[[1]](#endnote-1)[[2]](#endnote-2)Chrome Dino Run – Reinforcement Learning  
CS7IS2 Project (2019-2020)

Divyanshu Marwah, Sneha Srivastava, Anusha Gupta, Shruti Verma

marwahd@tcd.ie, ssrivast@tcd.ie, agupta@tcd.ie, sverma@tcd.ie

**Abstract.** Reinforcement Learning is one of the most advanced set of algorithms known to mankind which can compete in games and perform at par or even better than humans. In this paper we study most popular model free reinforcement learning algorithms along with convolutional neural network to train the agent for playing the game of Chrome Dino Run. We have used two of the popular temporal difference approaches namely Deep Q-Learning, and SARSA and then we have implemented Monte Carlo based Reinforcement training model to train the agent and then compare the rewards with respect to the iterations.

This document is a guideline for writing the final report for the CS7IS2 module *Artificial Intelligence*. You should follow its general structure as shown below.

You should not change its format (font, size, margin, space, etc.).

Report that not comply to the format or exceed the maximum length will be penalised (-5 marks).

Brevity is desirable in communication, however you should provide all those details necessary for the good understanding of the described methods and algorithms.

The report will be graded on the basis of:

* Originality;
* Technical soundness;
* Organisation;
* Clarity of presentation
* Adequacy of bibliography/Results (this last point strongly depends on the type of report)

**Your report should provide a survey and an experimental comparison of multiple solution approaches to a particular problem. This is a critical review of at least three papers that significantly contributed to advance the state-of-the-art for the problem you are analysing. It should not be a mere summary of the papers. You are expected to conduct an analytical review of the methods under analysis to try to find common aspect and differences, connections between methods, drawbacks and open problems. Unless the faced problem has emerged recently, students should choose their papers by diversifying the range of approaches used to solve the problem. A good guideline could be to choose a paper from a decade or two ago, and a couple of more recent papers. You need to experimentally evaluate approaches in a simulation of a problem, in a range of scenarios, and analyse the pros and cons of each approach.**

1 Introduction

In this section, you should introduce your work: what are the motivations behind this work? What is the relevant problem that you are investigating? Why is it relevant?

Briefly, introduce the background information required to understand the problem and the concepts that you will develop.

Learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of reinforcement learning. \_\_\_

The recent Deep Q-Network (DQN) algorithm [20] aims to tackle this problem, presenting one of the first successful combinations of RL and deep convolutional neural-networks (CNN) [13, 14], which are proving to be a powerful approach to representation learning in many areas.

Recent advances in deep learning have made it possible to extract high-level features from raw sensory data, leading to breakthroughs in computer vision [11, 22, 16] and speech recognition [6, 7].

One of the exciting benefits of model-based learning is the promise to substantially improve sample efficiency of deep reinforcement learning (see Chapter 8 in Sutton & Barto (2017)).

[Al-Naymat et al. 2019]

Chrome Dino or more popularly known as t-rex runner is a game which appears in Google Chrome in offline mode. The enemies in this game are the obstacles in the form of tree bushes/shrubs and birds. And the objective of the player is to stay safe from these obstacles. The player runs on its own and the only state of possible actions is to jump or do nothing. The score increases for every step taken while the player is alive.

In chrome Dino/T-rex runner, there are indefinite number of states that are possible (since it is an infinite canvas of obstacles at different distances), and they are given to the model as an input in the form of convoluted 4x4 array of pixels. Our aim is to compare different temporal difference approaches for RL and see which performs best in this kind of environment. We have not provided any game specific information to the network and let the model learn the next state based on the knowledge gained over the course of previous actions taken. The input given to the model is just the video signal (broken down into set of images using a CNN), set of possible actions (jump or do nothing), reward and termination state (end game, player died). Based on these given inputs the model learns when to take which action – similar to how a human play the game. In addition, we have kept the hyperparameters constant across all the algorithms to check the difference in convergence rate and compare their performance.

The paper is organized as follows: related work in this field is discussed in section 2, problem definition and different algorithms are described in detail in section 3. Section 4 presents the comparison, challenges and discussion on the results achieved. And finally, the conclusion and future work is outlined in section 5.

2 Related Work

In this section you will discuss possible approaches to solve the problem you are addressing, justifying your choice of the 3 you have selected to evaluate. Also, briefly introduce the approaches you are evaluating with a specific emphasis on differences and similarities to the proposed approach(es).

3 Problem Definition and Algorithm

The use of neural networks has given an opportunity to figure out solution of problems in complex and dynamic environments. Artificial Intelligence has moved from its humble start of object detection to its much complex applications like Google’s driverless cars. Different reinforcement learning algorithms have their own pros and cons and the trade-off between optimal solutions and safe solutions is the key to finding the best algorithm for the given problem.

A**s** Reinforcement learning process of learning what to do to maximise the reward score in any situation.

