approach in Section Ⅲ, which is further proved in the actual bus experiments in Section Ⅳ. At last, we conclude the paper in Section Ⅴ.

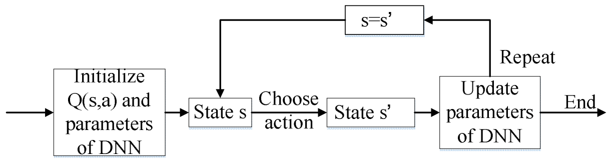
II. Problem Presentation and Solution

A. DQN Framework Introduction

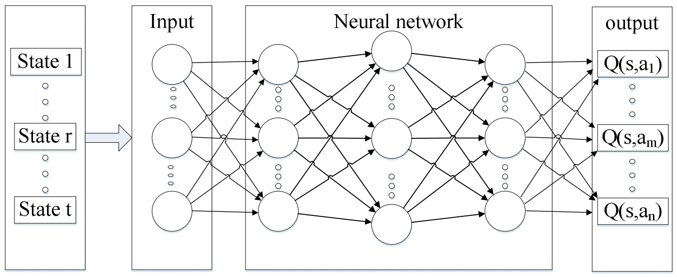
DQN in driverless bus decision making can be divided into two stages. The first stage is training. A lot of episodes are applied to explore the environment, meanwhile, the neural network is trained by the rewards from environment. Secondly, driverless bus chooses action by DQN in test stage.

The driverless bus decision making based on DQN is shown in Fig.1. The process of training DQN is shown in Fig.1(a), meanwhile, Fig.1(b) is structure of DNN. DNN's inputs are current states of driverless bus and the other vehicle's states. DNN's outputs are Q values of every actions the driverless bus can take. The driverless bus obtains rewards from environment. The weight of DNN is updated by gradient descent during training stage. After being trained, the driverless bus optimal action is selected according to the DQN model.

This paper was supported by National Natural Science Foundation of China (Grant No.61403426, 61403427). State Key Laboratory of Mechanical Transmissions of Chongqing University(SKLMT-KFKT-201602), State Key Laboratory of Robotics and System (HIT) (SKLRS-2017-KF-13).



(a) The process of training DQN



(b) Structure of DNN

Fig.1. DQN in driverless bus decision making

B. Reinforcement Learning

Q-learning[22] is a one-step learning algorithm. It establishes a matrix Q to store the value data between the state and the action. *S* contains the state of agent. The state at time *t* represents as $s_t\left(s_t \in S\right)$. And the action *A* contains the actions that the agent can take. The a_t means action the agent takes at s_t.

At station s_t, the agent chooses an action a_t by $\varepsilon-greedy$ strategy. After taking the action a_t, the agent moves to s_{t+1}. At the same time, the agent receives a reward value *r* after taking a_t from s_t to s_{t+1}, and the *Q*(*s,a*) updates according to the equation[23] (1).

$$Q\ (\mathrm{s,a}) = Q(\mathrm{s,a}) + \alpha[r + \gamma \max Q(\mathrm{s',a'}) - Q(\mathrm{s,a})] \quad (1)$$

where α is discount factor. $\gamma \in [0,1]$ is learning parameter. *Q*(*s*,*a*) is the value from s_{t+1}.

The $\varepsilon-greedy$ strategy is shown as equation (2). "a_t" is the action that the agent takes at time t. "$random$" is a random number ($random \in [0,1]$). ε is a threshold ($\varepsilon \in [0,1]$).

$$a_t = \begin{cases} random \text{ action } from\ A_t & ,if\ \ random < \varepsilon \\ \arg\max Q(s,a) & ,others \end{cases} \quad (2)$$

The state $\left(s_t, a_t, s_{t+1}, Q(s,a)\right)$ is stored in a matrix to train the Q matrix. At s_{t+1}, we take the same procedure in time *t*. After being trained by many episodes, the Q matrix stabilized at a value. And then we use the Q matrix to play games or make decision in autopilot.

