ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 48-53
doi: 10.18178/wcse.2018.06.009

Research on DQN Algorithm with Step Control Strategy Applied in ATO

Yanmei Guo [1] [+], Ziheng Wu [1], Xiangxian Chen [1], and Zhujun Ling [2]

[1] Department of Instrumentation Science and Engineering, Zhejiang University, Hangzhou, China

[2] Zhejiang Train Intelligent Engineering Technology Research Center Co., Ltd

Abstract. Achieving good speed control is the key to the operation of automatic train operation system. In this paper, the image processing method in deep learning is introduced. Combined with classical DQN algorithm and step control strategy, a speed controller with good tracking effect is designed. This method enhances the control accuracy and introduces the step control strategy into the DQN algorithm. With the combination of neural network and reinforcement learning, a good train speed control effect is achieved.

Keywords: ATO control, reinforcement learning, DQN algorithm, step control.

1. Introduction

Speed Controlling is an important part of ATO. Its primary goal is to adjust the tractive force and braking force of train in real time according to the different operating environments of train so as to realize the safe, reliable and efficient operation of train in accordance with predetermined instructions. The construction idea of ATO system is generally composed of the steps of establishing train model, simulating and verifying, designing control algorithm according to requirements, etc. Among them, ATO system is the most important to achieve speed tracking.

In the field of deep learning, prior to the advent of the DQN algorithm, the application of reinforcement learning is often limited to the field of manual feature extraction and low-dimensional state space observation. However, DQN algorithm realizes end-to-end reinforcement learning of direct learning strategies from high-dimensional perception inputs. [1] That is, DQN can accomplish a series of tasks with only a small amount of prior knowledge.

In the Atari 2600 classic game trial [2], DQN Agent uses only pixels and scores as input, which can reach the level of professional players. In different games, Agent uses the same algorithm, network structure and hyper parameters.

Based on the characteristics of direct learning strategy of DQN by getting perceived input from high dimension, this paper presents a speed control method based on the current hot deep reinforcement learning. Through reinforcement learning, the fuzzy control scheme is automatically generated on the existing train model, the demand target speed curve and the demand strategy. In this process, the control method of this paper introduces the image representation method in deep learning to describe our state space, and uses the classical CNN algorithm to get the corresponding output state, and then strengthens the CNN network by the DQN algorithm in reinforcement learning. According to the strategy we designed, the network can output the best output information.

[+] Corresponding author. Tel.: + 86 17367078393.
E-mail address: gym12840@163.com.

At the same time, based on the original DQN algorithm, this paper also introduces the idea of step control, realizes the precise control of the train, restrains the speed oscillation, extends the average control length, accelerates the convergence speed and achieves the precise tracking of the train speed.

2. Train Control and DQN Algorithm

2.1. Principle of train control

According to Hangzhou Metro Line 4, the sampling speed and traction / braking force data show that the train traction / braking force is not infinite precision, but distributed in a fixed discrete value. That is, the speed control of the existing train can be achieved by increasing the traction force and braking force with limited precision. Therefore, it is feasible to select the discrete control output to complete the control of ATO.

The real train system outputs discrete control variables by the conductor with the experience and real-time status of the train. The real-time status here is the existing status and the established target (for example, following the target curve, fixed-point parking, etc.). The role played by the conductor is an intelligent control system, which is also the agent that we introduce in reinforcement learning. Through the train simulator (train model) designed in 2 and 3, the target speed curve, and the DQN algorithm we introduce below, we hope that the algorithm we used can automatically generate agents that can handle this task autonomously in many iterations to replace the conductor to complete train control.

The reinforcement learning system we constructed can be described as follow. We need to give the system a target model and reward function, a train model, and let our agent interact with the environment to get rewards and improve our own decision making algorithms. By constantly iterative process, the reward would finally reach to our setting value [3-4]. In the whole process, the agent doesn't know the specific parameters of the train model or the mathematical expression. At the same time, it doesn't know the specific reward function. It only needs to complete the learning process through the input and output interaction with the train simulator, which is the generation of control algorithms. This means that our algorithm is model free and strategy free, we can choose any model and strategy for input to control the generation of the algorithm.

