EdgeRL: A Light-Weight C/C++ Framework for On-Device Reinforcement Learning

Sang-Soo Park, Dong-Hee Kim, Jun-Gu Kang, Ki-Seok Chung*
Dept. of Electronic Engineering
Hanyang University
Seoul. South Korea
po092000@hanyang.ac.kr, dongheekim@hanyang.ac.kr, gjk6626@hanyang.ac.kr, kchung@hanyang.ac.kr*

Abstract— **Advances in reinforcement learning (RL) have achieved significant success in many areas. However, RL typically requires a large amount of computation and memory. Often RL implemented in Python is too heavy to run on a resource-limited edge device. Therefore, making the RL model lighter is very important for on-device machine learning. In this paper, we propose a lightweight C/C++ RL framework aiming for RL on edge devices. The proposed RL framework is designed to run on a single-core processor that is typically included in a resource-limited embedded platform. The evaluation using OpenAI Gym's CartPole demonstration shows that the model can be trained on an edge device in real-time.**

Keywords; Reinforcement Learning; On-device learning; Edge device;

I. INTRODUCTION

Widespread adoption of artificial intelligence (AI) is rapidly changing our life. In particularly, deep neural network (DNN) attracts lots of attention due to its excellent performance in many application areas [1]. It has achieved remarkable success in computer vision, speech recognition, and control engineering.

One of the most successful deep learning methods is reinforcement learning (RL) [2]. The goal of RL is to enable an agent to learn a good strategy from repeated trials and received feedbacks on the trials. Especially, RL has shown great success in the field of control engineering [3]. Unlike the other DNN methods, the agent in RL explores their environment and learns a desirable action by itself. In other words, the agent is capable of actively adapting the environment to maximize the reward. Exploring a large design space to find an appropriate action through repeated trials typically requires a large amount of computation and memory usage.

Python is a high-level programming language, and one of its advantages is that it is very easy to read and write a Python program. Also, it is very easy to learn. Therefore, Python is one of the most popular programming languages for implementing machine learning (ML) models. On the other hand, ML models written in Python are known to be quite heavy in the sense that the execution speed is slow, and the memory usage level is high. Therefore, it may not be adequate to run the Python implementation on resource-limited embedded edge devices [4].

In this paper, we propose a lightweight C/C++ RL framework called EdgeRL aiming for RL on resource-limited edge devices. EdgeRL is designed to run on a single-core processor that has the runtime environment only for C/C++. The evaluation results show that on-device RL on an edge device is practically feasible using EdgeRL.

II. DNN FRAMEWORKS FOR REINFORCEMENT LEARNING

A. Reinforcement Learning (RL)

RL is an ML algorithm that learns by itself through many trials. Fig. 1 shows a pictorial representation of the RL process. In RL, the agent learns from interactions with an environment without any explicit supervision. The agent interacts with the environment by actions (a_t). It receives the feedback of its action from the environment (r_t) in terms of reward or penalty and observes the change of the environment as the result of the action. That is, the state of the environment (s_t) at time t is continuously monitored by performing action a_t and receiving feedback r_t. The agent moves to the next state s_{t+1} after receiving a reward r_t, with probability $P(s_{t+1}|s_t, a_t)$. The goal of the agent is to maximize the cumulative reward over time through its choices of actions.

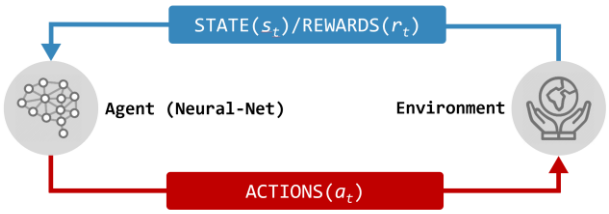

Figure 1. Process diagram of reinforcement learning

B. Framework Architecture

In a typical RL framework, the agent and the environment are implemented in Python. The feedback between the agent and the environment is implemented by way of exchanging memory objects in Python [4]. However, both the computation capability and the size of the memory on an edge device may not be sufficient to run RL in Python because interaction between the agent and the environment requires a huge amount of computation and memory usage. To solve this problem, we

propose a C/C++ RL framework that can run on an edge device without Python.

