Multi-objective optimization for autonomous driving strategy based on Deep Q Network

Tianmeng Hu¹ · Biao Luo¹ · Chunhua Yang¹

Received: 11 August 2021 / Accepted: 8 September 2021

Published online: 18 October 2021

© The Author(s) 2021 **OPEN**

Abstract

Autonomous driving is an important development direction of automobile technology, and driving strategy is the core of the autonomous driving system. Most works in this area focus on single-objective tasks, such as maximizing vehicle speed or lane-keeping, and rare attention has been paid to the quality of driving skills. Therefore, a multi-objective learning method is proposed for autonomous driving strategy based on deep Q-network, where two optimization objectives are involved, i.e., vehicle speed and passenger comfort. An end-to-end autonomous driving model is designed by using vehicle front camera images as inputs to the Q-network and makes decisions based on the output Q values. Considering the vehicle speed and passenger comfort, the reward function is designed for multi-objective optimization. To evaluate the effectiveness of the method, training and testing are performed in a simulator, and a single-objective strategy with the goal of maximizing speed is designed for comparison. The results show that the proposed multi-objective autonomous driving strategy can strike a balance between vehicle speed and passenger comfort. Compared with the single-objective strategy, the multi-objective strategy has a significant improvement in comfort, while the average speed is only slightly reduced.

Keywords Autonomous driving · Deep Q-Network · Deep reinforcement learning · Multi-objective optimization

1 Introduction

Autonomous driving is a potential and promising technology to improve road traffic efficiency and relieve traffic pressure compared with manual driving. In addition, it also provides convenience for people who are unable to drive. The goal of autonomous driving is to make the vehicle drive without part or full human control. Noting the road environment is dynamic and complex, an intelligent autonomous driving decision system is essential. To achieve the goal of autonomous driving, some studies employed non-learning approaches, which require manually designed driving strategies [1, 2]. It is feasible to design driving strategies with the help of human experience and knowledge, while artificially designed strategies are difficult to adapt to complex and dynamic environments. Some researchers also used supervised learning methods [3–5], where labeled data-action pairs were applied to train a neural network. However, these methods require a large amount of labeled data, which is very difficult to obtain and process. In addition, most works in this area focus on single-objective tasks, such as maximizing vehicle speed or lane-keeping [6], while the quality of driving skills was rarely considered, e.g., passenger comfort, efficiency in transporting passengers to their destinations.

✉ Biao Luo, biao.luo@csu.edu.cn; Tianmeng Hu, tianmeng0824@163.com; Chunhua Yang, ychh@csu.edu.cn | ¹Central South University, Changsha, China.

Reinforcement learning (RL) is a type of machine learning. The reinforcement learning agent obtains feedback through interaction with the environment, and selects actions based on the feedback policy. Reinforcement learning is good at solving sequential decision-making and planning problems, it is suitable for learning autonomous driving strategies. However, most research in this area have also been limited to single objectives, such as safety or speed. The motivation of this paper is to achieve multi-objective optimization on autonomous driving strategy with reinforcement learning by considering two objectives: optimizing vehicle speed and optimizing passenger comfort. By proposing a multi-objective reward function for both objectives, the multi-objective reinforcement learning approach is developed based on deep Q-network [7, 8].

2 Related work

The development of reinforcement learning is closely related to optimal control theory, including dynamic programming [9], adaptive dynamic programming [10]. Watkins et al. used value functions to evaluate strategies and proposed the Q-Learning method [11]. To solve more complex problems with larger state space and action space, deep reinforcement learning (DRL) was proposed. DRL combines deep learning with reinforcement learning, using neural networks to reduce the dimensionality of high-dimensional data before processing them using reinforcement learning methods. Mnih et al. proposed deep Q-network (DQN) [7, 8], which uses convolutional neural networks (CNN) for image downscaling and feature extraction, and then combined it with Q-learning algorithms for training RL agents to play Atari video games, and the algorithms outperformed human levels. Subsequently, many kinds of improved algorithms have been reported. For example, dueling DQN [12], deep deterministic policy gradient (DDPG) [13], asynchronous advantage actor-critic framework (A3C) [14], trust region policy optimization (TRPO) [15], and proximal policy optimization (PPO) [16]. Deep reinforcement learning methods have advantages in solving decision problems, which optimizes long-term rewards and does not require mathematical models and labeled data. Thus, they are applied to solve many difficult problems, such as, Go [17, 18], robot control [19], recommender systems [20–22], etc.

