C-07

Visualizing Performance of Reinforcement Learning Algorithms on a Simulator

Menglei Zhang†　　Yukikazu Nakamoto†

1. Introduction

Reinforcement learnings become to play key roles in embedded systems and auto driving systems are promising applications. Reinforcement learning have a lot of algorithms with various properties. We need to have criteria to choose a reinforcement learning algorithm in a specific application and consider obtain information on the purpose with visualization of the reinforcement learning algorithm. In this paper, we have developed visualization of the reinforcement learning algorithm, deep q-network (DQN), as the first step.

2. Implementation of Reinforcement Learning

In this paper, We use Pytorch to build a neural network model with multiple Convolutional layers, and use the OpenAI Gym environment to train our model, and use the Grad-CAM method to visualize the features captured by the Convolutional neural network in the form of heat maps, in a way that is easier for humans to understand Represents the features captured by the Convolutional neural network from the environment.

2.1 CarRacing-v0

The CarRacing-v0 environment[2] is a very simple problem similar to a tracking car, as shown in Fig.1 below. Our goal is using the reinforcement learning to control the car to run on the track and speed as fast as possible. This environment gives a simple continuous control task, which requires us to learn from the pixels obtained from the bird view, The CarRacing-v0 environment will return the screenshot of each frame as a state, the pixel is 96*96, so the state is a 96*96 image. Actions in the CarRacing-v0 environment include steer, gas, brake. We can use OpenAI Gym's Python API to control the movement of the car. When we train the neural network, we can give the environment state takes another downsampling to reduce the dimensionality of the neural network input. The reward given to the agent by the environment is defined to be divided into two parts:

(1) One is that as time passes, the price will always be paid, -0.1/frame, if it is calculated according to 30frams/s, it is -3/s;

(2) The vehicle track in the CarRacing-v0 environment is divided into many areas of the same width, which are called tiles. The second part is the passage the reward obtained by the tiles on the track will be awarded to the agent with a reward of 1000/N each time a tile passes, where N is the total number of tiles.

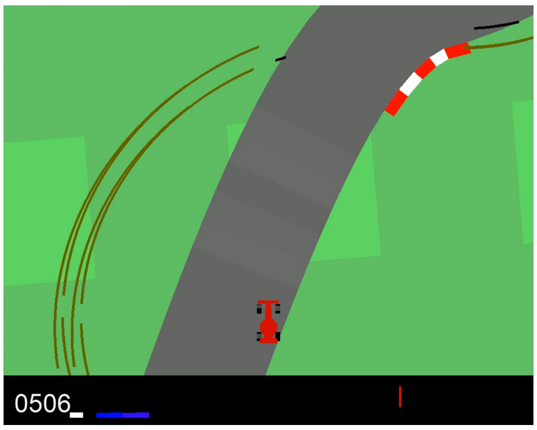

Figure 1　CarRacing-V0 environment

2.2 Deep Q-Learning

Deep Q-Learning is a variant of Q-Learning[1]. For Q-Learning, many real-world problems cannot be solved because many real-world problems are continuous, which means that their states are non-discrete. Q-Learning cannot create an infinite Q Table to record all Q values. Therefore, a method is needed to fit the calculation of the Q value. Because the characteristic of the neural network is that it can fit any equation in theory, the neural network can be used to complete this task well. This is the biggest difference between DQN and Q-Learning. The interaction process between our model and the OpenAI Gym environment can be simply illustrated by Fig.2.

2.2.1 Value Function Approximation

DQN is based on Q-Learning, an off-policy algorithm that learns better strategies by evaluating Q(s,a) and boosting strategies based on Q. In Q-Learning, a table is used to store all Q values, but this is hardly feasible in many real-world

†University of Hyogo

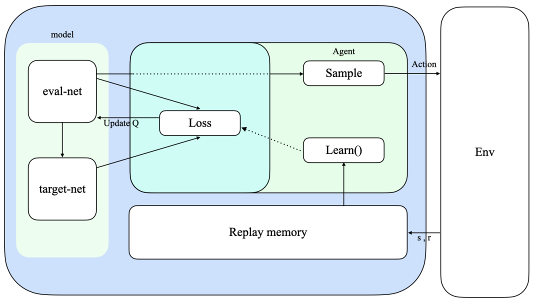

Figure 2 The structure of process

problems because there are simply too many states to store using a table. The feature of DQN is to replace Q(s,a) with a neural network, and no matter how many states there are, the final Q value can be output by matrix operation to reduce the dimensionality. This is Value Function Approximation

