2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

# Deep Q Learning Based High Level Driving Policy Determination

Kyushik Min, Hayoung Kim and Kunsoo Huh, *Member, IEEE*

***Abstract*— With the commercialization of various Driver Assistance Systems (DAS), those vehicles have some autonomous functions like Smart Cruise Control (SCC) and Lane Keeping System (LKS). It is believed that autonomous driving can be achieved by combining the DAS functions in the limited situations such as on highways. However, in order to coordinate the DAS functions for autonomous driving, a supervisor is needed to select an appropriate DAS function. In this paper, we propose a method for training a supervisor that selects proper DAS by deep reinforcement learning. The driving policy operates based on camera images and LIDAR data that are accessible in autonomous vehicles. Therefore, deep reinforcement learning network model is designed to analyze both camera image and LIDAR data. This system aims to drive in simulated traffic situation of highway without collision and with high speed. Unlike the systems which learn how to throttle, brake and steering directly, the proposed method can guarantee safe driving because the learned driving policy is based on the existing commercialized DAS functions. In order to verify the algorithms, a simulation tool is developed using Unity for highway environment with multiple vehicles and autonomous driving performance is compared with the proposed supervisor.**

## I. INTRODUCTION

### A. Motivation

Safe autonomous driving has been actively developed over the past few years. One of the important topic on autonomous driving is to find the best driving policy or supervisor that offers safety and comfort to the driver. The challenge on this problem is that the driving policy should satisfy the robustness regardless of various traffic environment. However, rule based algorithms are difficult to cope with various situations. Recently, deep reinforcement learning, which combines deep neural network model and reinforcement learning, shows significant progress in many fields [1]–[4]. Deep reinforcement learning algorithms perform self-training through various experience and show good performance using high-dimensional inputs such as raw image. Leveraging these advantages, we approach the autonomous driving problem with deep reinforcement learning.

### B. Related Works

The researches on the safe autonomous driving can be divided into two approaches: the first is traditional perception, planning and control framework and the other is learning based methods. The learning based methods have been recently popular due to successful process of high-dimensional features such as Convolutional Neural Network (CNN) in the field of computer vision [5]–[7]. The reinforcement learning algorithms can maximize the expected sum of future rewards and there are many studies to combine these two techniques for the autonomous driving.

For lane keeping, Rausch et al [8] proposed a method to train the network that predicts the steering angle directly based on the image obtained from the front camera. It was shown that the network can learn the steering angle for lane keeping by learning the features (lanes, etc.) automatically from the raw image of front camera. John et al [9] proposed mixture-of-expert framework for calculate proper steering angle for each scene by using long short term memory (LSTM). Each individual expert network models an expert drivers behavior in a specific partition of a given road scene such as straight driving, right turning and left turning. It worked well on multiple driving sequences as they considered variety of driving scenes. Al-Qizwini et al [10] proposed a regression network that predicts useful states for driving such as crossing errors, heading errors, and distances from obstacles in front camera images, rather than predicting steering angles directly from front camera images by using GoogLeNet [11] structure. The steering angle, throttle, and brake values are calculated using a simple if-else rule based algorithm using this information.

Sallab et al [12] proposed a method to learn driving policy for lane keeping using DQN (Deep Q Network) and DDAC (Deep Deterministic Actor Critic) in the absence of obstacles. They directly learned steering, acceleration, and deceleration to maximize expected future reward based on low-dimensional features such as speed, position of the track borders. As a result, the performance of lane keeping is improved by using DDAC which can be applied in continuous action space rather than DQN applicable in discrete action space. Zong et al [13] proposed a method to learn steering angle and acceleration value by applying DDPG [14] for obstacle avoidance. The above method directly learns the proper steering angle and amount of throttle and brake necessary to control the vehicle. However, in these cases, since the optimum policy changes every time the parameters of the vehicle are changed, there is a limitation that the learning should be newly performed for the optimum policy.

In this paper, we propose an autonomous driving framework using conventional DAS and deep reinforcement learning. Proposed method determines the DAS functions such as lane changing, cruise control and lane keeping that maximize average speed and number of overtaking with few lane changes. The action space is defined based on the behavior level and the driving policy is learned to make a behavior level decision such as lane keeping, lane change and cruise

Kyushik Min, Hayoung Kim and Kunsoo Huh are with the Department of Automotive Engineering, Hanyang University, Seoul, 04763, Republic of Korea {mks0813, hayoung.kim, khuh2}@hanyang.ac.kr

control. For the verification of proposed algorithm, the algorithm is tested in the simulation of the dense traffic situation and it is demonstrated that the performance is improved in terms of the average speed, number of overtake and number of lane change as the learning progresses while driving.