Previously, deep learning has had more applications in autonomous driving systems, while reinforcement learning has been used less often. In recent years, due to the development of deep reinforcement learning, many studies have applied DRL to autonomous driving. Mariusz et al. applied CNN to autonomous driving [3]. Sallab et al. applied the DQN algorithm to a lane-keeping task in a simulator [6]. Hyunmin et al. applied the DQN algorithm to an automatic braking system, and the RL agent was able to learn the braking strategy after 70,000 training steps in a simulator [23]. Wolf et al. used the DQN algorithm to train the RL agent to control an autonomous car in a simulator [24]. Chen et al. applied various DRL algorithms such as DDQN and soft actor-critic (SAC) in the simulator CARLA to achieve autonomous driving in complex traffic environments [25]. It is noted that most of these works focused on the safety or reliability of the driving process. In 2018, Kendall et al. applied deep reinforcement learning algorithms to a real environment [26]. They first trained in a simulator using DDPG and then tested it in a real environment, achieving results close to human level. Min et al. combined image data and distance sensor data as input to a DRL algorithm to train self-driving cars [27]. They designed rules based on experiential knowledge to guide the learning of RL agents. However, their approach only seeks to maximize the vehicle speed. Li et al. trained a multi-objective DQN to learn to drive on multi-lane roads and intersections [28]. Their work considered passenger comfort issues, but only used manually designed rules.

3 Preliminaries

In this section, some preliminaries are presented. First, a modeling description of autonomous driving is established. Then, in order to apply DRL to the autonomous driving problem, the interaction of vehicle and environment is modeled as a Markov decision process (MDP) [29]. Finally, a Q-network for end-to-end autonomous driving decision-making is constructed.

3.1 Problem formulation

The construction of the autonomous driving model and Markov decision process is presented in this subsection.

3.1.1 Autonomous driving model

The aim of this paper is to learn autonomous driving strategies that strike a balance between efficiency and comfort. We divide the autonomous driving model into three parts: perception, decision, and control.

1. The perception module is responsible for acquiring information from the environment. For human drivers, this part usually includes vision, hearing, and touch. In our model, we use the image data captured by a front camera as the input to a deep neural network.
2. The decision module is responsible for making driving decisions and picking driving actions. For human drivers, this part is thinking and decision-making. This is a high-level decision, and we learn driving strategies based on DQN.
3. The control module is responsible for executing driving actions. We use an environmental simulator responsible for the actual execution of the actions.

3.1.2 Markov decision process

An MDP consists of a tuple of five elements (S, A, P, R, γ) , where S is the state space, A is the action space, P is the state transfer function to describe the dynamic properties of the environment, R is the reward function, $R(S, a)$ denotes the immediate reward from taking action a at state S , $\gamma \in [0, 1]$ is the discount factor, and γ reflects the importance given to long-term rewards. We model the interaction process between the self-driving car and the environment as a Markov decision process.

State space S : The state is the environmental information perceived by the RL agent, and we use the image taken by the camera in front of the vehicle as the state, which is similar to the human driver's perspective. The image size is 80×80 pixels.

Action space A : Action space is the set of all actions that can be selected by the RL agent, and the action space of DQN is discrete. The most basic action space in autonomous driving contains 5 actions: accelerate, decelerate, turn left, turn right, and maintain. To achieve more accurate speed control, we define the action space as follows:

1. Change lane to the left;
2. Change lane to the right;
3. Maintain;
4. Slow acceleration;
5. Medium acceleration;
6. Fast acceleration;
7. Slow deceleration;
8. Medium deceleration;
9. Fast deceleration.

Reward function R : At each step, the RL agent selects an action, and the environment gives the corresponding feedback. The details of the reward function will be designed later.

3.2 Deep Q-Network

In this subsection, the architecture of deep Q-network and several data preprocessing methods are presented.