The Q-learning is shown as TABLE Ⅰ.

TABLE Ⅰ. The Q-learning algorithm

```
Algorithm Ⅰ: The Q-learning algorithm
For train-flag is true then:
    Initialize Q (s, a) arbitrarily
    Repeat (for each episode)
        Initialize s
        Repeat (for each step of episode):
            Choose a from s using epsilon-greedy-policy from Q
            Take action a, get r, s'
            $Q\ (\mathrm{s,a}) = Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$
            s = s'
        until s is terminal
```

C. The Deep Q Network(DQN)

In the previous method, the Q function is usually implemented using a matrix Q. But in the overtaking problem, it is difficult using the table-based Q-learning method to solve the problem in a limited time because the state space is so huge. To solve the problem, we use the deep stacked autoencoders (SAE) neural network[24] to approximate the Q function. Taking the states as inputs, and the Q values for every actions as outputs.

Autoencoders are the basis for constructing the deep SAE neural networks. The autoencoder is a neural network that reproduces the input signal as much as possible. As shown in Fig.2, the autoencoder has three layers including input layer, hidden layer and output layer.

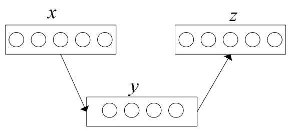


Fig.2. An autoencoder

Given the input *x*, we first encode *x* as *y*(*x*) using equation (3), and then decode *y*(*x*) into *z*(*x*) using equation (4). The purpose of the autoencoder is to make *z*(*x*) = *x*.

$$y(x) = f(Wx + b) \quad (3)$$

$$z(x) = f'\left(W'x + b'\right) \quad (4)$$

where W and W' are weight matrices, b and b' are bias vectors.

The activation function is a Sigmoid function for each node of the hidden layers. The average activation value of the unit *j* in the hidden layer for multiple samples is expressed as (5).

$$\hat{p}_j = \frac{1}{N}\sum_{i=1}^{N} a_j\left(x^{(i)}\right) \quad (5)$$

where N is the number of samples, a_j is the activation function, and $x^{(i)}$ is the *i*-th group of samples.

We add a constraint $\hat{p}_j$ for each node's average activation value, and $\hat{p}_j \approx p_0$, where p_0 is called sparsity parameter. The Kullback - Leibler Divergence is added to the objective function, which is used to punish p_0 to deviate from $\hat{p}_j$. The expression is (6).

$$KL\left(p_0 \,||\, \hat{p}_j\right) = p_0 \, log \frac{p_0}{\hat{p}_j} + (1 - p_0) \, log \frac{1 - p_0}{1 - \hat{p}_j} \quad (6)$$

After adding the penalty function, objective function is (7).

$$arg\ min \frac{1}{2}\sum || x - z(x) ||^2 + \beta \sum_{j=1}^{n} KL\left(p_0 \,||\, \hat{p}_j\right) \quad (7)$$

where β is the weight of the sparse penalty term, *n* is the number of the hidden units.

We use greedy hierarchical methods to train each layer in turn so that we can stack the automatic encoders in a greedy hierarchical method to pre-train the weight of the deep network. The Q function obtained is written as $Q(s_t,a_t,\theta)$, where θ is the values of SAE network.

During Q-learning, the deep SAE network is trained by minimizing the loss function over samples as (8).

$$L(\theta)=\left[\underbrace{Q(s,a,\theta')+\alpha\left(r+\gamma Q(s',a',\theta')-Q(s,a,\theta')\right)}_{target}-\underbrace{Q(s,a,\theta)}_{prediction}\right]^2 \quad (8)$$

where a record $(s_t',a_t',r_t',s_{t+1}')$ is drawn randomly from the replay memory.

