ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML032

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

USING DEEP REINFORCEMENT LEARNING FOR DESIGNING SUB-RELATIVISTIC ELECTRON LINAC LINAC

S. W. Shin†, M. Ghergherehchi, J. S Chai*, Department of Electrical and Computer Engineering, Sungkyunkwan Universtity, 16419, Suwon, Gyeonggi-do, Korea

Abstract

Generally, when designing an accelerator device, the design is based on the experience and knowledge of the designer. Most of the design process proceeds by changing the parameters and looking at the trends and then determining the optimal values. This process is time-consuming and tedious. In order to efficiently perform this tedious design process, a method using an optimization algorithm is used[1-4]. Recently, many people started to get interested in the algorithm used in AlphaGo, which became famous when it won the professional Go player developed by google[5]. The algorithm used in AlphaGo is an algorithm called reinforcement learning that learns how to get optimal reward in various states by moving around a solution space that the agent has not told beforehand. In this paper, we will discuss about designing an particle accelerator by applying Deep Q-network algorithm which is one kind of deep learning reinforcement learning.

INTRODUCTION

In 2013, Google's DeepMind team unveiled a program to play Atari games using a deep Q-network (DQN) algorithm with deep reinforcement learning[6], [7]. Until then, complex games were perceived as difficult to solve using artificial intelligence, but with the aid of algorithms developed by the DeepMind team, it became a challengeable problem. Currently, many researchers apply DQN algorithm to solve various problems[8].

Reinforcement learning can be applied to problems that require decisions to be made sequentially. To make a decision in a sequential manner is to take one action in the current state. This process of decision making is called the Markov decision process (MDP), which consists of state, action, reward, and policy. A computer algorithm that learns on its own is called an agent, and a state refers to static and dynamic state information that the agent has. An action is an agent's choice in a particular state, and an agent's action affects the state. Reward means how much the state has changed by the agent. In general, agents learn in the direction of receiving more positive rewards. The policy is the criterion that determines the choice before the agent takes action and is updated to receive more rewards each time. The MDP can be expressed as following.

State : $s \in S$
Actions : $a \in A$
Reward function : $R(r_t \mid s_t , a_t)$
Policy: $\pi(a_t \mid s_0,\ldots, s_t)$

Reinforcement learning takes priority over the value that is currently acquired rather than the value that will occur in the future. Therefore, by multiplying the reward obtained over time by the discount factor, the present value is evaluated to be higher than the future value.

The goal of reinforcement learning can be expressed as,

$$R_t = \sum_{i=0, \ldots, \infty} \gamma^i r_{t+i}$$

where γ=discount factor, R_t =sum of rewards.

DEEP Q-NETWORK ALGORITHM

Q-network is a model that provides information to obtain agent's high reward before taking action in specific state. The reinforcement learning algorithm using the Q-network aims to narrow the difference between the reward value predicted using the Q-network model and the reward value obtained when the actual action is performed. The model of the learned Q-network helps the agent get a high reward. The difference between the predicted reward value and the reward obtained from the actual action is called loss, and it can be expressed by the following equation.

$$L = \frac{1}{2}[r + \gamma max_{a'} Q(s', a') - Q(s, a)]^2$$

In other words, Q-network is learning to reduce loss. However, Q-network tends to be difficult to learn in case of problems with complicated states and actions. To solve this problem, a deep neural network method with several layers is applied. The Q-network algorithm with Deep neural network method is called DQN.

Adapting DQN to Linac Design

We have applied the DQN algorithm to a simple accelerator structure that can accelerate a single particle. The accelerator used in the experiment is aimed at accelerating the electron beam of 20 keV generated by the E-gun to 1 MeV, and the accelerator tube is composed of 20 cells. One cell generates an electric field of 20 MV/m. The accelerating tube used in the experiment has a length of 0.1 to 1 times the wavelength length, and there are total of 10 selectable actions.

In the accelerator design problem, the current state is the velocity of the beam, and the agent's action is the cell length of the RF cavity that matches the beam velocity. For efficient beam acceleration, the electric field and particle beam generated in the RF cavity must be synchronized.