2.2. DQN algorithm

Deep Q Network (DQN) is an algorithm that combines the advantages of Q Learning and Neural Network. Here we combine the idea of DQN (2015 google thesis [2]) to abstract train state into image input, and then use deep neural network (CNN) to complete Q value estimation of the state of the image input. Finally, the network can output results close to the best Q value.

The above part details the flow chart of the whole reinforcement learning algorithm and the construction of the train simulator. In the following, we will introduce how Agent can update itself through the sequence of {[*state, reward, action*]}.

First we analyse the train control problem. We can abstract the train operation into a Markov process with finite state. Each step can select an action in a finite set of actions. After the train system accepts the action, the state transition occurs, and at the same time agent gives the evaluation r, and jumps to the state S_{t+1}.

The goal that an agent in reinforcement learning system needs to achieve is to decide an optimal strategy to select the corresponding output state for a given input state so that the total expected value of the discount rewards will be maximized in the future when the entire decision-making process is completed. In other words, agent interacts with the environment and gets the sequence of observations, actions and rewards. The goal of agent is to select the actions that can maximize the rewards in the future.

Here we introduce the DQN algorithm in reinforcement learning. The idea of DQN is not to estimate the environmental model, but to directly optimize an iterative Q function. The DQN algorithm uses the deep convolution neural network (CNN) to approximate the Q function [1]:

$$Q^*(s,a) = \max_{\pi} E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ...|S_t = s, A_t = a] \quad (2)$$

$$= E_{s'}\left[r + \gamma \max_{a'} Q^*(s',a')|s,a\right] \quad (3)$$

Q-Learning is iteratively updated based on the following loss function:

$$L(w) = E\left[\left(r + \gamma \max_{a} Q'(s',a',w) - Q(s,a,w)\right)^2\right] \tag{4}$$

In the formula, s',a' are the next state and action, γ is the attenuation factor, which means that the expected return of the new state after completing the state transition discount to the expected return of the current state. The loss function is the difference between the target value and the current Q estimate. Neural network training is an optimization problem that optimizes a loss function $L(w)$ with the goal of minimizing the loss function. Therefore, we get the formula for updating the Q value of DQN as:

$$\mathrm{Q}(S_t,A_t) \leftarrow \mathrm{Q}(S_t,A_t) + \alpha(R_{t+1} + \gamma \max_{a} Q(s',a',w) - Q(s,a,w)) \tag{5}$$

The algorithm proves to be the best state function that can converge to our needs in iterations. Therefore, we design the Agent to implement the above algorithm, and in theory we can get the best control strategy in train operation. Where r is the reward function in the flowchart reward function.

2.3. Step control strategy

In the speed control images of DQN algorithm, we find that there is a phenomenon of frequent oscillations in speed. And because the optional set of actions is limited, it cannot always meet the condition that the speed after generating the motion coincides with the target curve speed. Usually, when the speed is greater than the target curve speed, a braking force is generated, and the speed returns to the target curve speed immediately, and then continues to fall below the target curve speed. When the speed is less than the target curve speed, a traction force is generated, and the speed returns to the target curve speed immediately, and then continues to rise above the target curve speed. The process repeated again and again, and therefore it produced a constantly oscillating speed tracking image.

Based on the above tracking characteristics, we introduce a step control strategy, which can be expressed as follows:

$$\Delta U = U_t + \beta U_{t-1} + \beta^2 U_{t-2} + \beta^3 U_{t-3} + \ldots + \beta^k U_{t-k} \tag{6}$$

Among them, β and k are the key parameters of the step control strategy, which respectively represent the attenuation coefficient and the cut-off frequency, that is, thrust increment ΔU at the current moment t is composed of the trust increment at the moment t and the attenuation value $\beta^k U_{t-k}$ which is the increment at moment *[t-k,t-1]* attenuating at the moment t .

3. Experiment

3.1. Image description

The strategy we use to abstract the train input state into an image is shown in Figure 1.