D. Description of DQN State, Action and Reward Function

1) Definition of DQN State Space

In this study, the DQN state space is described as $S=\{L,D_1,D_2,V,V_1,V_2\}$. The specific meaning of each letter is as Fig.3.

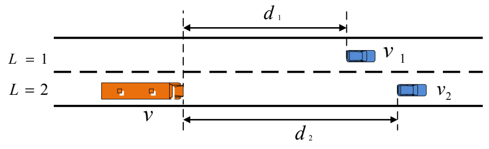


Fig.3. Definition of the state space

L indicates the lane where the driverless bus is located. We use the two-lane road so $L\in\{1,2\}$. The left lane is numbered 1,and the right lane is numbered 2.

We assume that d_{max} is the maximum distance the driverless bus can detect. The two parameters d_1 and d_2 are distances between the driverless bus and the nearest vehicles in the left lane and right lane. When there are no vehicles, $d_1=d_{max}$ and $d_2=d_{max}$. They are quantized into N_{d_1} and N_{d_2} according to equation (9) and (10).

$$D_1=\left\lceil \left(N_{d_1}\times d_1\right)/d_{max}\right\rceil \quad (9)$$

$$D_2=\left\lceil \left(N_{d_2}\times d_2\right)/d_{max}\right\rceil \quad (10)$$

We assume that v_{max} is the maximum speed of driverless bus and other vehicles. The speed of driverless bus is v. The speed of nearest vehicle in left lane is v_1. The speed of nearest vehicle in right lane is v_2. When there are no vehicles, $v_1=v_{max}$ and $v_2=v_{max}$. They are quantized into N_V, N_{V_1} and N_{V_2} according to equation (11), (12) and (13).

$$V=\left\lceil \left(N_V\times v\right)/v_{max}\right\rceil \quad (11)$$

$$V_1=\left\lceil \left(N_{V_1}\times v_1\right)/v_{max}\right\rceil \quad (12)$$

$$V_2=\left\lceil \left(N_{V_2}\times v_2\right)/v_{max}\right\rceil \quad (13)$$

2) Definition of DQN Action Space

In this study, the DQN action space is described as TABLE Ⅱ.

3) Definition of DQN Reward Function

The performance evaluation of the driving process mainly includes the safety, smoothness and fastness. The reward function is as (14).

$$r=\begin{cases}-500 & collision\\ 0.1V+1/V_{different}+0.1D_{different}+0.2L & no\ collision, V_{different}\neq 0\\ 0.1V+0.1D_{different}+0.2L & no\ collision, V_{different}=0\end{cases} \quad (14)$$

where $V_{different}$ represents the difference between the speed at the state s and the speed at the state s' by taking the action a, meanwhile $D_{different}$ represents the difference between the D at the state s and the D at the state s'. D is defined as (15).

$$D=\begin{cases}D_1 & L=1\\ D_2 & L=2\end{cases} \quad (15)$$

Firstly, whether a collision occurs is determined, indicating that security is the paramount. Then, V reflects fastness. Smoothness is reflected by $1/V_{different}$. Reward increases when the driverless bus accelerates, on the contrary, reward decreases when the driverless bus decelerates. $D_{different}$ is set to avoid the distance between the driverless bus and the vehicle in front of the driverless bus is too close. Considering the driverless bus should not stay in the overtaking lane for long time, the reward when the driverless bus stays in the overtaking lane should be less than the reward when the driverless bus stays in the right lane.

III. Simulation Results and Discussion

A. Simulation Environment

In this section, a series of experiments are conducted to verify the correctness of overtaking decision making using DQN. The driverless bus is 12 meters by 2.8 meters with a maximumspeed of 30 km/h. And, the size of other vehicles is 4 meters by 1.6 meters with a maximum speed of 30 km/h. Moreover, The environment of road is as follows: Using two lanes, each of which is 3.5 meters wide. The maximum perception distance of the driverless bus is allowed in 100 meters.