### C. Overview

The remaining sections of the paper are constructed as follows: Section II defines the markov decision process(MDP) environment for solving the autonomous driving problem and explains the observations, actions, and reward functions according to the MDP. Section III discusses the deep reinforcement learning algorithms and network architectures used in this study. In Section IV, the simulation results are obtained by deep reinforcement learning algorithms structured for multi inputs.

## II. Problem Formulation

### A. MDP for Driving Policy Learning

Markov Decision Processes (MDP) is a mathematical framework for decision making, which is composed of tuple $< S, A, T, R, \gamma >$ where set of states $S$, set of actions $A$, transition model $T$, reward function $R$ and discount factor $\gamma$ [15]. The problem to solve MDP is to find a policy that maximizes the expected sum of discounted reward $\prod_{t=0}^{T} \gamma^t R(s_t, a_t)$ for given reward function $R$ as a problem of finding policy for decision making. However, in recent deep learning, it is possible to effectively train the deep neural network from a large data set, and it is possible to solve the MDP by using some fixed state representation $\phi(s_t)$ obtained from the raw input instead of states $s_t$. Indeed, deep neural networks have learned better representations than handcrafted features in computer vision researches.

Driving policy learning also proceeds based on the MDP, in which the host vehicle interacts with the environment including surrounding vehicles and lanes. we define the observation states $S$, action space $A$, and reward function $R$ for driving policy learning as follows using the advantage of deep reinforcement learning that learn good representations itself.

### B. Observation

The observation state is constructed using both LIDAR sensor data and camera image data. The total coverage of sensor configuration can be seen in Fig. 1. The distances to obstacles for each 1 degree can be obtained from LIDAR sensor as shown in Fig. 2 and the raw image from frontside also can be obtained to configure observation. As the range data from LIDAR and the image data from camera have totally different characteristics, the network with multi-modal input scheme is used in this study.

### C. Action

The action space for driving policy is defined in discrete action space. As we leverage the advantages of conventional DAS, each action for this system is the activation of each DAS function. In longitudinal direction, there are three kinds

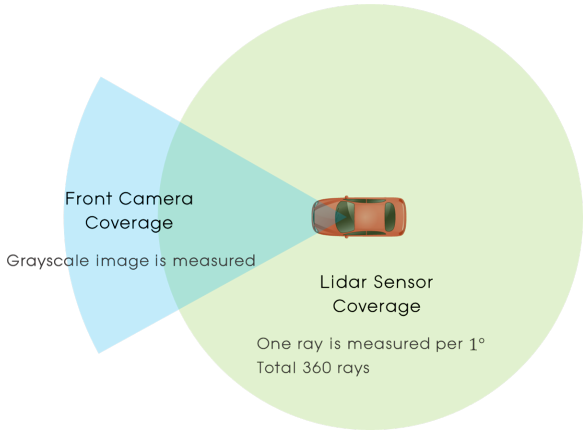


Fig. 1. Sensor configuration of the vehicle for simulation. The LIDAR sensor detects around the vehicle, and the camera sensor detects the frontside of the vehicle.

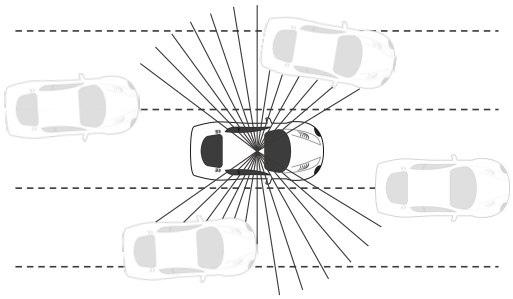

Fig. 2. LIDAR sensor example for surrounding vehicles. It returns an array of distances to the obstacles

of actions: 1. cruise control with speed $v + v_{cc}$ where $v_{cc}$ is additional target speed, which is set to 5km/h , 2. cruise control with current speed $v$, 3. cruise control with speed $v - v_{cc}$. These longitudinal actions are expected to cover autonomous emergency braking (AEB) and adaptive cruise control (ACC). In lateral direction, there are also three kind of actions: 1. keeping lane, 2. changing lane to left, 3. changing lane to right. As autonomous vehicle should drive both in longitudinal and lateral direction at same time, we defined 5 discrete actions. (No action, accleration, deceleration, lane change to left, lane change to right)

### D. Reward Function

If you select an action in terms of reinforcement learning, you will receive a reward as a result of the action. As we have seen, the problem to solve on the MDP is to find a driving policy that maximizes the expectation value of the future reward. This means that the optimal driving policy can be completely different depending on how the reward is designed. Therefore, it is very important to design the reward appropriately in order to learn the proper driving policy.