Image height	*80*
Image width	*80*
Pixel _ t	*0.05*
Pixel _ v	*0.05*
Crash _ limit	*0.3*

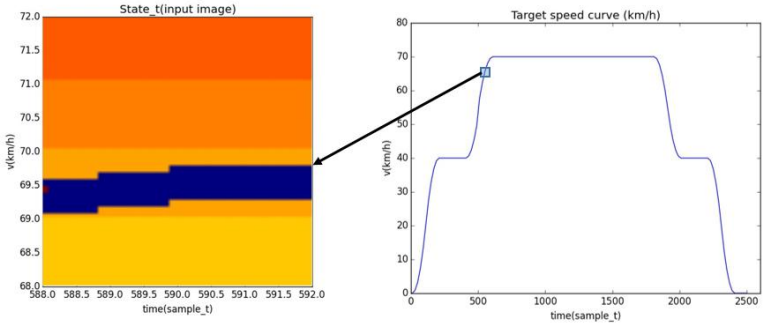


Fig. 1: Image of input state and target curve.

We render the target speed and the actual speed of the arbitrary sampling point to the image at any time. The precision is controlled by parameter adjustment of $pixel_t$, $pixel_v$. Our goal is to get the corresponding output based on the input image, so that the train can always run within the feasible speed area.

3.2. Neural network construction

For our output state space, we abstract the output state space into the space described by $\{max_{action}, K_U\}$. After embedding the step control strategy, the output state space is transformed into:

$$Action = \sum_{i=0}^{k=truncation} a_{t-i}\beta^i \quad (7)$$

Our convolution neural network (CNN) is shown in Figure 2. For any input of the state image S_t, we pass three convolutional layers, two fully connected layers, and finally output an output of length $[\max_{action}]$. Each value of the variable describes the Q value of action A_t corresponding to the state S_t estimated by the network, and the network selects the action output corresponding to the largest Q value. This process describes the best selection of discrete, fuzzy method to improve control accuracy.Compared with the traditional deep neural network, we made the following improvements in this paper:

(1) The input information is added. On the time axis, we expand the input information so that the network not only has the state space information of the current moment *t* but also the state space information of the moment [*t-3, t-1*].

(2) We cancel the pooling layer, because the image has the location-sensitive features, and the pooling disappears the location information of the image, and the de-pooling can make the characteristic images keep translation invariance.

(3) We increase the step control strategy.

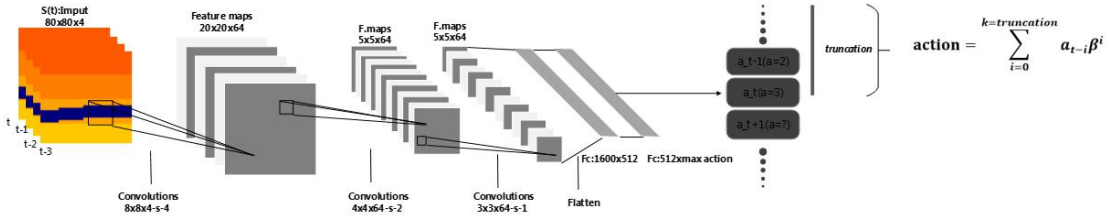


Fig. 2: Convolution neural network.

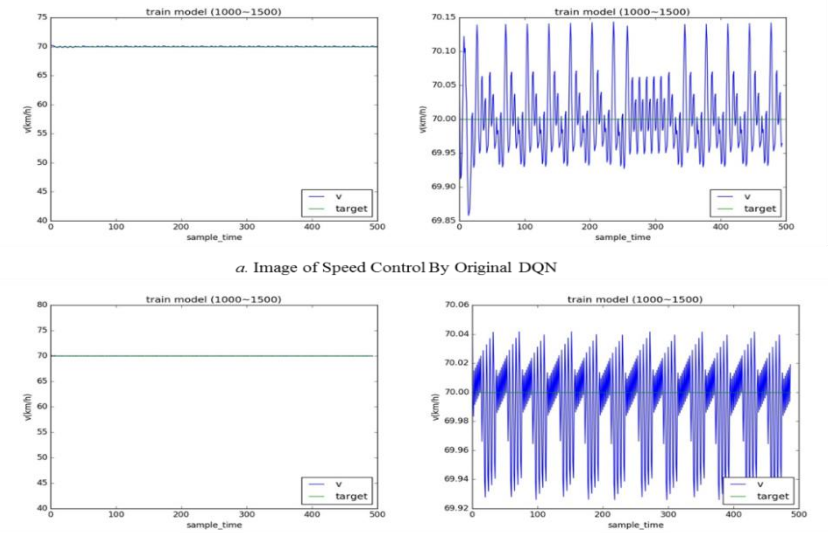
3.3. Step control strategy

By adding a step control strategy, our speed controlling effect has been significantly improved, compared with the previous speed control strategy by DQN algorithm. And it has the following improvements:

(1) Firstly, the speed control of the step control strategy greatly increases the average length of the control steps and fully fits any input states. The average step size of the DQN algorithm is 250 steps, and after the step control strategy is added, the speed tracking can be well implemented within 500 steps, and the absolute error is kept within a very small range.

