A Memory Efficient Deep Reinforcement Learning Approach For Snake Game Autonomous Agents

Md. Rafat Rahman Tushar[1]
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
rafat.tushar@northsouth.edu

Shahnewaz Siddique[2]
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
shahnewaz.siddique@northsouth.edu

arXiv:2301.11977v1 [cs.AI] 27 Jan 2023

***Abstract*—To perform well, Deep Reinforcement Learning (DRL) methods require significant memory resources and computational time. Also, sometimes these systems need additional environment information to achieve a good reward. However, it is more important for many applications and devices to reduce memory usage and computational times than to achieve the maximum reward. This paper presents a modified DRL method that performs reasonably well with compressed imagery data without requiring additional environment information and also uses less memory and time. We have designed a lightweight Convolutional Neural Network (CNN) with a variant of the Q-network that efficiently takes preprocessed image data as input and uses less memory. Furthermore, we use a simple reward mechanism and small experience replay memory so as to provide only the minimum necessary information. Our modified DRL method enables our autonomous agent to play Snake, a classical control game. The results show our model can achieve similar performance as other DRL methods.**

***Index Terms*—Deep Reinforcement Learning, Convolutional Neural Network, Deep Q Learning, Hyperparameter Tuning, Replay Size, Image Preprocessing**

I. Introduction

Complex problems can be solved in real-world applications by carefully designing Deep Reinforcement Learning (DRL) models by taking high dimensional input data and producing discrete or continuous outputs. It is challenging to build a agent using sensory data capable of controlling and acting in an environment. The environment is also complex and primarily unknown to the acting agent. The agent needs to learn the underlying distribution of the state and action spaces, and the distribution changes as the agent encounters new data from an environment. Previously reinforcement learning algorithms [1]–[5] were presented with lower constraint problems to demonstrate the algorithms effectiveness. However, these systems were not well generalized for high dimensional inputs; thus, they could not meet the requirements of practical applications.

Recently, DRL has had success in CNN based vision-based problems [6]–[8]. They have successfully implemented DRL methods that learn to control based on image pixel. Although the image-based DRL methods have enjoyed considerable success, they are memory intensive during training as well as deployment. Since they require a massive amount of memory, they are not suitable for implementation in mobile devices or mid-range autonomous robots for training and deployment.

All modern reinforcement learning algorithms use replay buffer for sampling uncorrelated data for online training in mainly off-policy algorithms. Experience replay buffer also improves the data efficiency [9] during data sampling. Since the use of neural networks in various DRL algorithms is increasing, it is necessary to stabilize the neural network with uncorrelated data. That is why the experience replay buffer is a desirable property of various reinforcement learning algorithms. The first successful implementation of DRL in high dimensional observation space, the Deep Q-learning [6], used a replay buffer of 10^6 size. After that, [8], [10]–[12], to name a few, have solved complex high dimensional problems but still use a replay buffer of the same size.

Experience replay buffer suffers from two types of issues. One is to choose the size of the replay buffer, and the second is the method of sampling data from the buffer. [13]–[15] consider the latter problem to best sample from the replay buffer. But the favorable size for the replay buffer remains unknown. Although [15] points out that the learning algorithm is sensitive to the size of the replay buffer, they have not come up with a better conclusion on the size of the buffer.

In this paper, we tackle the memory usage of DRL algorithms by implementing a modified approach for image preprocessing and replay buffer size. Although we want the agent to obtain a decent score, we are more concerned about memory usage. We choose a Deep Q-Network (DQN) [6] for our algorithm with some variations. Our objective is to design a DRL model that can be implemented on mobile devices during training and deployment. To be deployed on mobile devices, memory consumption must be minimized as traditional DRL model with visual inputs sometimes need half a terabyte of memory. We achieve low memory consumption by preprocessing the visual image data and tuning the replay buffer size with other hyperparameters. Then, we evaluate our model in our simulation environment using the classical control game named Snake.* The results show that our model can achieve similar performance as other DRL methods.

[1]Research Assistant.
[2]Assistant Professor, IEEE Member.
*GitHub implementation: https://github.com/rafattushar/rl-snake

II. Related Work

The core idea of reinforcement learning is the sequential decision making process involving some agency that learns from the experience and acts on uncertain environments. After the development of a formal framework of reinforcement learning, many algorithms have been introduced such as, [1]–[5].