When the vehicle travels in a dense traffic situation, it should satisfy the following three conditions: 1. To find policy that makes the vehicle to drive at high speed, 2. To drive with collision-free trajectory, 3. To not change the lane too often. We have designed the rewards to satisfy these three conditions.

$$r_v(v) = \frac{v - v_{min}}{v_{max} - v_{min}} r_{v,max} \tag{1}$$

$$r_{col} = \begin{cases} -r_{collision} & \text{if host vehicle colides} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$r_{lc} = \begin{cases} -r_{lanechange} & \text{if host vehicle changes lane} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$r_{overtake} = \begin{cases} r_{overtake} & \text{if host vehicle overtake other vehicle} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$r_{tot}(v) = r_v(v) + r_{col} + r_{lc} + r_{overtake} \tag{5}$$

where $v$ is current vehicle speed, $v_{max}$ is maximum speed that does not violate the law and $v_{min}$ is minimum speed. $r_{v,max}$ is the reward at maximum speed, $v_{lc}$ is the penalty for lane change, $r_{collision}$ is the penalty for collision and $r_{overtake}$ is reward for overtaking other vehicle.

In terms of safety, the collision have to be avioded in driving situation, so the $r_{collision}$ is given a larger value than any other reward value to prevent conflict behavior in the first place. Since it is desirable to travel as fast as possible under $v_{max}$, it is possible to obtain a linear reward according to the current speed below $v_{max}$. Optimal driving policy makes host vehicle to overtake slow proceding vehicles. Therefore, $r_{overtake}$ is added to final reward to encourage to find optimal policy. Finally, as it is not desirable to change the lane too frequently, value of $r_{lanechange}$ is set to learn to drive without unnecessary lane changes. All the parameters for rewards are defined as Table I.

Fig. 3 is an example of the $r_{tot}$ received when a vehicle traveling at 60 km/h changes its lane and then drives faster toward 70 km/h. The reward decreases with a lane change, but increases with a speed up and overtaking.

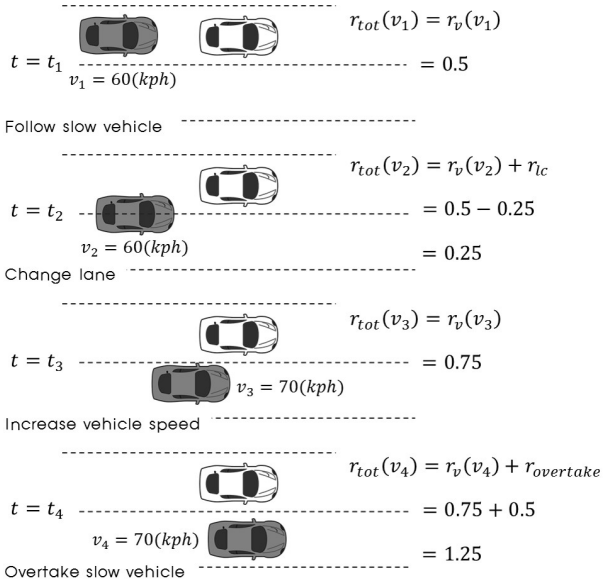


Fig. 3. Immediate reward for overtaking slow vehicle scenario

TABLE I
PARAMETERS FOR REWARDS

| | |
|---|---|
| $r_{v,max}$ | 1 |
| $v_{max}$ | 80km/h |
| $v_{min}$ | 40km/h |
| $r_{collision}$ | -10 |
| $r_{lc}$ | -0.25 |
| $r_{overtake}$ | 0.5 |

## III. DEEP RL ALGORITHMS FOR POLICY LEARNING

After big success of DQN in game domain which combined with reinforcement learning and neural network, deep reinforcement learning has been studied in various ways [16]. Especially, a number of studies have been conducted in the case of value-based deep reinforcement learning [17]–[22] that is based on DQN. Lately, rainbow DQN [23] was published which integrates recent value-based deep reinforcement learning studies. In this study, deep reinforcement learning algorithm is combined by DQN [1], Double DQN [17] and Dueling DQN [19]. In addition, the algorithms we used are briefly explained in following sections.