N_V, N_{V_1} and N_{V_2} are set to 4, 4 and 4 respectively, which are used to quantify the speed of the driverless bus and other two cars. N_{d_1} and N_{d_2} are *10* and *10* respectively, which represent the quantization distances between the driverless bus

TABLE Ⅱ. Definition of DQN action space

Number	**1**	**2**	**3**	**4**	**5**	**6**
Action	Change lane and Accelerate	Change lane and Keep speed	Change lane and Slow down	Keep lane and Accelerate	Keep lane and Keep speed	Keep lane and Slow down

and the other vehicles. So $V, V_1, V_2 \in \{0,1,2,3\}$; $L \in \{1,2\}$; $D_1, D_2 \in \{1,2,3,4,5,6,7,8,9,10\}$.

The other parameters of Q-learning are set as follows: α is set to 0.05; γ is set to 0.1; ε is set to $1/\sqrt{t}$, where t is the number of episodes experienced.

We adopt a four-layer SAE neural network with two hidden layers. We use the current state and the previous three states as inputs, therefore, there are 24 neurons in the input layer. We adopt the Q values of every possible actions as outputs, so there are 6 neurons in the output layer. Meanwhile, there are 20 and 12 neurons in the two hidden layers. The activation function is a Sigmoid function for each node of the hidden layers.

B. Simulation Results

1) Case1: Straight Road with Other Vehicles Moving at Constant Speed

In this case, the length of the road is 10km, which is 1000 after being quantified. We randomly generate 50 vehicles on the road, whose position, speed and lane are generated randomly. Meanwhile, the speed of the vehicles maintains constant. Initially, the driverless bus is at the starting point of lane 1 with speed of 3.

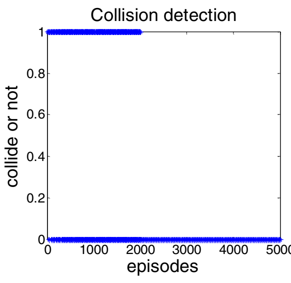


Fig.4. Collision detection

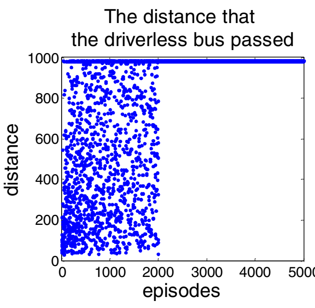


Fig.5. The distance that the driverless bus passed

Fig.4 and Fig.5 show the results of the driverless bus using DQN for overtaking decision making. DQN is trained for 2000 episodes, and the following episodes are taken to test the training results. In Fig.4, the vertical coordinate of 1 represents the collision of the driverless bus, and the vertical coordinate of 0 represents the collision of the driverless bus. Because we use the ε greedy strategy, meanwhile, the neural network has not been trained well, there are a lot of collisions within the first 2000 episodes. As shown in Fig.5, the vertical coordinate represents the distance that the driverless bus passed in each episode. The vertical coordinate of 1000 represents the driverless bus does not crash, while the vertical coordinate of less than 1000 represents the driverless bus crashes. After being trained, the driverless bus is able to pass the whole road without crash.

2) Case2: Straight Road with Other Vehicles Moving at Variable Speed

In this case, the speeds of the other vehicles are variable. Therefore, Q-learning requires more episodes for training, which is more complex compared with Case1.

