Autonomous Driving System based on Deep Q Learnig

Takafumi Okuyama, Tad Gonsalves
Department of Information and Communication Sciences,
Faculty of Science & Technology,
Sophia University, Tokyo, Japan.
e-mail: okuyama1625@gmail.com,
t-gonsal@sophia.ac.jp

Jaychand Upadhay
Department of Information TechnologyXavier Institute of Engineering, Mahim, Mumbai, Maharashtra, India.
e-mail: jaychand.u@xavierengg.com

***Abstract*—This paper deals with the simulation results of an autonomous car learning to drive in a simplified environment containing only lane markings and static obstacles. Learning is performed using the Deep Q Network. For a given input image of the street captured by the car front camera, the Deep Q Network computes the Q values (rewards) corresponding to the actions available to the autonomous driving car. These actions are discrete angles through which the car can steer for a fixed speed. The autonomous driving system in the car enforces the action that has the highest reward. Our simulation results show high accuracy in learning to drive by observing the lanes and bypassing obstacles.**

Keywords – autonomous driving vehicles; self-driving cars; reincorced learning: deep learning; deep Q network

I. INTRODUCTION

The application of Artificial Intelligence (AI) and Machine Learning (ML) techniques to the development of autonomous driving systems is currently a hotbed of research. In Japan, the government has set aside enormous funds to make autonomous driving technology a reality for the 2020 Olympics, because it is considered safe and efficient mode of transportation. Although a fully functional self-driving car is still a long way off, most major automobile manufacturers worldwide have reached advanced stages of developing self-driving cars. However, the current autonomous driving is very expensive as it tends to rely on highly specialized infrastructure and equipment like Light Detection and Ranging (LIDAR) for navigation, Global Positioning GPS) for localization and Laser Range Finder (LRF) for obstacle detection [1]. The focus of our study is to learn to drive by detecting road features relying solely on single camera vision.

Recently, Deep Learning [3], [6], [11] has shown great success in supervised learning in diverse areas such as image recognition [20]), speech recognition [16], sentiment analysis [4], game playing [15], and robot navigation [9]. As opposed to supervised learning where there exists an output (supervisor) for a given input, reinforced learning [5], [14], [17] is used when an agent has to learn to take the right action to affect a change in the environment in which it is placed. The right action of the agent is *reinforced* with a reward. Deep Reinforced Learning (DRL) combines reinforced learning with deep learning [2]. In practice, machine learning with images as inputs is achieved by Convolutional Neural Networks (CNN), which are modelled on the mammalian visual cortex processing [7], [8]. *AlexNet* [10] and *GoogLeNet* [18] CNN models have become renowned for their exceptional performance in image recognition.

Deep Q Network (DQN) is a form of reinforced learning in which the output of a CNN is not classification, but Q values (rewards) given to actions resulting from the input states. The DQN agent learns successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning. Recently, DQN has proved successful in the challenging domain of classic Atari 2600 games [13]. A huge amount of labeled data of human driving is necessary to train any autonomous deriving system. DQN learning enables the agent to learn from its own behavior instead of labeled data. After several hours of training, it learns to steer smoothly in between the lanes and avoid obstacles aided only by the visual information input.

This paper is organized as follows: Section II explains the DQL theory and Section III presents the CNN and DQN structures. The simulation results are presented in Section IV.

II. DEEP Q LEARNING

Q learning is variation of reinforcement learning. In Q learning, there is an agent having states and corresponding actions. At any moment, the agent is in some feasible state. In the next time step, the state is transformed to other state(s) by performing some action. This action is accompanied either by reward or a punishment. The goal of the agent is to maximize the reward gain. The Q learning algorithm is represented by the following update formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t - Q(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \right) \quad (1)$$

where $Q(s_t, a_t)$ represents the Q value of the agent in the state s_t, and action a_t at time t, rewarded with reward r_t. α is the learning rate and γ is the discount factor. The γ parameter is in the range [0,1]. If γ is closer to 0, the agent will tend to consider only immediate rewards. On the other hand, if it is closer to 1, the agent will consider future rewards with greater weight, thus willing to delay the reward. The Learning rate and Discount factor, described below, are the two most crucial parameters influencing the performances of the Q learning algorithm.