### A. DQN

Deep Q network [1] is the technique that combines convolutional neural network (CNN) and reinforcement learning. It uses raw pixel frame of the screen as observation $(O_t)$. $O_t$ assigned as input of CNN and CNN derives Q-values of each action $(A_t)$ as output. At this time, action is selected according to the $\epsilon - greedy$ technique. Next observation $O_{t+1}$ and reward $R_t$ are obtained after performing action. These obtained information is set as an experience $e_t = (O_t, A_t, R_t, O_{t+1})$ and stored in dataset $D_t = \{e_1, e_2, ..., e_t\}$. While training, experience replay technique [18] is used that randomly samples the data in the dataset and performs a mini-batch update. Neural network learns its parameters to minimize loss by performing stochastic gradient descent.

$$L(\theta) = (R_t + \gamma \max_{A_{t+1}} Q(O_{t+1}, A_{t+1}; \theta^-) - Q(O_t, A_t; \theta))^2 \tag{6}$$

In this case, $\gamma$ is a discount factor that determines how much of the expected sum of the future reward is considered. $\theta$ is parameters of neural network and $\theta^-$ is parameters of target network which copies all the parameters of neural network every specific steps for stable training.

### B. Double DQN

After the success of DQN, many algorithms have been proposed that reinforce various disadvantages of DQN. One of them is double DQN (DDQN) algorithm [17] which reduces the overestimation problem of the action value, which DQN has. When calculating target value, same Q-function is used for both selecting and evaluating action in the case of DQN.

$$y^{DQN} = R_t + \gamma \max_{A_{t+1}} Q(S_{t+1}, A_{t+1}; \theta^-) \tag{7}$$

As a result, overestimation of value occurs, which causes the performance degradation of the DQN. DDQN solves the problem of value overestimation by separating the process of selecting and evaluating action using two Q-functions when determining the target value.

$$y^{DDQN} = R_t + \gamma Q(S_{t+1}, \underset{A_{t+1}}{argmax} Q(S_{t+1}, A_{t+1}; \theta); \theta^-) \tag{8}$$

### C. Dueling DQN

In the case of Dueling DQN [19], convolution part of the DQN network is used as it is and fully connected part is divided into two separated estimators: one for state-value and the other for advantage of each action. The final equation for calculating Q-value is changed to aggregate the stats-value and action advantages.

$$Q(S_t, A_t; \theta, \alpha, \beta) = V(S_t; \theta, \beta) + A(S_t, A_t; \theta, \alpha) - \frac{1}{|A|} \sum_{A_t} A(S_t, A_t; \theta, \alpha)) \tag{9}$$

In the equation, $\alpha$ is fully connected layer parameters of advantage function $A$ and $\beta$ is fully connected layer parameters of state-value $V$. Dueling DQN, which uses two stream network, shows significant performance improvement than DQN. In addition, Dueling DQN guarantees the robustness of performance even if the number of actions is increased.

## IV. Simulation and Results

Deep reinforcement learning algorithms (DQN, DDQN, Dueling DQN) are combined to find optimal driving policy in simulations. The objective of the algorithms is to drive at a high speed without collision to other vehicles and without unnecessary lane changes.

### A. Simulator

The simulator used in this paper is constructed using Unity and Unity ML-Agents. The simulated road environment is a highway driving roadway composed of five lanes. Other vehicles are generated in the center of the random lane within a certain distance to the host vehicle. In addition, it is assumed that other vehicles do not collide to each other in most cases and can perform five actions (acceleration, deceleration, lane change to right lane, lane change to left lane, keep current state). Various actions by other vehicles change the simulation environment in many ways, so agent enables to experience many different situations. The observations from the simulator are two types: one is image and the other is LIDAR-like range array. As there is a camera on front, raw pixel image is observed for every step. LIDAR sensor detects the range of 360 degrees which has one ray per degree. If the ray hits an object, it returns the distance between the host vehicle and the object. If there is no obstacle, it returns the maximum sensing distance for every step of simulator.


Fig. 4. Simulator example image.