Synchronized particle beams can be accelerated or even decelerated. The particle beam sets the acquired or

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML032

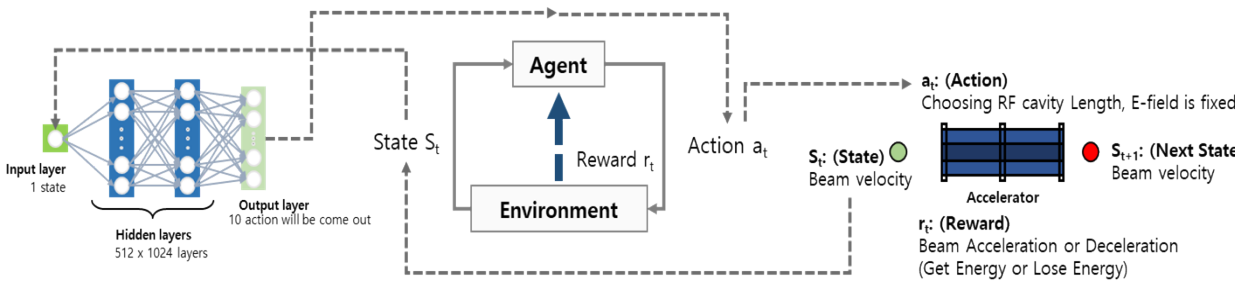


Figure 1: Adapted DQN structure to linac design.

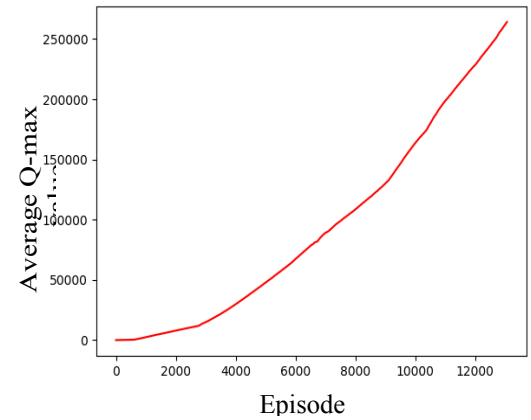
reduced energy to reward. At each moment, agents must perform actions appropriate to their state, so that they meet the MDP condition and can be solved with the reinforcement learning algorithm. The reward can be set as an acquired or lost energy. Since every agent must perform an action corresponding to the state, it satisfies the MDP condition, and it can be understood that it can be solved by the reinforcement learning algorithm.

A structure of the applied DQN algorithm is shown in Fig. 1. The agent obtains information from the Q-network for optimal action, and the Q-network uses a neural network consisting of two hidden layers to learn the model with the best performance. Since the input of a neural network is the number of states, it is 1 and the output is 10 because it is the number of actions. The hyper-parameters and the simulation environment variables of the neural network used in the learning are shown in the following Table 1. Simulation was conducted using ASTRA, which was developed by DESY and verified and used in several electron linac facilities[9].

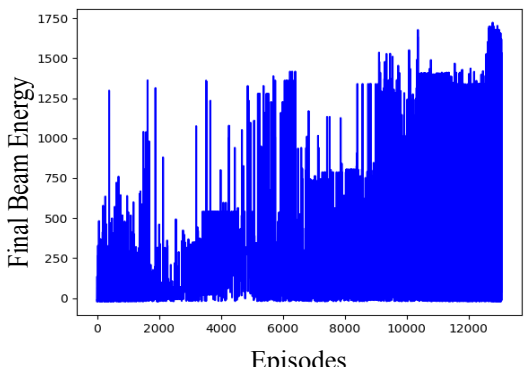
Table 1: Hyper-parameter and Environment Variable

Name	Value
Initial energy	20 keV
Electric field	20 MV/m
Activation function	Rectified Linear Unit(ReLU)
Discount factor	0.99
Learning rate	0.001
Hidden layer size	1,024
Batch size	64
Replay memory size	20,000

The DQN algorithm used in this study is the algorithm that was used when DeepMind team of Google played Atari game. It is featured by adding deep neural network and experience replay method to existing Q-network algorithm to improve learning efficiency. By making the layer thicker than the existing Q-network, we can solve the complicated problem that cannot be learned because of the thin layer.



(a)



(b)

Figure 2: Average Q-max value and summation of rewards during learning.

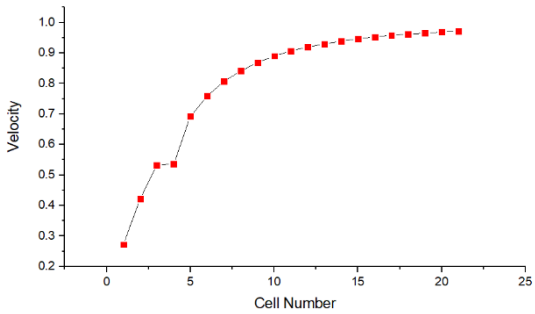
Prior to using the experience replay, DQN would learn by using rewards that occurred in action. Therefore, if the direction of learning is shifted to one side, it would have learned wrong. However, if the experience replay method is used, save it in the buffer instead of learning it when reward comes in. The stored reward is extracted by randomly sampling as much as the mini-batch size. Therefore, it is in principle excluded from being trapped at the local maximum.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

