Autonomous Braking System via Deep Reinforcement Learning

Hyunmin Chae¹, Chang Mook Kang¹, ByoungDo Kim¹, Jackyum Kim¹,
Chung Choo Chung² and Jun Won Choi²†

Abstract—In this paper, we propose a new autonomous braking system based on deep reinforcement learning. The proposed autonomous braking system automatically decides whether to apply the brake at each time step when confronting the risk of collision using the information on the obstacle obtained by the sensors. The problem of designing brake control is formulated as searching for the optimal policy in Markov decision process (MDP) model where the state is given by the relative position of the obstacle and the vehicle’s speed, and the action space is defined as whether brake is stepped or not. The policy used for brake control is learned through computer simulations using the deep reinforcement learning method called deep Q-network (DQN). In order to derive desirable braking policy, we propose the reward function which balances the damage imposed to the obstacle in case of accident and the reward achieved when the vehicle runs out of risk as soon as possible. DQN is trained for the scenario where a vehicle is encountered with a pedestrian crossing the urban road. Experiments show that the control agent exhibits desirable control behavior and avoids collision without any mistake in various uncertain environments.

I. INTRODUCTION

Safety is one of top priorities that should be pursued in developing autonomous vehicles. For safe autonomous driving, autonomous vehicles should perceive the environments using the sensors such as camera, GPS, Lidar, and radar, and control the vehicle such that it can travel to the destination without any accidents. Since it is inevitable for an autonomous vehicle to encounter with dangerous and unexpected situations, it is critical to develop the reliable driving systems that can cope well with such uncertainties. Recently, several safety systems including collision avoidance, pedestrian detection, and front collision warning (FCW) have been proposed to enhance the safety of the autonomous vehicle [1]–[4].

One critical component for enabling safe autonomous driving is the autonomous braking systems which can reduce the velocity of the vehicle automatically when a threatening obstacle is detected. The autonomous braking should offer safe and comfortable brake control without exhibiting too early or too late braking. Most conventional autonomous braking systems are rule-based, which designate the specific brake control protocol for each different situation. Unfortunately, this approach is limited in handling all scenarios that can happen in real roads. Hence, it is necessary to develop the intelligent braking system which avoid the accidents in a principled and goal-oriented manner.

Recently, interest in machine learning has explosively grown up with the rise of parallel computing technology and a large amount of training data. In particular, the success of deep neural network (DNN) technique led the researchers to investigate the application of machine learning for autonomous driving. The DNN has been applied to autonomous driving from camera-based perception [5]–[7] to end-to-end approach which learns mapping from the sensing to the control [8], [9]. Reinforcement learning (RL) technique has also been improved significantly as DNN was adopted. The technique, called deep reinforcement learning (DRL), has shown to perform reasonably well for various challenging robotics and control problems. In [10], the DRL technique called Deep Q-network (DQN) was proposed, which approximates Q-value function using DNN. The DQN achieved the performance surpassing the human experts for various Atari video games. Recently, the DRL is applied to design control systems for autonomous driving in [11], [12].

In this paper, we propose a new autonomous braking system based DRL, which can intelligently control the velocity of the vehicle in situations where collision is expected if no action is taken. The proposed autonomous braking system is described in Fig. 1. The agent (i.e., vehicle) interacts with the environment where the position of the obstacle changes in time, thus affecting the risk of collision at each time step. The agent receives the information on the obstacle’s position using the sensors and adapts the brake control to the state change such that the chance of accident is minimized.

Fig. 1. The proposed DRL-based autonomous braking systems.
In our work, we design the autonomous braking system for the urban road scenario where a vehicle faces the pedestrian who crosses the street at random timing. In order to find the desirable brake action for the given pedestrian’s location and vehicle’s speed, we need to allocate appropriate reward function for each state-action pair. In our work, we focus on finding the desirable reward function which strikes the balance between the penalty imposed to the agent when accident happens and the reward obtained when the vehicle quickly gets out of risk. Using the reward function we carefully designed, we apply DQN method to learn the policy that decides the timing of brake based on the given pedestrian’s state. Note that this mechanism enables goal-oriented braking operation, thus offering more robust control behavior as compared to the conventional rule-based approaches.

Via computer simulations, we evaluate the performance of the proposed autonomous braking system. In simulations, we consider the uncertainty of the vehicle’s initial velocity, pedestrian’s initial position, and whether the pedestrian will cross or not. The experimental results show that the proposed braking system exhibits desirable control behavior and achieves excellent performance in collision avoidance for various test scenarios.