978-1-5386-6331-8/18/$31.00 ©2018 IEEE

A. *Learning Rate*

The learning rate determines the strength with which the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 will make the agent consider only the most recent information. In fully deterministic environments, a learning rate is optimal. When the problem is stochastic, the algorithm still converges under some technical conditions on the learning rate, that require it to decrease to zero. In practice, often a constant learning rate is used.

B. *Discount Factor*

The discount factor determines the importance of future rewards. A factor of 0 will make the agent short-sighted by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. If the discount factor meets or exceeds 1, the action values may diverge. Even with a discount factor only slightly lower than 1, the Q learning leads to propagation of errors and instabilities when the value function is approximated with an artificial neural network. In that case, it is known that starting with a lower discount factor and increasing it towards its final value yields accelerated learning.

III. CNN OVERVIEW

A. *Convolutional Neural Network (CNN)*

Neural Networks, in general, imitate the function of biological neurons whose excitation signals can be on/off depending on the strength of the input stimulus. The well-known Multi-Layer Feedforward Neural Network which includes the combination matrix operation and nonlinear activation function performs quite well in solving classification problems. For example, the error in classifying the 0-9 digit images in the MINST dataset is as low as 3% . However, in the classification of the CIFAR-10 dataset which has ten classes of images such as dogs, birds and so on, the test error increases to about 50%. The poor performance of MLPs in classifying images has given rise to a new kind of a Neural Network, called Convolutional Neural Nets (CNN). A CNN, for example, with three convolution layers interlinked with max pooling layer showed only16.6% test error in classifying the images in the CIFAR-10 dataset [8].

B. *Deep Q Network (DQN)*

CNN is essentially a classification structure for classifying images into labelled classes. The various layers of the CNN extract image features and finally learn to classify the images. Hence, the outputs of a typical CNN represent the classes or the labels of the classes, the CNN has learnt to classify (Fig. 1). A Deep Q Network is a variation of CNN. The outputs are not classes, but the Q values (or rewards) corresponding to each of the actions the agent has learnt to take in response to its state in the environment (Fig. 2).

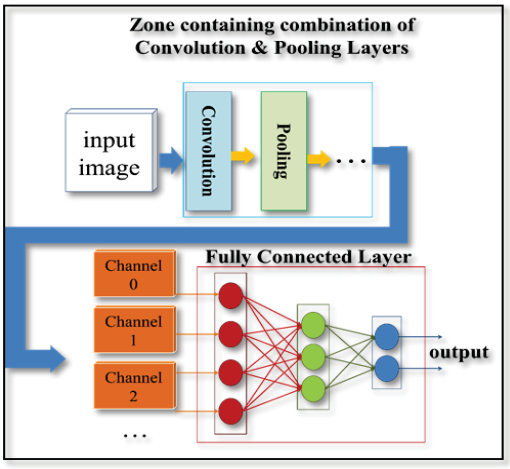

Figure 1. Classifying images using CNN

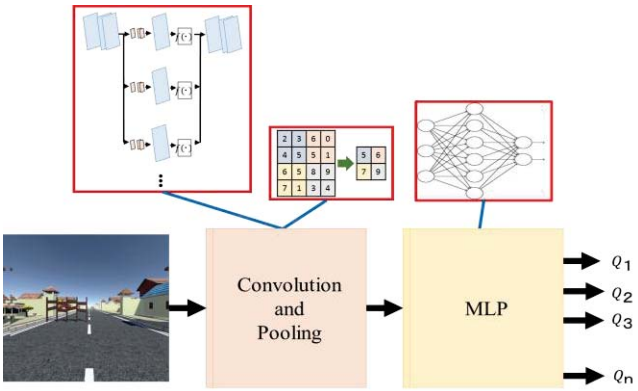

Figure 2. Deep Q network

In our model, the input to the DQN is the image of the street the car sees in front of it at a given point of time.

C. *Deep Q Network (DQN)*

The Deep Q Network computes the Q value (reward) for a given state of the environment using the following algorithm (Fig. 3) [19].

```
Initialize all Q values;
Repeat: (for each episode)
Choose a random state s;
WHILE (true) Do (Repeat for each step in the episode)
        Select action a∈ A(s) according to the policy
        execute the action, a;
        Observe new state, s';
        Receive immediate reward, r;
        Update Q(s, a) using Equation (1);
Until s is one of the goal states, Go to Repeat;
Until the desired number of episodes has been investigated;
Stop.
```

Figure 3. Deep Q learning algorithm

IV. EXPERMINETAL RESULTS

A. Learning Environment

Fig. 4 and Fig. 5 show the learning environment created in the simulation study. On a straight road bounded by footpaths on sides, obstacles are placed in random positions every 30 meters.