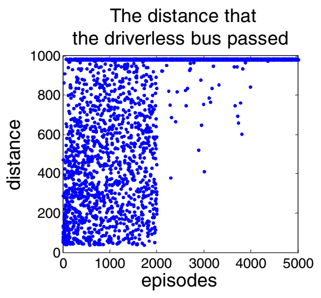


Fig.6. Training for 2000 episodes

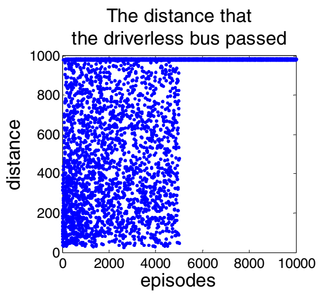
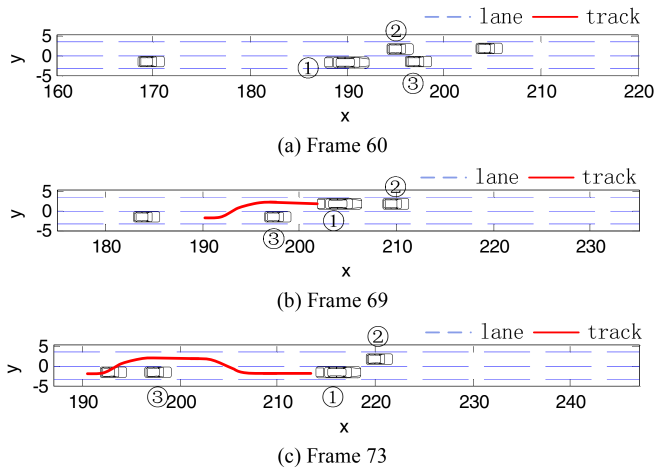


Fig.7. Training for 5000 episodes

In Fig.6 and Fig.7, the vertical coordinate represents the distance that the driverless bus passed in each episode. There are still collisions after being trained for 2000 episodes. However, the driverless bus is able to pass the whole road without crash after being trained for 5000 episodes.



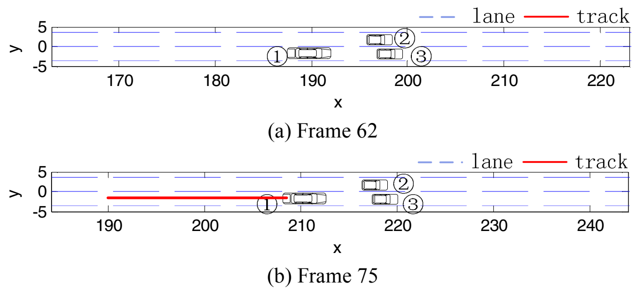
(a) Frame 60

(b) Frame 69

(c) Frame 73

Fig.8. The overtaking process

The specific overtaking process is shown in Fig.8. Vehicle ① is our driverless bus and the red lane represents the trajectory of the driverless bus passed. In the 60th frame, the speeds of the vehicle ② and the vehicle ③ are 0, therefore the driverless bus needs to wait. When the vehicle ② moves at speed 2, the driverless bus changes to the left lane and follows it. After overtaking the vehicle ③, the driverless bus returns to the right lane and accelerates to a speed of 3.



(a) Frame 62

(b) Frame 75

Fig.9. The following process

The following process is shown in Fig.9. There are two vehicles of the same speeds in two lanes. The driverless bus follows them in the right lane until there is an opportunity to overtake.

C. *Validity of Security and Fasrness*

In order to verify our algorithm for solving the problem of driverless bus decision making, we compare DQN algorithm with sarsa-algorithm in driverless bus decision making.

We have made 5000 episodes for the both algorithms in each experiment. Both of them are trained for 2000 episodes. The following episodes are used to test the training result.

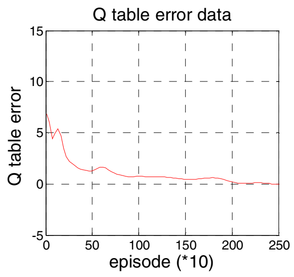


Fig.10. Q-table error data in training by sarsa-algorithm

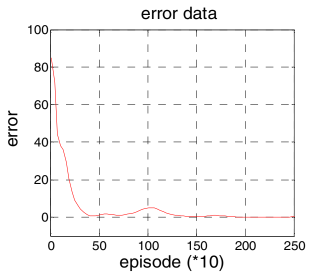
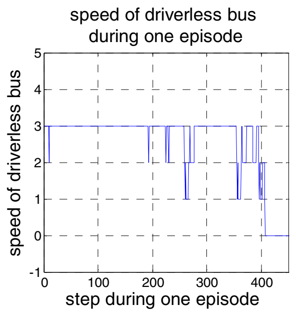