The rest of this paper is organized as follows. In Section II, we describe the basic scenarios and the framework of the proposed system. In Section III, we provide the details of the DQN design for autonomous braking. The experimental results are provided in Section IV and the paper is concluded in Section V.

II. System Description

In this section, we describe the overall structure of the autonomous braking system. We first define the possible scenarios for autonomous braking and explain the detailed operation of the proposed system.

A. Scenarios

One of the factors that hinders safe driving in autonomous driving is the threat of accidents from nearby objects, e.g., pedestrians. Many accidents could happen when the vehicle cannot detect a pedestrian or fails to stop ahead of it when a pedestrian crosses the road. Hence, in order to avoid accidents, we need smart autonomous braking system which can detect the threat that can potentially cause accidents in advance and perform appropriate brake actions to stop vehicle in front of the obstacle. However, there exist various degrees of uncertainty which make the design of autonomous braking difficult such as

- Vehicle’s initial velocity
- Pedestrian’s position
- Pedestrian’s speed
- Pedestrian’s crossing timing
- Pedestrian’s moving direction
- Sensor’s measurement error
- Road’s condition

Even if a pedestrian is detected accurately, it is hard to know when it can become a threat to the vehicle so that the braking strategy should change for different situations. (see Fig. 2)

Hence, for the given state of the pedestrian, the autonomous braking system should intelligently decide how the brake is applied.

![Behavior of a pedestrian modeled by the discrete-state Markov process.](image)

In our system, we consider the scenario where behavior of the pedestrian follows the discrete-state Markov process described in Fig. 3. The state $S_{nobody}$ implies that the sensors have not detected any obstacle. Once a pedestrian is detected, the state $S_{nobody}$ can change to the state $S_{stay}$ or the state $S_{cross}$, where $S_{stay}$ is the state that the pedestrian stays at sidewalk and $S_{cross}$ is the state that the pedestrian crosses the road. The pedestrian’s initial position can be either from far-side and near-side of the vehicle and the pedestrian walking speed can vary between $v_{pedmin} \text{ m/s}$ and $v_{pedmax} \text{ m/s}$. Note that the vehicle’s initial velocity distributes between $v_{vehmin} \text{ m/s}$ and $v_{vehmax} \text{ m/s}$. It is difficult to know the transition probabilities of the Markov process and the distribution of the pedestrian’s states in practical scenarios. Therefore, reinforcement learning approach can be effective
in learning the brake control strategy through the interaction with environment.

B. Autonomous Braking System

The detailed operation of the proposed autonomous braking system is depicted in Fig. 4. The vehicle is moving at speed \( v_{veh} \) and at the position \( (vehpos_x, vehpos_y) \). As soon as a pedestrian is detected, the autonomous braking system receives the relative position of the pedestrian, i.e., \( (pedpos_x - vehpos_x, pedpos_y - vehpos_y) \) from the sensors, where \( (pedpos_x, pedpos_y) \) is the location of the pedestrian. Using the vehicle's velocity \( v_{veh} \) and the relative position \( (pedpos_x - vehpos_x, pedpos_y - vehpos_y) \), the vehicle decides whether it will step brake at each time step. The interval between consecutive time steps is given by \( \Delta T \). We consider only two brake actions: no braking \( a_{nothing} \) and braking \( a_{decel} \). We can include more brake actions with more refined steps, e.g., strong, mid, weak brake actions, which are not considered in this work.

III. DEEP REINFORCEMENT LEARNING FOR AUTONOMOUS BRAKING SYSTEM

In this section, we present the details of the proposed DRL-based autonomous braking system. We first introduce the structure of the DQN and explain the architecture and the reward function used to train the DQN.

A. Structure of DRL

Our system follows the basic RL structure. The agent performs an action \( A_t \) given state \( S_t \) under policy \( \pi \). The agent receives the state as feedback from the environment and gets the reward \( r_t \) for the action taken. The state feedback that the agent takes from sensors consists of the velocity of the vehicle \( v_{veh} \) and the relative position to the pedestrian, \( (pedpos_x - vehpos_x, pedpos_y - vehpos_y) \) for the past \( n \) time steps. Possible action that agent can choose is either deceleration \( a_{decel} \) or keeping the current speed \( a_{nothing} \). Within this relationship between the agent and the environment, the goal of our proposed autonomous braking system is to maximize the accumulated reward called “value function” that will be received in the future within a episode. Using the computer simulations, the agent learns from interaction with environment episode-by-episode. One episode starts when a pedestrian is detected. Note that the initial position of the pedestrian and the initial velocity of the vehicle are random. The vehicle drives on a straight way based on the brake policy \( \pi \). If the distance between the vehicle and the pedestrian is less than the safety radius \( l \), it is considered as a collision event. (see Fig. 4) The episode is over if at least one of the following events occurs

- **Stop**: the vehicle completely stops, i.e., \( v_{veh} = 0 \).
- **Bump**: the distance between car and pedestrian is below the safety radius \( l \).
- **Pass**: the vehicle reaches the destination located 5 meters past the pedestrian.

