Performing Deep Recurrent Double Q-Learning for Atari Games

Felipe Moreno-Vera
felipe.moreno@ucsp.edu.pe

Universidad Catlica San Pablo, Arequipa, Perú

1 Introduction

Currently, there is an increase the number of application in Reinforcement Learning, specially in Deep Reinforcement Learning with new techniques. One of application of DRL (Deep Reinforcement Learning) is in Games like AlphaZero (Go, Chess, etc) and video games like Mario, Top racer, Atari, etc. Deep Reinforcement Learning is considered like a third model in Machine Learning (with Supervised Learning and Unsupervised Learning) with a different learning model and architecture.

Sutton [1] define various models to describe Reinforcement Learning and how to understand it. DeepMind was the first to achieve this Deep Learning with AlphaZero and Go game using Reinforcement Learning with Deep Q-Learning (DQN) [2] and Deep Recurrent Q-Leaning (DRQN) [3], follow up by OpenAI who recently suprased professional players in Star Craft 2 (Gramve created by Blizzard) and previously in Dota 2 developed by Valve. Chen et al. [4] proposed a CNN based on DRQN using Recurrent Netowrks (a little variance of DRQN model using LSTM on agents actions to extract more information from frames.

2 Proposed model

We implement the CNN proposed by Chen et al. [4] with some variations in the last layers and using ADAM error. The first attempt was a simple CNN with 3 Conv 2D layers, with the Q-Learning algorithm, we obtain a slow learning process for easy games like SpaceInvaders or Pong and very low accuracy in complicated games like Beam Rider or Enduro. Then, we try modifying using Dense 512 and 128 networks at last layer with linear activation and relu, adding a LSTM layer with activation tanh.

List of Hyperparameters		
Iterations	10 000000	number of batch iterations to the learning process
miniBatch size	32	number of experiences for SGD update
Memory buffer size	900000	SGD update are sampled from this number of most recent frames
Learning Rate	0.00025	learning rate used by RMS Propagation
Training Frequency	4	Repeat each action selected by the agent this many times
Target Network Update Frequency	40000	number of parameter updates after which the target network updates
Update Frequency	10000	number of actions by agent between successive SGD updates
Replay start size	50000	The number of Replay Memory in experience to learn or forgot achievements
Exploration max	1.0	Max value in exploration
Exploration min	0.1	Min value in exploration
Exploration Steps	850000	The number of frames over which the initial value of e reaches final value
Gamma or Discount Factor	0.99	Discount factor γ used in the Q-learning update

Table 1: Hyperparameters used in models

In Table 2 we present our Hyperparameters using in our models, we denote this list of hhyperparameters as the better set (in our case). We run models over a NVIDIA GeForce GTX 950 with Memory 1954MiB using Tensorflow, Keras and GYM (Atari library) for python. We implement DDQN, DRQN, DQN and our proposed to combine DRQN with Double Q-Learning [5] algorithm using LSTM.

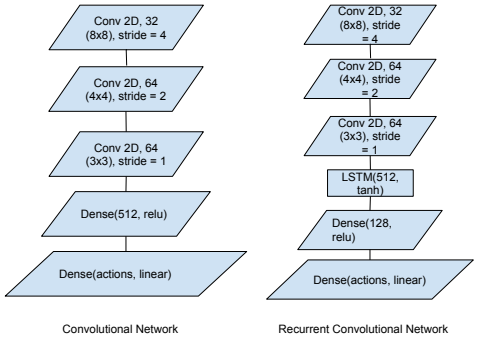


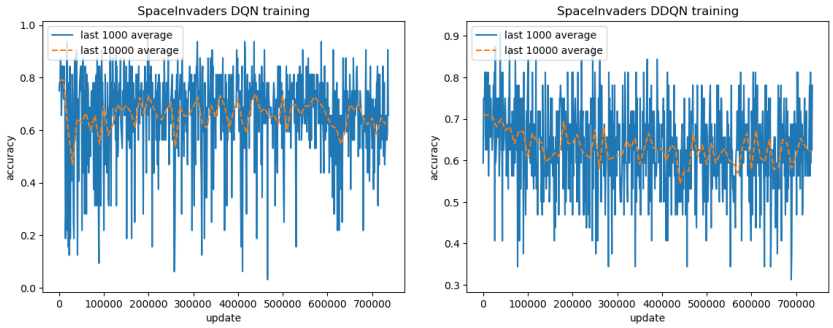
Fig. 1: Convolutional Networks used in our models.

3 Results

After to run our algorithms for 5 days with 10M Batch iterators, we obtain results for each model in each respective game. We get best scores for the 4 games mentioned above (SpaceInvaders, Enduro, Beam Rider and Pong), We compare with Volodymyr et al. [6] Letter about best scores form games obtained by DQN agents and professionals gamers (humans).

Models and respective Scores				
Model	SpaceInvaders	Enduro	Pong	Beam Rider
DQN	1450	1095	65	349
DRQN	1680	885	39	594
DDQN	2230	1283	44	167
DRDQN	2450	1698	74	876

Table 2: Results Scores of Space Invaders, Enduro, Pong and Beam Rider.



(a) DQN Accuracy (b) DDQN Accuracy

Fig. 2: DDQN vs DQN Accuracy.

References

1. Richard S. Sutton, Reinforcement Learning Architectures.
2. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Daan Wierstra, Alex Graves, Ioannis Antonoglou, Martin Riedmiller, Playing Atari with Deep Reinforcement Learning.
3. Matthew Hausknecht and Peter Stone, Deep Recurrent Q-Learning for Partially Observable MDPs.
4. Clare Chen, Vincent Ying, Dillon Laird, Deep Q-Learning with Recurrent Neural Networks.
5. Hado van Hasselt, Arthur Guez and David Silver, Deep Reinforcement Learning with Double Q-learning.
6. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis, Human-level control through deep reinforcement learning
7. Github Repository, https://github.com/Jenazads/LatinXinAI/tree/master/2019/