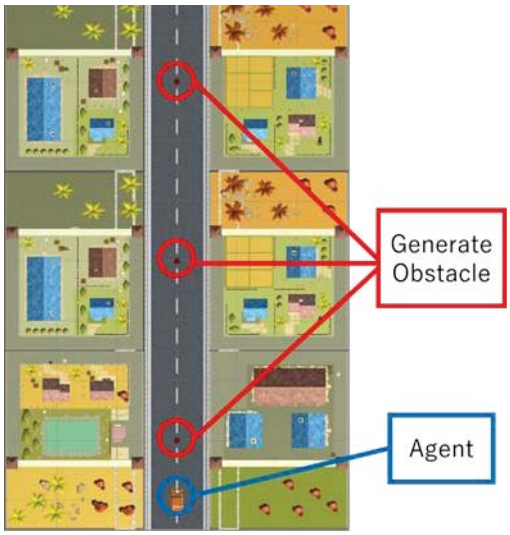

Figure 4. Obstacle locations on street

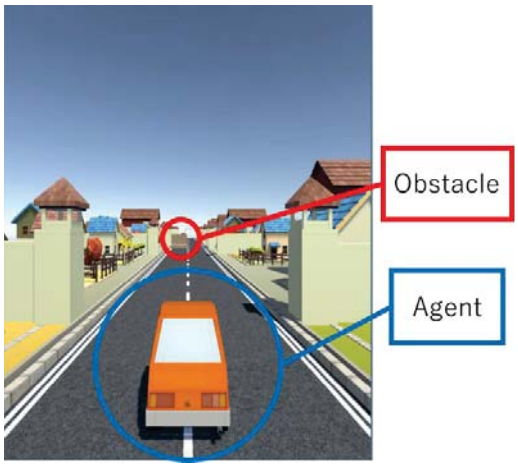

Figure 5. Generating obstacles

The obstacles are of the following types: human figures (Fig. 6a), vehicles (Fig. 6b), and blocks (Fig. 6c). These static obstacles are randomly generated from a pre-fabricated pool.

Figure 6a. Human widgets

Figure 6b. Vehicle widgets

Figure 6c. Block widgets

The simulation environment is created using Unity (https://unity3d.com/) . The interaction between Unity and Python program is provided by the Unity-Python conversion modules (http://ailab.dwango.co.jp/en/). Learning is implemented using the chainer framework (http://chainer.org/). The learning program is executed on a workstation with the following specifications: 8-core Xenon CPU x 2, 256 GB RAM and Nvidia P100 GPU x 5.

B. Agent Action

Figure 7. Action plan

The DQN, interprets the road scene in front of the car from the camera input and suggests a specific plan of action for that given time slot. To simplify the action plan, we consider only three kinds of actions: steer 10 degrees to the left, keep straight, steer 10 degrees to the right (Fig. 7).

C. Results of Learning

The state-action cycle is repeated till the agent reaches the goal. If in the state-action cycle, the agent ends up hitting an obstacle or crossing one of the bounding lanes, the learning episode is discontinued, and the agent begins a new and fresh episode. If it continues on the course successfully, it is rewarded for every obstacle it overcomes. The agent updates its information on the trajectory every 0.15 seconds, sends it to the learning program and takes the appropriate action dictated by the learning program. We have trained the

agent to avoid obstacles when driving at a constant speed of 10 m/sec, 15 m/sec and 20 m/sec. The highest scores achieved while learning for each of these speeds are shown in Fig. 8 and the longest distances covered by the learning agent for each of the speeds without hitting any of the lanes or obstacles are shown in Table I. As expected, the learning ability of the agent decreases with increasing driving speeds.