Once one episode is over, the next episode starts with the state of environment and the value function reset.

B. Deep Q-Network

Q-learning is one of the popular RL methods which searches for the optimal policy in an efficient way [13]. Basically, the Q-value function \( q_{\pi}(s, a) \) is defined as

\[
q_{\pi}(s, a) = E_{\pi}\left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid S_t = s, A_t = a \right]
\]  

(1)

for the given state \( s \) and action \( a \). The Q-value function is the expected sum of the future rewards which indicates how good the action \( a \) is given the state \( s \) under the policy of the agent \( \pi \). The contribution to the Q-value function decays exponentially with the discounting factor \( \gamma \) for the rewards with far-off future. For the given Q-value function, the greedy policy is obtained as

\[
\pi(s) = \arg \max_a q_{\pi}(s, a).
\]  

(2)

One can show that for the policy provided in (2), the following Bellman equation should hold [13]:

\[
q_{\pi}(s, a) = E_{\pi}\left[ r_{t+1} + \gamma \max_{a'} q_{\pi}(S_{t+1}, a') \mid S_t = s, A_t = a \right].
\]  

(3)

In practice, since it is hard to obtain the exact value of \( q_{\pi}(s, a) \) satisfying the Bellman equation, the Q-learning method uses the following update rule for the given one step backups \( S_t, A_t, r_{t+1}, S_{t+1} \):

\[
q_{\pi}(S_t, A_t) \leftarrow q_{\pi}(S_t, A_t) + \alpha \left( r_{t+1} + \gamma \max_a q_{\pi}(S_{t+1}, a) - q_{\pi}(S_t, A_t) \right)
\]  

(4)

However, when the state space is continuous, it is impossible to find the optimal value of the state-action pair \( q_{\pi}(s, a) \) for all possible states. To deal with this problem, the DQN method was proposed, which approximates the state-action value function \( q(s, a) \) using the DNN, i.e., \( q(s, a) \approx q_{\theta}(s, a) \) where \( \theta \) is the parameter of the DNN [10]. The parameter \( \theta \) of the DNN is then optimized to minimize the squared value of the temporal difference error \( \delta_t \):

\[
\delta_t = r_{t+1} + \gamma \max_a q_{\theta}(S_{t+1}, a') - q_{\theta}(S_t, A_t)
\]  

(5)

For better convergence of the DQN, instead of estimating both \( q(S_t, A_t) \) and \( q(S_{t+1}, a') \) in (5), we approximate \( q(S_t, A_t) \) and \( q(S_{t+1}, a') \) using the Q-network and the target
network parameterized by $\theta$ and $\theta^-$, respectively. The update of the target network parameter $\theta^-$ is done by cloning Q-network parameter $\theta$, periodically. Thus, (5) becomes

$$\delta_t = r_{t+1} + \gamma \max_a q_\theta(S_{t+1}, a') - q_\theta(S_t, A_t)$$

(6)

To speed up convergence further, replay memory is adopted to save one step backups by memory size and take a part of them randomly from the memory by batch size [10]. The backups in the batch is used to calculate the loss function $L$ which is given by

$$L = \sum_{t \in B} \delta_t^2,$$

(7)

where $B$ is the backups in the batch. Note that the optimization of parameter $\theta$ for minimizing $L$ is done through the stochastic gradient decent method.