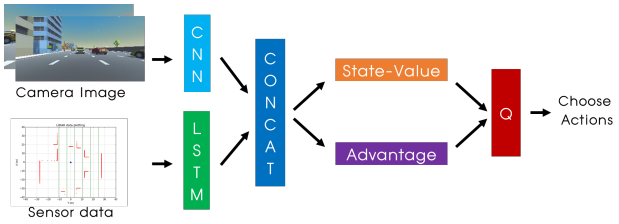


Fig. 5. Muti-modal driving policy network architecture. It takes two kinds of inputs: camera raw images with grayscale and LIDAR sensor data with array of distances in meter.

TABLE II
HYPERPARAMETERS OF DRIVING POLICY NETWORK

| Data | Type | Actuation | Hyperparameters |
|---|---|---|---|
| Camera data | Convolution | ReLU | patch size = (8x8)<br>stride = 4<br># of filters = 32 |
| | Convolution | ReLU | patch size = (4x4)<br>stride = 2<br># of filters = 64 |
| | Convolution | ReLU | patch size = (3x3)<br>stride = 1<br># of filters = 64 |
| Sensor data | LSTM | - | time steps = 4<br># of cell states = 256 |
| Concatenated data | Fully Connected | ReLU | # of units = 512 |

### B. Network Architecture and Hyper-parameters

For training driving policy, the convolutional layers of original DQN structure is changed to analyze the multi-input, because the proposed algorithm uses two different inputs at the same time. In the case of camera image, data is refined using the convolutional neural network like DQN and LIDAR data is refined using the long short-term memory (LSTM). After concatenating these two refined data, this data is used as an input of the fully connected layer to obtain Q-values which determine action. Deep reinforcement learning algorithm have structure as shown in Fig. 5. In the case of Dueling DQN, it uses two stream fully connected layer structure that derives both state-value and action-advantage. The overall structure of driving policy network and the hyperparameters for this network are listed in Table I. Adam optimizer [24] is used for training the policy network. The other hyperparameters used are as follows: the learning rate is 0.00025, the batch size is 32, the replay memory size is 100,000 and the number of skipped frames is 4.

TABLE III
COMPARISIONS OF PERFORMANCE

| Input Configuration | Average Speed (km/h) | # of Average Lane Change | # of Average Overtaking |
|---|---|---|---|
| Camera only | 71.88 | 16.8 | 35.8 |
| LIDAR only | 69.48 | 18.4 | 32.2 |
| Multi-Input | 73.54 | 30.2 | 42.2 |

### C. Results

The proposed driving policy determination algorithm is implemented using Tensorflow [25]. Average speed, number of lane changes and number of overtakes are shown in Fig. 6, Fig. 7 and Fig. 8, respectively. To verify the advantages of multi-input architecture, which combines features from camera and LIDAR through CNN and LSTM respectively, two policy networks for camera input only and LIDAR input only are additionally implemented. We compare those three different network architectures for different inputs: camera, LIDAR, both camera and LIDAR. As training proceeds, the autonomous vehicle tends to overtake more vehicles and drive in faster speed without unnecessary lane changes in every input configuration. As can be seen from the result (Table III), multi-input architecture shows the best performance in average speed and number of average overtaking, that the values are 73.54km/h and 42.2, respectively. However, the number of lane changes is the highest when using the architecture for multi-input, which has 30.2 as average value. Although the goal of the proposed algorithm is to decrease the number of unnecessary lane changes, the result of multi-modal input is the highest in the terms of the number of lane changes. However, In the case of architecture for camera and architecture for LIDAR, they sometimes show to follow the front vehicle without changing the lane, even if the preceding vehicle is slow. Therefore, bigger number of lane changes than those results is reasonable value to find optimal policy.

## V. CONCLUSIONS

In this paper, the driving policy network is proposed for autonomous driving with fully utilizing conventional DAS functions, which guarantees safety in most cases. Using deep reinforcement learning algorithm, the trained autonomous vehicle successfully drives in simulated highway scenario in high speed without unnecessary lane changes. As the proposed policy network used multi-modal inputs, the vehicle drives better than the one with single input in terms of average speed, the number of lane changes and the number of overtaking. The result of this study shows the possibility that autonomous vehicle can be controlled by the supervisor which is trained through deep reinforcement learning.