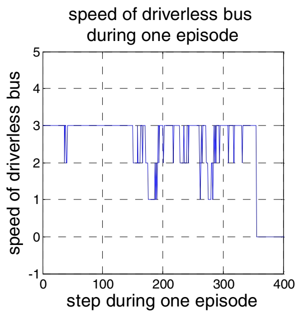
Fig.11. θ error data by DQN algorithm

The sarsa-algorithm is On-Policy algorithm because the strategies to follow when choosing an action and updating an action value function are the same, that is, the ε *-greedy* strategy. While the Q-learning algorithm is Off-Policy algorithm because it updates the action value function directly using the maximum $Q(s,a)$ value. The Off-Policy algorithm can converge faster compared with the On-Policy algorithm.

Fig.10. and fig.11. give the $\left[r+\gamma \max Q(s',a')-Q(s,a)\right]$ data in sarsa-algorithm and the θ error data in DQN algorithm. Where X axis is the episode, and Y axis is the error data of Q table in sarsa-algorithm and the error data of θ in DQN. We observe that both Q table in sarsa-algorithm and weight θ in DQN converge to a fixed value. Sarsa-algorithm takes approximately 2000 episodes to converge, but DQN algorithm adopts about 1300 episodes to converge. In this case, DQN algorithm reduces nearly 15% episodes to get a well training result.



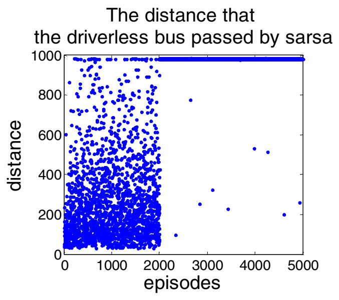
(a) DQN



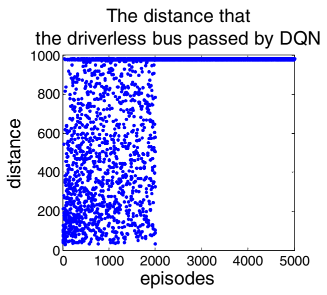
(b) Sarsa

Figure.12. driverless bus speed during one test

Fig.12 shows the speed of driverless bus in an episode. X axis is the step during one episode. Y axis is the speed of driverless bus by sarsa-algorithm and DQN algorithm. The average speed of driverless bus is 2.73 by DQN, and in sarsa the average speed is 2.515. The speed of driverless bus by DQN algorithm is 8.5% higher than the speed of driverless bus by sarsa-algorithm.