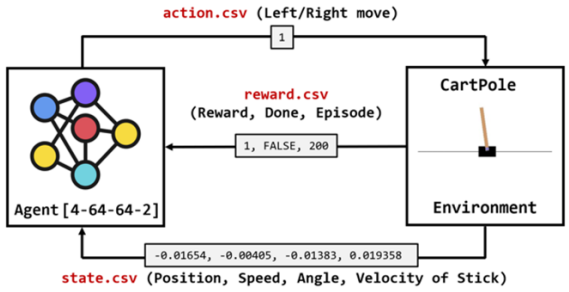

Figure 2. Archiecture of EdgeRL

Fig. 2 shows the overall architecture of the proposed RL framework, called EdgRL. OpenAI Gym's CartPole [5] is chosen as the target problem to solve through RL. The goal of CartPole is to move the cart along a track to prevent the attached pole from falling over. All the information exchange is conducted using the CSV files. The agent sends actions (`action.csv`) to the environment (`CartPole`) and receives rewards (`reward.csv`) and states (`state.csv`) from the environment.

III. Experiments and Discussions

A. *Experimental Setup*

In this paper, the target edge device that the proposed RL framework will be run is an NXP's i.MX6 SoC platform. The target device is equipped with a single ARM Cortex A9 processor with the maximum clock frequency of 1GHz and a 512MB DDR3 memory module. Cortex A9 has a 128bit NEON unit for single instruction multiple data (SIMD) processing. To evaluate the performance of real-time training, the proposed RL framework was run both on a workstation and on the target board. The detailed specification of the two platforms is shown in Table I.

TABLE I. Specification of Platforms Used for Evaluation

	Workstation	**Edge device**
Processor/# of Cores	Intel i7-9700K (8)	ARM Cortex A9 (1)
Processor Clock	3.6GHz	1.0GHz
Computing Power	424.3GFLOPS	1GFLOPS
Memory Size	32GB DDR4	512MB DDR3
OS	Ubuntu 16.04	Yocto 1.8
Library	OpenBLAS	

For performance evaluation, OpenAI Gym was used as the workload. OpenAI Gym is one of the most widely used benchmarks in RL. The agent is configured with a multi-layer perceptron (MLP) model (4-64-64-2, Sigmoid, REINFORCE algorithm), and the environment is set to CartPole v0. The score curve and the training time have been compared.

B. *Score curve and Execution time*

Fig. 3 shows the score results when EdgeRL is trained on-the target platform in real time. A score on CartPole indicates how well the user has played the game, and the higher the score, the better [5]. In the experiment, the initial score is -100, and as the learning progresses, the score increases. The number of episodes is about 1,000. The highest score was achieved in around 850 episodes. The proposed EdgeRL is trained on both the workstation and the target edge platform to compare the execution time. Table II shows the results of the training time required for about 950 episodes. The raw computational power of the workstation is about 400 times better than that of the target platform, but the execution time is about 2.6 times better. This result shows that on-device RL on an edge device is practically feasible using EdgeRL.

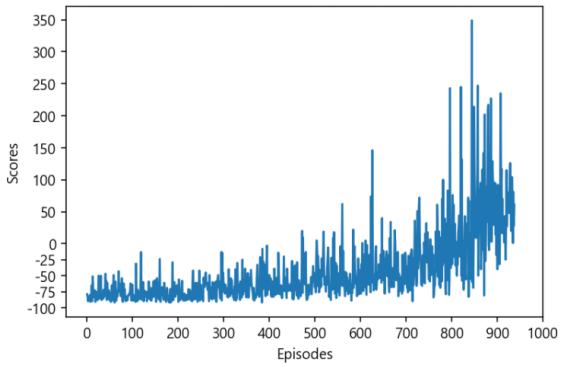

Figure 3. Score graph of CartPole

TABLE II. Execution Time of EdgeRL on CartPole

	Workstation	**Edge device**
Episodes	948	959
Execution Time	307s	802s

IV. Conclusion

In this paper, a light-weight C/C++ RL framework for an edge device was presented. Experiments results confirmed that the proposed C/C++ RL framework can effectively train a problem in real time on an embedded edge device.

Acknowledgment

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-01304, Development of Self-learnable Mobile Recursive Neural Network Processor Technology).

References

[1] Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 105(12), 2295-2329.

[2] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 34(6), 26-38.

[3] Ibarz, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., & Levine, S. (2021). How to train your robot with deep reinforcement learning: lessons we have learned. The International Journal of Robotics Research, 40(4-5), 698-721.

[4] Watanabe, H., Tsukada, M., & Matsutani, H. (2020). An FPGA-Based On-Device Reinforcement Learning Approach using Online Sequential Learning. arXiv preprint arXiv:2005.04646.

[5] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.