2.3 Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) is a method of visualizing the abstract features of Cnvolutional neural networks[2]. CAM is the class activation mapping. It is a picture of the same size as the original picture. The pixel value of each position on the picture ranges from 0 to 1, and is generally represented by a grayscale image from 0 to 255[4]. It can be understood as the contribution distribution to the prediction output. The higher the score, the higher the response of the original picture to the network and the greater the contribution.Convolutional neural networks are very effective in many tasks, but compared to traditional machine learning algorithms, such as decision trees or logistic regression, the content and rules they learn are difficult to present in a way that humans understand, so they are used by many people. Think of it as a "black box". Grad-CAM is a algorithm that can indicate which pixels in the image have a strong influence on the output of the Convolutional neural network[5]. In other words, you can know which positions the Convolutional neural network is sensitive to pixel values.

3. Experiments

(1) We created a DQN model and trained it in the following hardware environment.

OS	Windows10 19043.1110
CPU	E5 2670V3
GPU	RTX 3060
RAM	64GB
Pytorch	LTS 1.8.1

(2) We use Pytorch to build the neural network, Our model mainly uses 3-layer Convolutional Neural Network,The CarRacing-v0 environment will return screenshots of each step, and the neural network will learn the characteristics of each screenshot to control the movement of the vehicle.

4. Result and Discussion

(1) After training for 800 episodes, it took a total of 67 minutes and 32 seconds. As shown in Fig.3, the score of each game is gradually improving.

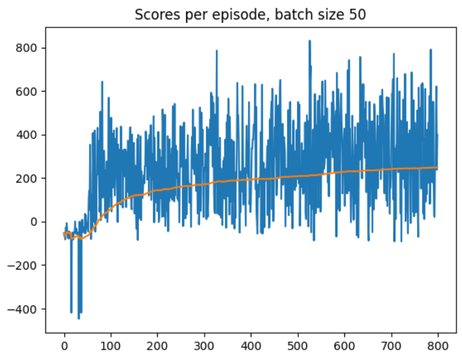

Figure 3 Scores per episode

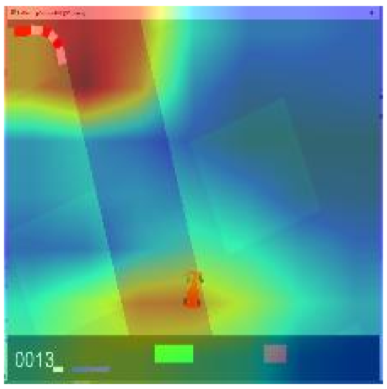

Figure 4 Heat map of the first frame

(2) We randomly selected a screenshot, and used Grad-CAM to visualize their features.As shown in Fig.4, it can be seen intuitively from the heat map that the Convolutional neural network is indeed sensitive to the front end of the runway and controls the movement of the car by paying attention to the runway ahead.

5. Concolusion

We evaluated the possibility of Convolutional neural network visualization through Grad-CAM in DQN. In the future, we will create more complex models and train the models on more realistic simulators. Grad-CAM can also be used in other reinforcement learning models with a neural network structure, and we are verifying this possibility. At the same time, we develop other visualization methods in the reinforcement learning algorithms except DQN.

References

[1] Sutton, R and Barto, A : REINFORCEMENT LEARNING, An Introduction, 2nd ed., Bradford Books (2018).

[2] OpenGym : CarRacing-v0, `https://gym.openai.com/envs/CarRacing-v0/`, access (2021-7-23)

[3] Selvaraju, R., Das, A.,Vedantam, R., Cogswel, M., Parikh, D., and Batra, D. : Grad-CAM: Why did you say that? Visual Explanations from Deep Networks, International Journal of Computer Vision, vol. 128, pp. 336–359 (2020).

[4] Gildenblat, J. and contributors : Class Activation Map methods implemented in Pytorch, GtiHub, `https://github.com/jacobgil/pytorch-grad-cam`, access (2021-2023).

[5] Kazuki Nagamine, Satoshi Endo, Koji Yamada, Naruaki Toma, Yuhei Akamine : Analysis of the Action Value of Deep Q Network by visualization, FIT2018 : F-042