C. Reward Function

Unlike video games, the reward should be appropriately defined by a system designer in autonomous braking system. As mentioned, the reward function determines the behavior of the brake control. Hence, in order to ensure the reliability of the brake control, it is crucial to use the properly defined reward function. In our model, there is conflict between two important objectives of brake control; 1) collision should be avoided no matter what happens and 2) the vehicle should get out of the risky situation quickly. If it is unbalanced, the agent becomes either too conservative or reckless. Therefore, we should use the reward function which balances two conflicting objectives. Taking this into consideration, we propose the following reward function

$$r_t = \alpha v_t - \beta v_t \mathbf{1}(S_t = \text{bump}) + \lambda \mathbf{1}(S_t = \text{pass})$$

(8)

$$\alpha, \beta, \lambda > 0$$

where $v_t$ is the velocity of the vehicle at the time step $t$ and $\mathbf{1}(x = y)$ has a value of 1 the statement inside is true and 0 otherwise. The first term $\alpha v_t$ in the reward function guides the agent to maintain its velocity since deceleration would leads to smaller accumulated reward in the future. On the other hand, the term $-\beta v_t \mathbf{1}(S_t = \text{bump})$ indicates the negative reward (i.e., penalty) that the agent receives when the accident occurs. Note that this penalty is proportional to the vehicle’s velocity, which reflects the damage to the pedestrian in case of collision. Without such dependency on the velocity, we observe that the agent does not reduce the speed in situation when the accident is not avoidable. This is why the agent desires to maximize the first term while the second term cannot be controlled. Note that $\alpha$ and $\beta$ are the parameters used to control the balance between two conflicting terms. The final term $\lambda \mathbf{1}(S_t = \text{pass})$ is added to prevent the agent to stop at around the destination. When the agent arrives at the destination, the episode will be over so that without this term, the agent does not receive any reward. As a result, the agent would be reluctant to drive to the destination.

IV. Experiments

In this section, we evaluate the performance of the proposed autonomous braking system via computer simulations.

A. Simulation Setup

In simulations, we used the commercial software PreScan which models vehicle dynamics in real time [16]. We generated the environment in order to train the DQN by simulating the random behavior of the pedestrian. Suppose that the location information obtained from the sensors is given to the agent. We added slight measurement noise to the relative position of the pedestrian. In each episode, the initial position of vehicle is set to $(0, 0)$. The initial velocity of the vehicle is uniformly distributed between $v_{veh}^{min} = 5$ m/s and $v_{veh}^{max} = 10$ m/s. Initially, the pedestrian stands either at the far-side or at near-side of the vehicle with equal probability. The pedestrian crosses the road once the vehicle passes the pedestrian crossing point $p_{trig}$. The pedestrian crossing point is also uniformly distributed between 20 m and 30 m. The safety radius $l$ for the pedestrian is set to 6 m. The agent chooses the brake control between $a_{decel} = -3$ m/s$^2$ and $a_{nothing} = 0$ m/s$^2$ every $\Delta T = 0.2$ second. The detailed simulation setup is summarized below.

- Initial velocity of vehicle $v_{init} \sim U(5, 10)$ m/s
- Velocity of pedestrian $v_{ped} \sim U(1, 1.5)$ m/s
- Initial pedestrian position $pedpos_x \sim U(60, 70)$ m
- Trigger point(crossing scenarios) $p_{trig} \sim U(20, 30)$ m
- Radius of safety circle $l = 6$ m
- $\Delta T = 0.2$ s
- $a_{decel} = -3$ m/s$^2$, $a_{nothing} = 0$ m/s$^2$

In the performance test, we try four different scenarios

- Scenario 1 : Crossing from farside of the vehicle
- Scenario 2 : Crossing from nearside of the vehicle
- Scenario 3 : Standing on farside of the vehicle
- Scenario 4 : Standing on nearside of the vehicle

Note that in scenario 3 and 4, we set the pedestrian crossing point $p_{trig}$ to positive infinity.

B. Training of DQN

We trained the DQN using 100,000 episodes. The network used in the DQN consists of five fully-connected layers which have 40, 30, 15, 5 and 2 nodes each. We used the RMSProp [15] algorithm to minimize the loss with learning rate $\mu = 0.0005$ and set the batch size to 32. The summary of the configuration is provided below:

- State buffer size: $n = 5$
- Network architecture: five fully-connected layers
- Nonlinear function: leaky ReLU [14]
- Number of nodes for each layers : [15(Input layer), 40, 30, 15, 2(Output layer)]
- RMSProp optimizer with learning rate 0.0005
- Replay memory size: 20000
- Batch size: 32
- Reward function: $\alpha = 1$, $\beta = 80$, $\gamma = 20$
Fig. 5 shows the plot of the total accumulated award (i.e., value function) achieved for each episode during the training. We observe the value function converges after going through 70,000 episodes and high total reward is steadily attained in the end.