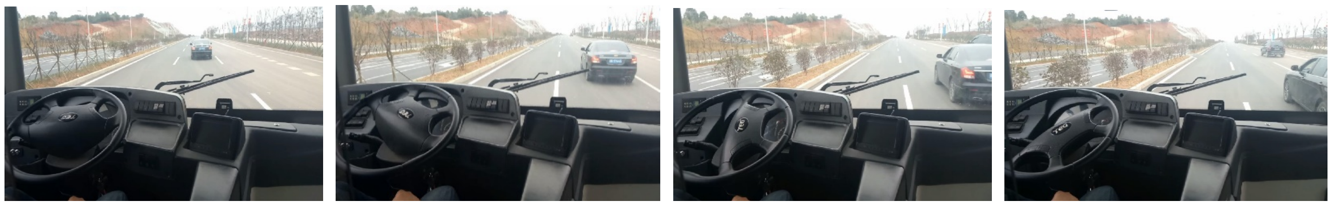
(a) Sarsa



(b) DQN

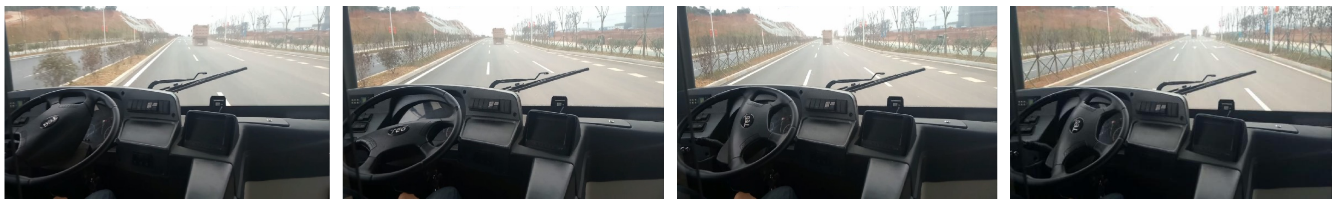
Fig.13. the distance of driverless bus passed

Fig.13 shows the distance of driverless bus. X axis means the episode. Y axis means the distance of driverless bus. The blue point shows the distance data. The driverless bus always gets collisions after being trained by sarsa-algorithm, but by the DQN algorithm the driverless bus gets fewer collisions.



(a) Frame 100 (b) Frame 200 (c) Frame 225 (d) Frame 250

Fig.14. Change to the left lane



(a) Frame 500 (b) Frame 550 (c) Frame 600 (d) Frame 650

Fig.15. Return to the right lane

Here are reasons why DQN performs better. Firstly, the Off-Policy algorithm can converge faster compared with the On-Policy algorithm. The sarsa-algorithm may not be trained very well after 2000 episodes. Secondly, DQN has a better generalization ability. When faced with a state never encountered, DQN can select appropriate action based on past experiences, however, sarsa-algorithm just select action randomly, which leads to more collisions.

IV. Results of Applying DQN to driverless Bus

We adopt 20 times real bus experiments on the road of 5km including three lanes, but all of the experiments are executed only on left two lanes and are completed successfully. The driverless bus has a maximum speed of 30 km/h. The overtaking process is as follows.

In Fig.14, the driverless bus changes to the left lane when the speed of the vehicle in front of the driverless bus is very low. In Fig.15, the driverless bus returns to the right lane after overtaking.

V. conclusion

As the unpredictable behavior of human drivers makes rule-based decision making methods more complex, the driverless bus using rule-based decision making methods is cautious, which may cause unnecessary delays. Reinforcement learning has received more and more attention recently. Although the construction of the model is not mentioned, the training process of DQN is the process of implicitly modeling.

The DQN proposed in this paper is a new approach that combines DNN (deep stacked autoencoders) and Q-learning. It is adopted to make overtaking decisions for driverless bus under structured urban environment. Experiments show that the driverless bus, which makes decisions using DQN, can travel without collision at a higher speed compared with sarsa. We believe that the application of DQN to driverless bus or even intelligent transportation systems will promote faster development of the entire unmanned field.

Acknowledgment

This paper was supported by National Natural Science Foundation of China (Grant No.61403426, 61403427). State Key Laboratory of Mechanical Transmissions of Chongqing University(SKLMT-KFKT-201602), State Key Laboratory of Robotics and System (HIT) (SKLRS-2017-KF-13).

References

[1] Wadud, Zia, “Fully automated vehicles: A cost of ownership analysis to inform early adoption." Transportation Research Part A Policy & Practice, pp. 101, 2017.

[2] Prakash, Shashikala, "Active vibration control of a full scale aircraft wing using a reconfigurable controller." Journal of Sound & Vibration 361, pp. 32-49, 2016

[3] Kurt, Arda, and Ümit Özgüner, "Hierarchical finite state machines for autonomous mobile systems." Control Engineering Practice 21, pp. 184-194, February 2013.