Our problem statement of the Chrome Dino Run has the model learning to figure out a way to jump over obstacles like the cactus and avoid birds and run on a plain field based on the actions it takes and its subsequent rewards. The base code remains the same across the different algorithms and compare the performance with respect to loss function and score versus timestamp.

a) The code has the first part that uses selenium to make an interface between our python code and the browser.

b) We then define the classes as the agent for taking actions and game state.

c) Then we proceed to pre process the image after grabbing it from the frame to reduce its input dimensionality.

d) The model is built that performs image convolution.

e) The model is then trained with respect to the different algorithms and that then selects the next actions based on the rewards and epochs are run and performance is noted.

Under a given policy π, the true value of an action a in a state s is, we make the standard assumption that future rewards are discounted by a factor of γ per time-step, and define the future discounted return at time t as

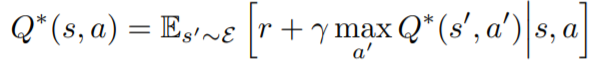
Rt = PT t 0=t γ t 0−t rt 0 , where T is the time-step at which the game terminates. Θ is the weight.

3.1 DQN(Deep Q Networks)

The combination of deep neural networks with Q Learning is called as Deep Q Network Learning. Q learning might work well in small state space but with more complex and sophisticated environments it drastically reduces in performance. The environment in a video game will be quite large and the actions that can be taken are multiple with each state that can be represented as pixels. To iteratively store q values in such a large environment is computationally expensive. We make use of deep neural networks to estimate the q values in each state action pair. The optimal q value that comes out of the neural network part of the model should be able to satisfy the Bellman Equation.

The addition of Experience Replay enhances the performance of the DQNs. [1]It stores the states, actions, transitions, rewards and terminal states and makes batches to update the q values. [1] Secondly the use of not all the frames actually improves the performance greatly. Four frames are grabbed and convoluted and taken as input.

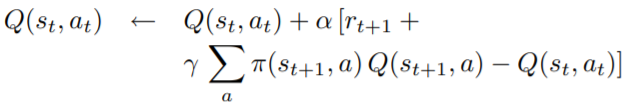
Update Rule:

It is an off-policy algorithm that focusses on finding the maximum q value and chooses the next step in a greedy manner.

3.2 Estimated SARSA

SARSA stands for State-Action-Reward-State-Action is also a method of Reinforcement Learning. It is an on policy method and it is defined as over “ state-action pair, rather than just the state”. [2]This follows a policy to take the next value of q instead of Q learning which has a greedy approach and doesn’t follow a policy. It uses the knowledge regarding the stochasticity in policy to perform updates that has a lower variance which in turn leads to better learning rate.

Update Rule for SARSA



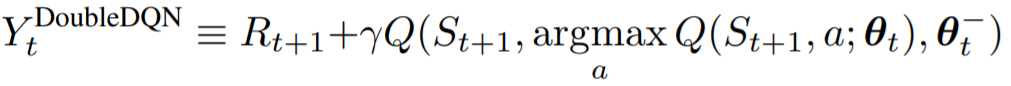
Expected SARSA takes the mean of all the values of Q in the current action.The update rule of the expected SARSA can be expressed as:

3.3 DDQN(Double Deep Q Networks)

The major concern with Deep Q learning is that it is known to overestimate the action values and hence negatively affects the performance.

Update Rule:

Its update is the same as for DQN, but replacing the target Y DQN t with Y



This means that, as in Qlearning, we are still estimating the value of the greedy policy according to the current values, as defined by θt. However, we use the second set of weights θ 0 t to fairly evaluate the value of this policy. [3]

In comparison to Double Q-learning (4), the weights of the second network θ 0 t are replaced with the weights of the target network θ − t for the evaluation of the current greedy policy. [3]

It has been experimentally shown to reduce the bias in the results.

4 Experimental Results

This section should provide the details of the evaluation. Specifically:

* Methodology: describe the evaluation criteria, the data used during the evaluation, and the methodology followed to perform the evaluation.
* Results: present the results of the experimental evaluation. Graphical data and tables are two common ways to present the results. Also, a comparison with a baseline should be provided.
* Discussion: discuss the implication of the results of the proposed algorithms/models. What are the weakness/strengths of the method(s) compared with the other methods/baseline?

5 Conclusions

Provide a final discussion of the main results and conclusions of the report. Comment on the lesson learnt and possible improvements.

A standard and well formatted bibliography of papers cited in the report. For example:

# References

|  |  |
| --- | --- |
| [1] | V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *DeepMind Technologies,* 2013. |
| [2] | H. v. Seijen, H. v. Hasselt, S. Whiteson and M. Wiering, "A Theoretical and Empirical Analysis of Expected Sarsa," *IEEE,* Vols. 978-1-4244-2761-1/09, 2009. |
| [3] | v. H. Hasselt , A. Guez and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *Google DeepMind,* 2015. |

1. [↑](#endnote-ref-1)
2. [↑](#endnote-ref-2)