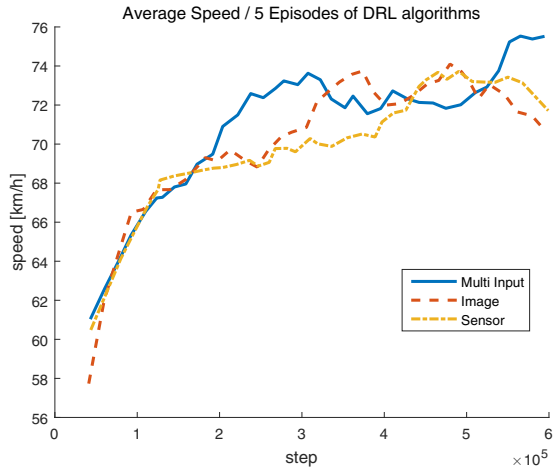


Fig. 6. Average speed per 5 episodes by multi input, image input and sensor input

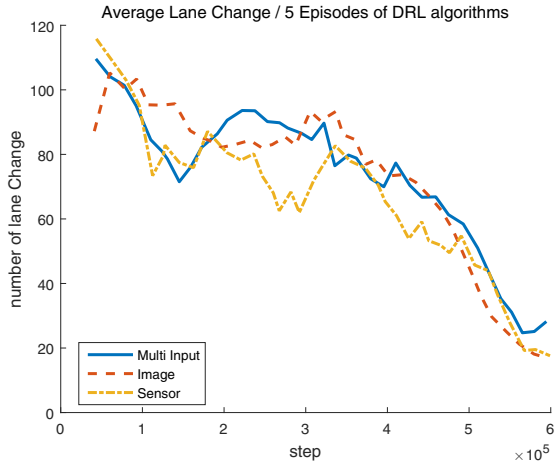


Fig. 7. Average number of lane changes per 5 episodes by multi input, image input and sensor input

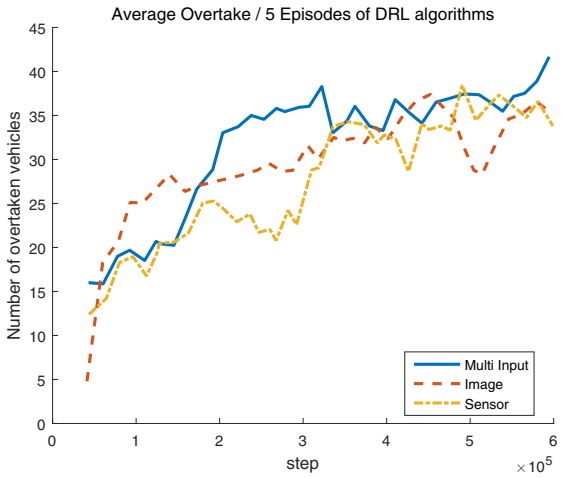


Fig. 8. Average number of overtake per 5 episodes by multi input, image input and sensor input

## ACKNOWLEDGMENT

This work was supported in part by a grant(18TLRP-B101406-04) from Transportation logistics research Program funded by Ministry of Land, Infrastructure and Transport of Korean government.

Also, this work was one of the projects of Machine Learning Camp Jeju 2017 hosted by Tensorflow Korea. Lastly, We thank the anonymous reviewers for their reviews.

## REFERENCES

[1] 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, p. 529, 2015.

[2] 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.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[4] A. DeepMind, "Reduces google data centre cooling bill by 40%," 2016.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[8] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 4914–4919.

[9] V. John, S. Mita, H. Tehrani, and K. Ishimaru, "Automated driving by monocular camera using deep mixture of experts," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 127–134.

[10] M. Al-Qizwini, I. Barjasteh, H. Al-Qassab, and H. Radha, "Deep learning algorithm for autonomous driving using googlenet," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 89–96.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions." Cvpr, 2015.

[12] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *arXiv preprint arXiv:1612.04340*, 2016.

[13] X. Zong, G. Xu, G. Yu, H. Su, and C. Hu, "Obstacle avoidance for self-driving vehicle with reinforcement learning," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 11, no. 2017-01-1960, 2017.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[15] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

[16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[17] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, vol. 16, 2016, pp. 2094–2100.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[19] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[20] 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*, 2016, pp. 1928–1937.

[21] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[22] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," *arXiv preprint arXiv:1707.06887*, 2017.

[23] 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," *arXiv preprint arXiv:1710.02298*, 2017.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.