(2) The introduction of step control strategy solves the phenomenon of speed oscillation very well, and the convergence speed has also been greatly improved.

The figure below is a comparison between the original speed control obtained by DQN algorithm and the speed control by adding the step control strategy. As can be clearly seen from the figure, the control effect by the step control is significantly improved, and the absolute error is also greatly reduced. After 1000 training data statistics, the absolute error between the controlled speed by the original DQN algorithm and the target speed stabilise at *abs (dv) = 0.0724*. After adding the step control strategy, the absolute error between the obtained speed and the target speed is reduced to *abs (dv) = 0.0169*.



a. Image of Speed Control By Original DQN

b. Image of Speed Control By Step Control Strategy

Fig. 3: Image of speed control by original DQN and step control strategy.

3.4. Variable speed control

By adding the step control, the algorithm completely realizes the control of the train model in the uniform speed in the range of 1000-1500, so we try to train the controllers for the five train models in the whole train speed range respectively, as shown in Figure5. The two sets of speed control section of the speed control chart.

The figure shows the effect of speed control on two groups of variable speed stages. It can be seen from the figure that the speed rising section and the uniform section, the speed decreasing section and the uniform speed section all achieve a good connection, and the speed error is controlled in a subtle range. This also shows that the algorithm can achieve speed control over the entire operating range, which verifies the effectiveness and adaptability of the algorithm.

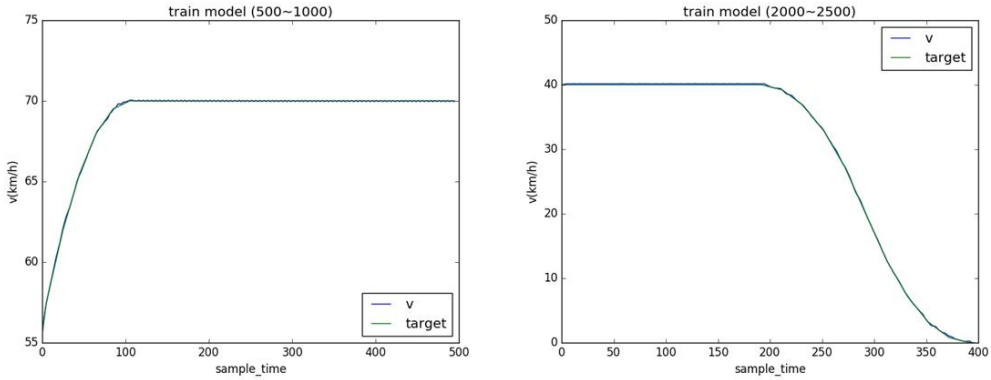


Fig. 4: Image of speed control in two variable speed stages.

4. Conclusion

This paper presents an automatic train control algorithm based on DQN algorithm, and add the step control strategy to achieve speed control over the entire operating range. We describe the state space by means of image representation in deep learning, and use the classical CNN algorithm to get the corresponding output state. Then we can use DQN algorithm to achieve the best output according to the strategy we designed. The introduction of step control strategy makes our algorithm have a better performance which has achieved stable controlling both in variable speed and uniform speed range.

5. Acknowledgements

This topic comes from 2015BAG19B03, the 2015 national science and technology support project jointly undertaken by Zhejiang University and Zhejiang Train Intelligent Engineering Technology Research Center Co., Ltd.

6. References

[1] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. Nature, 2015, 518(7540):529.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller. Playing Atari with Deep Reinforcement Learning [J]. Computer Science, 2015.

[3] LI Ning, GAO Yang, LU Xin, et al. A learning agent based on reinforcement learning. Computer Research and Development, 2001, 38(9): 1051 ~ 1056.

[4] L P Kaelbling, M L Littman, A W Moore. Reinforcement Learning: A survey. Journal of Artificial Intelligence Research,1996,4:237~285.