Q-learning [1] is a model-free asynchronous dynamic programming algorithm of reinforcement learning. Q-learning proposes that by sampling all the actions in states and iterating the action-value functions repeatedly, convergence can be achieved. The Q-learning works perfectly on limited state and action space while collapsing with high dimensional infinite state space. Then, [6] proposes their Deep Q-network algorithm that demonstrates significant results with image data. Among other variations, they use a convolutional neural network and replay buffer. Double Q-learning [16] is applied with DQN to overcome the overestimation of the action-value function and is named Deep Reinforcement Learning with Double Q-Learning (DDQN) [8]. DDQN proposes another neural network with the same structure as DQN but gets updated less frequently. Refined DQN [17] proposes another DRL method that involves a carefully designed reward mechanism and a dual experience replay structure. Refined DQN evaluate their work by enabling their agent to play the snake game.

The experience replay buffer is a desirable property of modern DRL algorithms. It provides powerful, model-free, off-policy DRL algorithms with correlated data and improves data efficiency [9] during data sampling. DQN [6] shows the power of replay buffer in sampling data. DQN uses the size 10^6 for replay buffer. After that, [8], [10]–[12], [17], among others, have shown their work with the same size and structure as the replay buffer. Schaul et al. propose an efficient sampling strategy in their prioritized experience replay (PER) [13]. PER shows that instead of sampling data uniform-randomly, the latest data gets the most priority; hence the latest data have more probability of being selected, and this selection method seems to improve results. [15] shows that a large experience replay buffer can hurt the performance. They also propose that when sampling data to train DRL algorithms, the most recent data should the appended to the batch.

III. Method

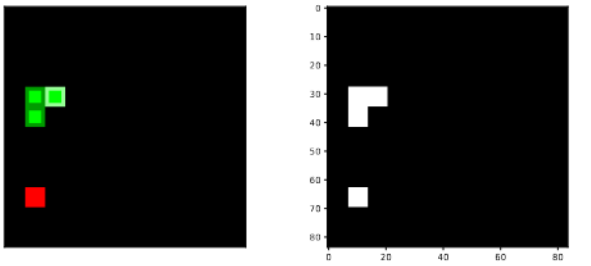
Our objective is to reduce memory usage during training time while achieving the best performance possible. The replay memory takes a considerable amount of memory, as described later. We try to achieve memory efficiency by reducing the massive replay buffer requirement with image preprocessing and the buffer size. The buffer size is carefully chosen so that the agent has the necessary information to train well and achieves a moderate score. We use a slight variation of the deep Q-learning algorithm for this purpose.

TABLE I
Reward Mechanism for Snake Game

Moves	Rewards	Results
Eats an apple	+1	Score Increase
Hits with wall or itself	-1	End of episode
Not eats or hits wall or itself	-0.1	Continue playing games

TABLE II
Memory Requirement for Different Pixel Data

	RGB	Grayscale	Binary
Data Type	float	float	int
Size (kB)	165.375	55.125	6.890
Memory Save % w.r.t. RGB	0%	67%	96%
Memory Save % w.r.t. Grayscale	-	0%	87.5%



(a) Before preprocessing (b) After preprocessing

Fig. 1. Visual image data before and after preprocessing

A. Image Preprocessing

The agent gets the RGB values in the 3-D array format from the games' environments. We convert the RGB array into grayscale because it would not affect the performance [18] and it saves three times of memory. We resize the grayscale data into 84×84 pixels. Finally, for more memory reduction, we convert this resized grayscale data into binary data (values only with 0 and 1). The memory requirement for storing various image data (scaled-down between 0 and 1) is given in Table II. Table II shows that it saves around 67% from converting RGB into grayscale and around 96% from converting RBG into binary. Also, the memory requirement reduces by around 87.5% converting from grayscale into binary. Visual pixel data transformation with preprocessing is given in Fig. 1. The preprocessing method is presented using a flowchart in Fig. 2.

B. Game Selection and Their Environments

The use-case of our target applications is less complex tasks. For this reason, we implemented the classical Snake game [19]

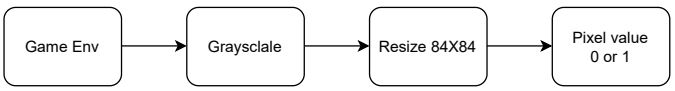


Fig. 2. Diagram of image preprocessing