[4] Zheng, Rui, C. Liu, and Q. Guo, "A decision-making method for autonomous vehicles based on simulation and reinforcement learning." International Conference on Machine Learning and Cybernetics IEEE, pp. 362-369, 2014.

[5] Ngai, Daniel Chi Kit, and N. H. C. Yung, "A Multiple-Goal Reinforcement Learning Method for Complex Vehicle Overtaking Maneuvers." IEEE Transactions on Intelligent Transportation Systems 12, pp. 509-522, February 2011

[6] Cunningham, A. G, E. Galceran, and R. M. Eustice, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving." IEEE International Conference on Robotics and Automation IEEE, pp. 1670-1677, 2015.

[7] Liu, Wei, "Situation-aware decision making for autonomous driving on urban road using online POMDP." Intelligent Vehicles Symposium IEEE, pp. 1126-1133, 2015.

[8] Furda, Andrei, and L. Vlacic, "Enabling Safe Autonomous Driving in Real-World City Traffic Using Multiple Criteria Decision Making." IEEE Intelligent Transportation Systems Magazine 3, pp. 4-17, 2011.

[9] Czubenko, M, Z. Kowalczuk, and A. Ordys, "Autonomous Driver Based on an Intelligent System of Decision-Making, “ Cognitive Computation, pp. 569-581, July 2015.

[10] Shalev-Shwartz, Shai, S. Shammah, and A. Shashua, "Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving.", 2016.

[11] Lei, Tai, and L. Ming, "A robot exploration strategy based on Q-learning network." IEEE International Conference on Real-Time Computing and Robotics IEEE, pp. 57-62, 2016.

[12] Das, P. K., H. S. Behera, and B. K. Panigrahi, “Intelligent-based multi-robot path planning inspired by improved classical Q-learning and improved particle swarm optimization with perturbed velocity." Engineering Science & Technology An International Journal, pp. 651-669, January 2016.

[13] Hu, Hong, L. Pang, and Z. Shi, “Image matting in the perception granular deep learning." Elsevier Science Publishers B. V, p. 102, 2016.

[14] Chandra, B., and R. K. Sharma. "Deep Learning with Adaptive Learning Rate using Laplacian Score." Expert Systems with Applications63, pp. 1-7, 2016.

[15] Wang, Weining, "A multi-scene deep learning model for image aesthetic evaluation." Image Communication 47, pp. 511-518, 2016.

[16] Yoo, Youngjin, et al, "Deep Learning of Image Features from Unlabeled Data for Multiple Sclerosis Lesion Segmentation." International Workshop on Machine Learning in Medical Imaging Springer, Cham, pp. 117-124, 2014.

[17] Ma, Xiaorui, J. Geng, and H. Wang, “Hyperspectral image classification via contextual deep learning." Eurasip Journal on Image & Video Processing 2015, p. 20, 2015.

[18] Gu, Shixiang, "Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates." 2016.

[19] Isele, David, "Navigating Intersections with Autonomous Vehicles using Deep Reinforcement Learning." 2017.

[20] Sch&Xf, Bernhard, and lkopf, “Artificial intelligence: Learning to see and act." Nature 518.7540, p. 486, 2015.

[21] Mnih, V, "Human-level control through deep reinforcement learning, “Nature 518.7540, p. 529, 2015.

[22] Yuan, Xu, L. Buşoniu, and R. Babuška, "Reinforcement Learning for Elevator Control." IFAC Proceedings Volumes, pp. 2212-2217, February 2008.

[23] Morita, K, J. Jitsev, and A. Morrison, "Corticostriatal circuit mechanisms of value-based action selection: Implementation of reinforcement learning algorithms and beyond." Behavioural Brain Research, pp. 110-121, 2016.

[24] Lange, S, and M. Riedmiller. "Deep auto-encoder neural networks in reinforcement learning." *International Joint Conference on Neural Networks* IEEE, pp. 1-8, 2010.

[25] .