3.2.1 Data Preprocessing

To make the RL agent learn better, the received image data is pre-processed first.

1. Image size conversion: The original input image is transformed to a uniform size of 80×80 pixels.
2. Grayscale: The original RGB image with red, green, and blue channels is converted to the grayscale image to reduce the data dimension.
3. Image normalization: The gray value of the original image ranges from 0 to 255. To facilitate neural network learning, it is normalized to $[-1, 1]$. This is shown in equation (1):

$$x_{normalize} = \frac{x_{image} - \frac{255}{2}}{\frac{255}{2}} \quad (1)$$

where x_{image} is the original gray value of the pixel and $x_{normalize}$ is the gray value of the pixel after normalization.

4. Multi-frame image stacking: Since a single frame does not contain any motion information, the motion trend can only be presented by successive multi-frame images. Obviously, the motion information is crucial to autonomous driving, so we stack the nearest 4 frames as the input to the DQN. That is, the dimension of input data is $80 \times 80 \times 4$.

3.2.2 Q-network architecture

The architecture of the Q-network is shown in Fig. 1. The input of the Q-network is images of size $80 \times 80 \times 4$. The network is constructed by three convolutional layers and two fully connected layers, where the convolutional layers extract the image features and the fully connected layers output the Q values of each of the nine actions in the action space.

4 Multi-objective reinforcement learning approach

In this section, a multi-objective reward function is proposed to implement multi-objective reinforcement learning. The reward function is crucial for reinforcement learning models. By designing the reward function, we are able to guide the learning of the RL agent. We have two optimization objectives: to improve the speed of the vehicle and to improve the comfort of the passengers. Therefore, the reward function for speed optimization and the reward function for comfort optimization are designed, respectively. In addition, we also use a reward function to improve the learning efficiency of RL agents.

4.1 Reward function for speed optimization

In order to optimize the speed of the agent vehicle, higher speeds should be rewarded and lower speeds should be punished. Combined with the experimental tests, we design the following reward function architecture.

Speed reward: Speed reward is provided based on the speed of the vehicle, which is calculated as follows:

$$r_{speed}(v) = \frac{v - 35}{\max(v, 50) - 35} \quad (2)$$

where v is the current speed of the vehicle (km/h). The purpose of the speed reward function is to optimize speed by considering three aspects: 1) penalizing too low speed; 2) encouraging high speed; 3) the reward value should not grow infinitely. The maximum value of the speed reward is 1. When the vehicle speed is less than 35km/h, r_{speed} is negative, forcing the RL agent to increase the speed. When the vehicle speed is greater than 35km/h and less than 50 km/h, r_{speed} increases linearly with the vehicle speed; when the vehicle speed is greater than 50km/h, r_{speed} equals 1 and does not increase anymore to avoid too high speed.

Fig. 1 The architecture of the Q-network

Overtaking reward: The purpose is to encourage the RL agent to overtake and avoid the situation where the RL agent follows the previous car at a lower speed for a long time. For each overtaking, the RL agent receives a reward with a value of 0.3. The overtaking reward $r_{overtake}$ is calculated as follows:

$$r_{overtake} = \begin{cases} 0.3, & \text{if agent overtakes a vehicle} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The purpose of the speed reward and the overtaking reward is to increase the speed of the vehicle, and the two are added together to form the reward function for speed optimization:

$$r_{optimize_speed} = r_{speed} + r_{overtake} \quad (4)$$

4.2 Reward function for comfort optimization

There are many factors that affect passenger comfort, such as temperature, noise, vehicle acceleration, and vehicle vibration. However, most of the above factors depend on the environment and only the vehicle acceleration is controllable. Therefore, we consider two main aspects: the longitudinal acceleration of the vehicle, and the lane change frequency of the vehicle. Vehicle acceleration has a significant impact on passenger comfort. Driving behaviors such as rapid acceleration, rapid brake, and frequent lane changes can lead to a decrease in comfort. Combined with the experiment, the reward function for comfort optimization is designed as follows.