TABLE I. SUCCESSFUL LEARNING

Experiment	Car speed (meters/sec)	Distance covered (meters)
1	10	10,077
2	20	4,907
3	30	2,173

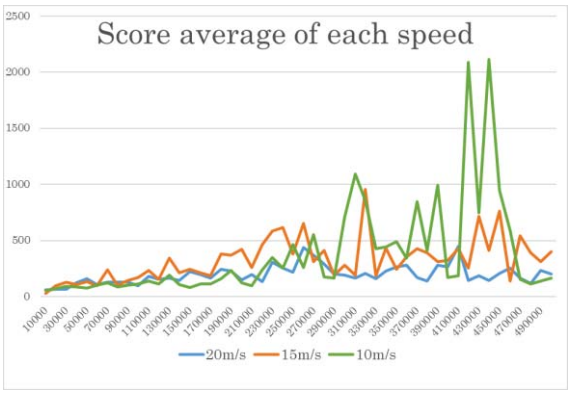

Figure 8. Score average of three different speeds

V. CONCLUSION

In this paper, we proposed a simulation study of an autonomous agent learning to drive in a simplified environment consisting of only lane markings and static obstacles. We used a Deep Q Network to train the agent in the simulated environment. Future prospects of our study include verifying the simulation results and further tuning them using Robocar (fully functional driving car, 1/10th the size of standard commercial car, (Fig. 9) driving on the laboratory floor (Fig. 10). The verification results are expected to demonstrate that learning autonomous driving in a simulated environment is a step towards driving on real streets.

Figure 9. RoboCar 1/10 (ZMP, Japan)

Figure 10. Learning verification on laboratory floor

REFERENCES

[1] M. Aly, "Real time detection of lane markers in urban streets," 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, 2008, pp. 7-12, doi: 10.1109/IVS.2008.4621152 8.

[2] K. Arulkumaran, N. Dilokthanakul, M. Shanahan, and A. A. Bharath, "Classifying options for deep reinforcement learning," in Proc. IJCAI Workshop Deep Reinforcement Learning: Frontiers and Challenges, 2016.

[3] Y. Bengio, Learning deep architectures for AI. Foundations and Trends in Machine Learning 2, 1–127 (2009).

[4] M. Y. Day and Y. D. Lin, "Deep Learning for Sentiment Analysis on Google Play Consumer Review," 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, USA, 2017, pp. 382-388, doi: 10.1109/IRI.2017.

[5] C. Diuk, A. Cohen, & M. L. Littman, An object-oriented representation for efficient reinforcement learning. Proc. Int. Conf. Mach. Learn. 240–247 (2008).

[6] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau,and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," Nature, vol. 542, no. 7639, pp. 115–118, 2017.

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, & R. R. Salakhutdinov, (2012). Improving neural networks by preventing co-adaptation of feature detectors. (https://arxiv.org/pdf/1207.0580.pdf accessed on Aug. 20, 2017).

[8] Y. Jia, E. Shelhamer, J. Donahue, et al., "Caffe: convolutional architecture for fast feature embedding," arXiv preprint arXiv:1408.5093, 2014.

[9] K. M. I. Khalilullah, S. Ota, T. Yasuda and M. Jindai, "Development of robot navigation method based on single camera vision using deep learning," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 2017, pp. 939-942, doi: 10.23919/SICE.2017.8105675

[10] A. Krizhevsky, I. Sutskever, & G. E. Hinton, (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25 (NIPS 2012). pp. 1106–1114.

[11] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[12] Y. Li. (2017). Deep reinforcement learning: An overview. arXiv. [Online]. Available: https://arxiv.org/abs/1701.07274

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

[14] M. Riedmiller, T. Gabel, R. Hafner, & S. Lange, Reinforcement learning for robot soccer. Auton. Robots 27, 55–73 (2009).

[15] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre et al., “Mastering the game of go with deep neural networks and tree search,” Nature, vol.529, no. 7587, pp. 484–489, 2016.

[16] C. Spille, S.D. Ewert, B. Kollmeier, B.Meyer, Predicting speech intelligibility with deep neural networks, Computer Speech & Language 48(2018) 51-66.

[17] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. Cambridge, MA: MIT Press, 1998.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, & A. Rabinovich. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9.

[19] B. Wang, J. w. Li and H. Liu, "A Heuristic Reinforcement Learning for Robot Approaching Objects," 2006 IEEE Conference on Robotics, Automation and Mechatronics, Bangkok, 2006, pp. 1-5, doi: 10.1109/RAMECH.2006.252749

[20] X. Zhang et al., "Classification of mammographic masses by deep learning," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 2017, pp. 793-796, doi: 10.23919/SICE.2017.8105545