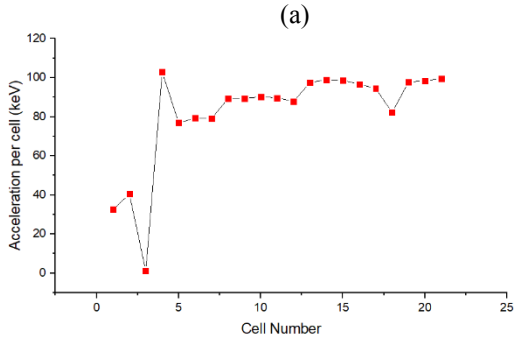
ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML032

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

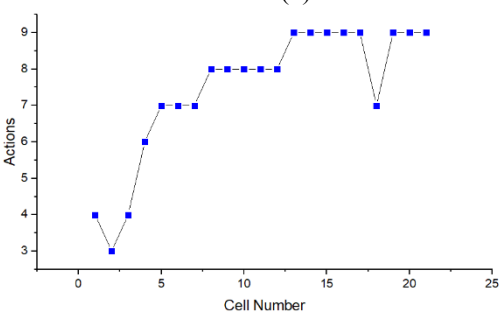
The learning process of the DQN algorithm applied to the accelerator design is shown in the following figure. The goal of the DQN is to predict best choice when the agent takes action in a particular state. The agent selects the action with the highest Q-value at each moment. Therefore, many researchers check the average value of Q-max during learning. The agent selects only the one with the highest Q-value for each action, so if the learning is successful, the average value of Q-max should be a graph that draws the upward direction. Also, if the good Q-function is well-learned and has a good model, the reward should increase with time.



(a)



(b)



(c)

Figure 3: ASTRA results:(a) velocity of electron beam (b) energy gain per cell (c) actions of each states

In the Figure 2, we can see that the average Q-max value and reward value increase gradually as the episode progresses.The ASTRA results of the electron accelerator obtained using the learned model are shown in the (c)

Figure 3. In the fourth cell, the agent does not select an accelerating RF cavity as in other cells, but rather selects a reduced RF cavity. It is assumed that this is an artificial action to match the synchronous phase of the incident beam with the electric field. In the 17th cell, selecting the 7th action rather than the 9th action is shown as a mistake in the model and we can see that the algorithm needs more optimization.

DISCUSSION

The accelerator design satisfies the MDP condition, therefore reinforcement learning can be used to solve the problem. The DQN algorithm was applied to the accelerator design problem, and it was confirmed that the learning was done properly by checking the average Q-max value and total reward value during learning.

In this study, deep reinforcement learning was applied to a simple structure and its usability was confirmed. It can be used in various places in the field of accelerator in the future. For example, recently developed dielectric laser-driven accelerators require more than 1000 cells for non-relativistic electron beam acceleration. Therefore, for the design of an accelerator tube with high acceleration efficiency, it can not be designed by a person directly controlling more than 1000 cells so automatic optimization algorithm is required. In this paper, the state is velocity, the reward is acceleration, and the action is only RF cavity. However, future studies will be able to input various state and reward variables such as transverse, longitudinal emittance, and beam current and action can be various magnets and beam line components as well as the RF cavity to the beam line design.

The advantage of deep reinfoecement learning is that once model is learned, it can be reused at different system. For example, the model learned in the up stream of the beam line can be applied in a down stream without further learning. Therefore, the learned model is applicable regardless of the length of the beam line.

ACKNOWLEDGMENTS

This work has supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT)(NRF-2017R1D1A1B03035711).

REFERENCES

[1] Z. Tang, Y. Pei, and J. Pang, “Optimal design of a standing-wave accelerating tube with a high shunt impedance based on a genetic algorithm,” *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 790, pp. 19–27, 2015.

[2] A. Hofler *et al.*, “Innovative applications of genetic algorithms to problems in accelerator physics,” *Phys. Rev. Spec. Top. - Accel. Beams*, vol. 16, no. 1, pp. 1–40, 2013.

[3] S. W. Shin *et al.*, “Design of 6 MeV X-band electron linac for dual-head gantry radiotherapy system,” *J. Korean Phys. Soc.*, vol. 71, no. 12, pp. 1048–1055, 2017.

[4] R. Bartolini, M. Apollonio, and I. P. S. Martin, “Multiobjective genetic algorithm optimization of the beam dynamics in linac drivers for free electron lasers,” *Phys. Rev. Spec. Top. - Accel. Beams*, vol. 15, no. 3, pp. 1–8, 2012.

[5] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[6] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning,” pp. 1–9, 2013.

ISBN: 978-3-95450-184-7 doi:10.18429/JACoW-IPAC2018-THPML032

[7] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[8] Richard Evans, "DeepMind AI Reduces Google Data Centre Cooling Bill by 40%."
[9] DESY, "A Space Charge Tracking Algorithm (ASTRA)." [Online]. Available, http://www.desy.de/~mpyflo/.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.