Lane change reward $r_{lanechange}$, which gives a negative reward for lane change actions:

$$r_{lanechange} = \begin{cases} -0.7, & \text{if agent changes lane} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

The speed change reward $r_{speedchange}$, which gives a reward according to the acceleration or deceleration action chosen by the RL agent, aims to avoid the RL agent to change speed frequently, or taking too high acceleration. $r_{speedchange}$ is calculated as follows:

$$r_{speedchange} = \begin{cases} -0.05, & \text{slow speed change} \\ -0.3, & \text{medium speed change} \\ -0.5, & \text{fast speed change} \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

The collision reward $r_{collision}$, which gives a reward based on whether the RL agent crashes or not. Collision events must be avoided, both for comfort and safety reasons. Due to the high cost of a collision, the value of the collision reward is high:

$$r_{collision} = \begin{cases} -10, & \text{if agent colides} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

The purpose of the above lane change reward, speed change reward, and collision reward is to improve passenger comfort, and we add them together to form the reward function for comfort optimization:

$$r_{optimize_comfort} = r_{speedchange} + r_{lanechange} + r_{collision} \quad (8)$$

4.3 Reward function for learning assistance

Reinforcement learning agents generally start learning from nothing, which requires a lot of interaction with the environment. Therefore, Reward functions $r_{maintain}$ and $r_{avoidance}$ are designed to improve learning efficiency.

$r_{maintain}$: Since the probability of RL agents exploring the “maintain” action in the early stage is small, RL agents rarely choose the “maintain” action. Therefore, we set a reward $r_{maintain}$: if the vehicle speed is higher than 50km/h, the “maintain” action will be rewarded with +0.1.

$r_{avoidance}$: The simulator can provide the distance between the vehicle and the surrounding obstacle. Using this knowledge, we set the reward $r_{avoidance}$: if there is an obstacle ahead, then taking a lane change or deceleration action will receive a +0.2 reward.

We add the two rewards together to form r_{assist} , which guides the exploration of RL agents:

$$r_{assist} = r_{maintain} + r_{avoidance} \quad (9)$$

Meanwhile, several rules are established to shrink the action space and help RL agent learn faster.

1. If the vehicle is currently in the leftmost (or rightmost) lane, changing lanes to the left (or right) is not allowed.
2. Lane-keeping: If the vehicle is not in the center of the lane, move the vehicle to the center of the lane. This rule is implemented by the simulator [27], so the RL agent only needs to select the lane, not the specific steering angle.

4.4 Multi-objective reward function

The final multi-objective reward function is constructed as follows:

$$r_{multi_objective} = r_{optimize_speed} + r_{optimize_comfort} + r_{assist} \quad (10)$$

In the above multi-objective reward function, both the reward for speed maximization and the punishment for comfort reduction are included. Clearly, these two objectives are in conflict. On the one hand, to maximize speed, the vehicle must increase its speed as soon as the situation allows and avoid other slower vehicles by changing lanes frequently, which may also require frequent hard braking. On the other hand, to maximize comfort, the vehicle is required to drive as smoothly as possible. Therefore, the purpose of designing a multi-objective reward function is to find a more reasonable driving strategy by seeking a balance between efficiency and comfort.

5 Experiments

In this section, the effectiveness of the proposed approach is verified by simulations. Some implementation details of the experiments are presented and the results of the experiments are discussed.

5.1 Environment simulator

For cost and feasibility reasons, we experiment in a simulator. We use an open-source simulator developed by Min et al. based on the Unity ML module [27], and modify it as needed. The simulator is able to simulate the driving environment and vehicle sensors. The driving environment is random and dynamic, including a five-lane urban expressway, other vehicles on the road, and background elements such as sky and buildings. Other vehicles on the road are randomly generated and take random actions with a certain probability. Meanwhile, roads and buildings are constantly changing. Fig. 2 shows the simulator interface.

5.2 Details

1. Exploration: We use the ϵ – greedy exploration policy. The RL agent does not always pick the action with the largest Q value during learning, but chooses a random action with probability ϵ . During the learning process, the probability of taking a random action is gradually reduced.
2. Experience Replay [7]: There is a strong correlation between the data obtained from the RL agent's interaction with the environment. To break this correlation, the DQN algorithm stores the data sampled by the RL agent and randomly selects samples from it during training.
3. Episode termination: If a vehicle collides or the vehicle travels more than a certain distance, the current episode is terminated and the training of the next episode begins.
4. Pre-exploration: Before starting to train the neural network, the RL agent performs a certain number of steps of exploration, interacts with the environment to obtain data samples, and stores these samples in the "experience

Fig. 2 A screenshot of the simulator interface

replay". The purpose of pre-exploration is to ensure that there are enough samples in the experience replay pool for sampling.