![Graph showing the value function convergence](image)

**C. Test Results**

Safety test was conducted for four different scenarios. For each scenario, we test the performance of the proposed system 10,000 times. The simulation parameters are varied according to the distribution specified in the section IV.A. We evaluate the performance by counting the number of episodes which end up with the state of Bump, Stop, and Pass, respectively. Table I provides the results for each test scenario. We observe that the agent avoids collision successfully for all cases in the scenario 1 and 2 (i.e., crossing scenarios). The agent just passed the pedestrian without unnecessary stops for the scenario 3 and 4 (i.e., standing scenarios). Fig. 6 shows the plot of actual position of the stopped vehicle and that of the pedestrian for 1,000 trials in the scenario 1. We see that the vehicle stops around 10 meters in front of the pedestrian for most of cases. This seems to be reasonably safe braking operation if we account for the safety radius of $l = 6 \, \text{m}$. Note that this distance can be adjusted by changing the reward parameters $\alpha$ and $\beta$. Overall, the experimental results show that the proposed autonomous braking system exhibits reliable and consistent brake control performance.

![Graph showing the vehicle and pedestrian positions](image)

**TABLE I**

| Scenario   | Bump | Stop | Pass |
|------------|------|------|------|
| Scenario 1 | 0%   | 100% | 0%   |
| Scenario 2 | 0%   | 100% | 0%   |
| Scenario 3 | 0%   | 0%   | 100% |
| Scenario 4 | 0%   | 0%   | 100% |

![Figure 6: Vehicle position graph](image)

**V. CONCLUSIONS**

We have presented the new autonomous braking system based on the deep reinforcement learning. The proposed system learns an intelligent way of brake control from the experiences obtained under the simulated environment. We designed the autonomous braking systems using the DQN method with carefully designed reward function. We show through computer simulations that the proposed autonomous braking system exhibits desirable and consistent brake control behavior for various scenarios where behavior of the pedestrian is uncertain. One limitation of the proposed system is incapability of training of the agent in real environment since it learns from trial and errors. However, this problem will be resolved by developing more accurate and realistic simulator. It would be interesting to find a reward function that can lead to the human-like brake action using inverse reinforcement learning techniques. This topic is left for the future research.

**REFERENCES**

[1] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, “Survey of pedestrian detection for advanced driver assistance systems,” IEEE transactions on pattern analysis and machine intelligence, vol. 32, no. 7, pp. 1239–1258, July 2010.

[2] A. Vahidi and A. Eskandarian, “Research advances in intelligent collision avoidance and adaptive cruise control,” IEEE Trans. Intelligent Transportation Systems, vol. 4, no. 3, pp. 143-153, Sept. 2003.

[3] K. Lee and H. Peng, "Evaluation of automotive forward collision warning and collision avoidance algorithms," Vehicle System Dynamics, vol. 43(10), pp. 735-751, Feb. 2007.

[4] T. Gandhi and M. M. Trivedi, "Pedestrian collision avoidance systems: a survey of computer vision based recent studies," IEEE Trans. Intelligent Transportation Systems Conference, pp. 976-981, Sept. 2006
[5] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andrilukam, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng, "An empirical evaluation of deep learning on highway driving," arXiv preprint, arXiv:1504.01716, Apr. 2015.
[6] D. Tome, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," Signal Processing: Image Communication, vol. 47, pp. 482-489, May. 2016.
[7] R. S. Tomar and S. Verma, "Neural network based lane change trajectory prediction in autonomous vehicles," Transactions on computational science XIII, Springer Berlin Heidelberg, pp. 125-146, 2011.
[8] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao and K. Zieba, "End to end learning for self-driving cars," arXiv preprint, arXiv:1604.0731, Apr. 2016.
[9] C. Chen, A. Seff, A. Kornhauser and J. Xiao, "DeepDriving: learning affordance for direct perception in autonomous driving," Proceedings of the IEEE International Conference on Computer Vision(ICCV), pp. 2722-2730, 2015.
[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518(7540), pp. 529-533, Feb. 2015.
[11] J. Koutnik, J. Schmidhuber and F. Gomez, "Evolving deep unsupervised convolutional networks for vision-based reinforcement learning," Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM, pp. 541-548, 2014.
[12] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: a reinforcement learning approach," IEEE Trans. Intelligent Transportation Systems, vol. 12, no. 4, pp. 1248-1260 Dec. 2011.
[13] R. S. Sutton and A. G. Barto, "Reinforcement learning: an introduction," MIT Press, 1998
[14] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," Proceedings of the 27th International Conference on Machine Learning(ICML), pp. 807-814, 2010.
[15] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Devide the gradient by a running average of its recent magnitude," COURSERA: Neural network for machine learning, 4.2, 2012.
[16] PreScan, https://www.tassinternational.com/prescan