5. Hyperparameter: Table 1 shows the hyperparameter settings of the algorithm.

5.3 Results

We trained the RL agent for one million steps, running on an NVIDIA RTX 3070 GPU, which took about 8 to 10 hours.

Fig. 3 shows the variation of the reward value during the training. In the early stage, the reward obtained by the RL agent is low, because the RL agent performs a lot of random exploration at this time. The reward tends to increase and eventually converge to the highest reward value that can be obtained. This is because the RL agent receives a

Table 1 Hyperparameter setting of DQN algorithm

Hyperparameter	Value	Hyperparameter	Value
Maximum training steps	1000000	Learning Rate	0.00025
Experience replay size	100000	Batch size	32
Pre-exploration steps	50000	Discount Factor γ	0.99
Target network update cycle	10000	Initial value of ϵ	1
Network input size	$80 \times 80 \times 4$	Final value of ϵ	0.1

Fig. 3 The variation of the reward value during the training

higher speed reward (the maximum speed reward is 1) in the later stages and rarely receives a negative reward. This indicates that the DQN algorithm converges successfully. In Section 4.3, several rules are used to assist in reinforcement learning. Without the help of these rules, in the early stage of training, the RL agent takes random actions and the reward value should fluctuate around 0. Adding these rules can avoid the RL agent to choose some wrong actions and improve the learning efficiency. According to Fig. 4, it can be seen that the reward values are mostly positive at the early stage of training, which proves that the rules used are effective.

To demonstrate the advantages of the proposed multi-objective driving strategy, we trained a single-objective driving strategy optimized for speed only, referring to the reward function used by Min et al. in his study [27]. The reward function for the single-objective strategy is as follows:

$$r'_{speed}(v) = \frac{v - 40}{v_{max} - 40} \quad (11)$$

$$r'_{overtake} = \begin{cases} 0.5, & \text{if agent overtakes a vehicle} \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

$$r'_{collision} = \begin{cases} -10, & \text{if agent colides} \\ 0, & \text{otherwise} \end{cases} \quad (13)$$

$$r_{single_objective} = r'_{speed} + r'_{overtake} + r'_{collision} \quad (14)$$

We used the same hyperparameters as in Table 1. We tested each of the two strategies for 50,000 steps. In the tests, the RL agent picks the actions exactly according to the maximum Q value. Fig. 4 shows the speed of the vehicle under both strategies. Fig. 5 shows the acceleration of the vehicle under both strategies. We calculated the total number of lane changes for each episode during the test, as shown in Fig. 6.

Table 2 and Table 3 show the statistics about speed and acceleration, respectively. The statistics are calculated based on the results of the 50,000-step test. We used the root mean square value (RMS) of acceleration to evaluate passenger comfort, referring to the method of the international standard ISO 2631-1 [30]. A qualitative evaluation method for passenger comfort is also given in ISO 2631-1 [30], as shown in Table 4. According to Table 4, the comfort level of the proposed multi-objective driving strategy is evaluated as “a little uncomfortable” and the RMS value is very close to “not uncomfortable”. Meanwhile, the comfort rating of the single-objective strategy is between “uncomfortable” and “very uncomfortable”.

Fig. 4 The figure shows the speed of the vehicle under both strategies. The data shown here is from one episode of the test, each data point in the graph represents the average speed of 50 steps

Fig. 5 The figure shows the acceleration of the vehicle under both strategies. The data shown here is from one episode of the test, the acceleration is computed by speed

Fig. 6 The total number of lane changes for each episode. The statistics were computed by running a greedy policy for 50000 steps

Table 2 The table shows the speed statistics for both strategies. The statistics were computed by running a greedy policy for 50000 steps

Multi-objective strategy	Statistics	Single-objective strategy	Statistics
Average speed	53.5 km/h	Average speed	62.6 km/h
Maximum speed	60.0 km/h	Maximum speed	79.9 km/h
Standard deviation	5.49	Standard deviation	11.60

Table 3 The table shows the acceleration statistics for both strategies

Multi-objective strategy	Statistics	Single-objective strategy	Statistics
Average acceleration	0.09 m/s ²	Average acceleration	0.86 m/s ²
Maximum acceleration	2.83 m/s ²	Maximum acceleration	4.88 m/s ²
Root mean square value	0.32	Root mean square value	1.31

The statistics were computed by running a greedy policy for 50000 steps

Table 4 The table shows a qualitative evaluation method for passenger comfort, from ISO 2631-1 [30]

RMS	Comfort level
< 0.315	Not uncomfortable
0.315 ~ 0.63	A little uncomfortable
0.5 ~ 1.0	Fairly uncomfortable
0.8 ~ 1.6	Uncomfortable
1.25 ~ 2.5	Very uncomfortable
> 2	Extremely uncomfortable

This method gives a rough evaluation based on the root mean square (RMS) of the acceleration

Comparing these statistics, the maximum speed of the vehicle with the single-objective strategy reached 80 km/h, but the speed fluctuated dramatically; the maximum speed of the vehicle with the multi-objective strategy was around 60 km/h, but the speed changed smoothly. Although the maximum speed achieved by the single-objective driving strategy is much higher, the average speed difference between the two strategies is not much. The reason for this phenomenon is the presence of many other vehicles on the road. If the vehicle's speed is too high, it must brake and change lanes frequently, which not only leads to a decrease in comfort, but also to a loss of average speed. Conversely, if the vehicle maintains an appropriate speed, it is possible to strike a balance between comfort and average speed. The single-objective strategy tends to take higher speed as possible, which leads to frequent lane changes and frequent emergency braking, resulting in dramatic speed fluctuations and failure to have a significant increase in average speed. This indicates that the single-objective strategy is poorly comfortable and does not offer great advantages in terms of transport efficiency. The multi-objective driving strategy is: not to blindly increase speed even if conditions permit, but to maintain a more reasonable speed level. In this strategy, although the maximum speed of the vehicle is lower, the need for emergency braking and lane changes is rare because the speed difference between the RL agent vehicle and the surrounding vehicles is small. Also, the average speed can reach a high level due to the more stable speed. Our multi-objective strategy not only provides higher comfort but also ensures sufficient average speed.

6 Conclusion

A multi-objective reinforcement learning approach has been proposed, with the purpose of learning a multi-objective autonomous driving strategy. A multi-objective reward function has been constructed to optimize the two objectives of vehicle speed and passenger comfort. Experiments were performed in a simulator. Compared to the single-objective strategy, the multi-objective driving strategy achieves much more gentle speed changes and very few lane changes with sacrificing average speed slightly. The results showed that the proposed approach can strike a good balance between transportation efficiency and passenger comfort, which is more in accordance with the practical requirements.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China under Grants 62022094 and 61873350, the Innovation-Driven Project of Central South University, China under Grant 2020CX032, and the Hunan Provincial Natural Science Foundation of China under Grants 2020JJ2049 and 2019JJ30032.

Authors' contribution TH: Visualization, Writing - original draft, Methodology. BL: Conceptualization, Funding acquisition, Supervision. CY: Writing -review & editing. All authors read and approved the final manuscript.

Declarations

Competing interests The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. González D, Pérez J, Milanés V, Nashashibi F. A review of motion planning techniques for automated vehicles. *IEEE Trans Intell Transp Syst*. 2015;17(4):1135–45.
2. Paden B, Čáp M, Yong SZ, Yershov D, Frazzoli E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans Intell Veh*. 2016;1(1):33–55.
3. Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, Zhang J, et al. End to end learning for self-driving cars. 2016. arXiv preprint [arXiv:160407316](https://arxiv.org/abs/1604.07316)
4. Chen C, Seff A, Kornhauser A, Xiao J. Deepdriving: Learning affordance for direct perception in autonomous driving. In: *Proceedings of the IEEE international conference on computer vision*. 2015. p. 2722–2730.
5. Xu H, Gao Y, Yu F, Darrell T. End-to-end learning of driving models from large-scale video datasets. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. p. 2174–2182.
6. Sallab AE, Abdou M, Perot E, Yogamani S. End-to-end deep reinforcement learning for lane keeping assist. 2016. arXiv preprint [arXiv:161204340](https://arxiv.org/abs/1612.04340)
7. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, et al. Human-level control through deep reinforcement learning. *Nature*. 2015;518(7540):529–33.
8. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. Playing Atari with deep reinforcement learning. 2013. arXiv preprint [arXiv:13125602](https://arxiv.org/abs/1312.5602)
9. Bellman R. Dynamic programming and Lagrange multipliers. *Proc Natl Acad Sci USA*. 1956;42(10):767.
10. Werbos P. Advanced forecasting methods for global crisis warning and models of intelligence. *General Syst Yearb*. 1977;22:25–38.
11. Watkins CJ, Dayan P. Q-Learning. *Mach Learn*. 1992;8(3–4):279–92.
12. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: *International conference on machine learning*. 2016. p. 1995–2003.
13. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: *International conference on learning representations (Poster)*. 2016.
14. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K. Asynchronous methods for deep reinforcement learning. In: *International conference on machine learning*. 2016. p. 1928–1937.
15. Schulman J, Levine S, Abbeel P, Jordan M, Moritz P. Trust region policy optimization. In: *International conference on machine learning*. 2015. p. 1889–1897.
16. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017. arXiv preprint [arXiv:170706347](https://arxiv.org/abs/1707.06347)
17. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*. 2016;529(7587):484–9.
18. Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, et al. Mastering the game of Go without human knowledge. *Nature*. 2017;550(7676):354–9.
19. Finn C, Levine S. Deep visual foresight for planning robot motion. In: *2017 IEEE international conference on robotics and automation*. 2017. p. 2786–2793.
20. Zheng G, Zhang F, Zheng Z, Xiang Y, Yuan NJ, Xie X, Li Z. Drn: A deep reinforcement learning framework for news recommendation. In: *Proceedings of the 2018 world wide web conference*. 2018. p. 167–176.
21. Hu Y, Da Q, Zeng A, Yu Y, Xu Y. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018. p. 368–377.
22. le E, Jain V, Wang J, Narvekar S, Agarwal R, Wu R, Cheng HT, Lustman M, Gatto V, Covington P, et al. Reinforcement learning for slate-based recommender systems: a tractable decomposition and practical methodology. 2019. arXiv preprint [arXiv:190512767](https://arxiv.org/abs/1905.12767)
23. Chae H, Kang CM, Kim B, Kim J, Chung CC, Choi JW. Autonomous braking system via deep reinforcement learning. In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems*. 2017;1–6.
24. Wolf P, Hubschneider C, Weber M, Bauer A, Härtl J, Dürr F, Zöllner JM. Learning how to drive in a real world simulation with Deep Q-Networks. In: *2017 IEEE intelligent vehicles symposium*. 2017. p. 244–250.
25. Chen J, Yuan B, Tomizuka M. Model-free deep reinforcement learning for urban autonomous driving. In: *2019 IEEE intelligent transportation systems conference*. 2019. p. 2765–2771.
26. Kendall A, Hawke J, Janz D, Mazur P, Reda D, Allen JM, Lam VD, Bewley A, Shah A. Learning to drive in a day. In: *2019 international conference on robotics and automation, IEEE*; 2019. p. 8248–8254.
27. Min K, Kim H, Huh K. Deep Q learning based high level driving policy determination. In: *2018 IEEE intelligent vehicles symposium*. 2018. p. 226–231.
28. Li C, Czarnecki K. Urban driving with multi-objective deep reinforcement learning. In: *Proceedings of the 18th international conference on autonomous agents and multiAgent systems*. 2019. p. 359–367.
29. Bellman R. A markovian decision process. *J Math Mech*. 1957;6(5):679–84.
30. International Organization for Standardization. Mechanical vibration and shock—evaluation of human exposure to whole-body vibration—part 1: general requirements, ISO 2631-1:1997 edn. Geneva: International Organization for Standardization; 1997.