Demonstration of a modified Double Deep Q-Learning algorithm in Atari shooting games

Mario Cantero, Maryam Alghfeli, Nouf Alshamsi
Mohamed Bin Zayed University of Artificial Intelligence

Introduction

Reinforcement learning is about building an agent that learns from actions by interacting with the environment through trial and error. It receives rewards as feedback to solve control tasks or decision-making problems. The agent attempts to learn an optimal policy and performs an action for a given state to maximize expected future rewards through training.

Project Objectives

This project aims to create an agent that learns to play shooting games from the Atari games environments that share the same action space. The objective is to train an agent that plays Atari games better than an agent that follows a random policy.

Project Overview

Model Architecture

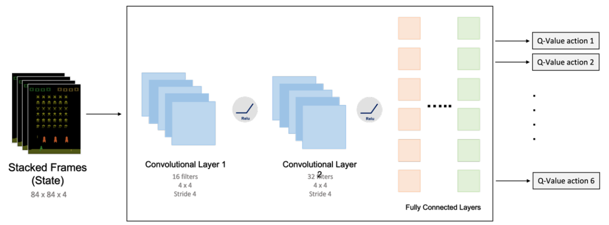
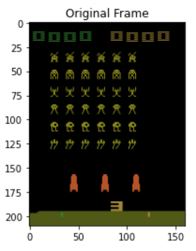


Fig. 1: Double Deep Q-Learning Network Architecture

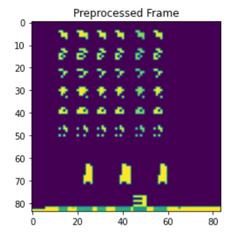
Preprocessing

The input frames are processed by:

- Stacking the frames for the agent to have a sense of movement.
- Converting the RGB image into a grayscale image.
- Cropping and resizing the image to capture the playing area.
- Normalizing the image before feeding it to the network.

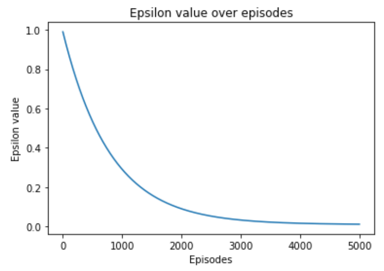


(a) Input Frame.



(b) Preprocessed Input Frame.

Epsilon Greedy Policy



Hyperparameters

- Replay memory capacity: 20000
- Initialization of the replay memory: 256
- Range of random frame skipping: [3,5]
- Range of number of initial random actions (left, right): [10, 20]
- C is updated every 15 steps.

Modified Deep Q-Learning Algorithm

```
Algorithm 1 Modified Deep Q-learning with Experience Replay and Weight Update Delay
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weight $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for $episode = 1, M$ do
  Get $\varepsilon$ value from the decay function according to current episode
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  Initialize $skip\_counter = 0$
  Get a random integer value $n\_skipped\_frames$ between 3 and 5 inclusive
  Get a random integer value $K$ between 10 and 20
  for $counter = 1, K$ do
    Perform a right or left movement action $a_t$ randomly
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_(s_{t+1})$
  end for
  while $!Done$ do
    if $skip\_counter \% n\_skipped\_frames == 0$ then
      Select a random action $a_t$ with probability $\varepsilon$, otherwise select $a_t = argmax_a Q(\phi(s_t), a; \theta)$
      Get a random integer value $n\_skipped\_frames$ between 3 and 5 inclusive
    end if
    Increase $skip\_counter$ value by one
    Execute action $a_t$ in emulator and observe reward $r_t$, frame $x_{t+1}$, and $Done$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1}, Done)$ in $D$
    Sample random mini-batch of 16 transitions $(\phi_j, a_j, r_j, \phi_{j+1}, Done)$ from $D$
    Set vector $y_j = \begin{cases} r_j, \text{ if episode terminates at step } j+1 \\ r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^-), \text{ otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters $\theta$
    Reset $\hat{Q} = Q$ every $C$ steps
  end while
end for
```

Results

- Space invaders did overcome an agent with random policy.
  - A Kolmogorov-Smirnov hypothesis test was used to check if two sets of 500 samples from each agent don't follow the same distribution.
  - P-value = 3.7e-09
- For the other two games, the hyperparameters and raining length were not good enough to achieve the objective.

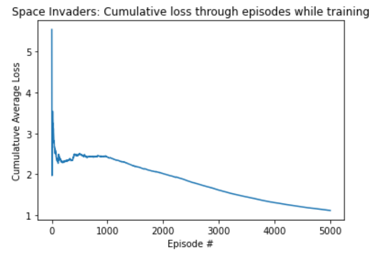


Fig. 1 The decreasing behavior of the loss in the training phase indicates that the agent is learning

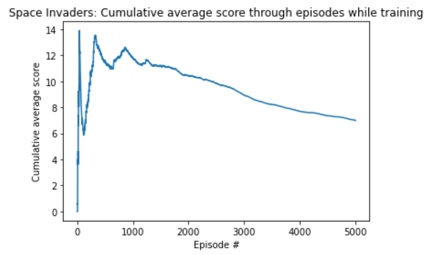
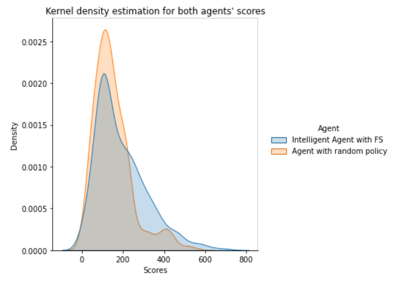


Fig. 1 The scores in the training phase are reaching convergence

Comparison



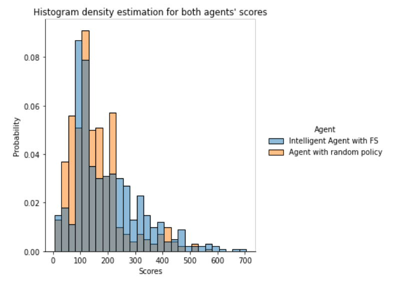


Fig. 1 From the distribution plots, we can notice that the trained agent generally performs better than an agent that only follows random actions

Future Improvements

- Train the network with a larger number of episodes.
- Boost the performance using the transfer learning method for the agent to adapt to different environments [1].
- Optimize the model's weights using genetic evolution algorithm [2].
- Use the prioritize experience that replays important transitions more frequently [3].

References

[1] Akshita Mittel and Purna Sowmya Munukutla. "Visual transfer between atari games using competitive reinforcement learning". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2019, pp. 0–0.

[2] Matthew Hausknecht et al. "A neuroevolution approach to general atari game playing". In: IEEE Transactions on Computational Intelligence and AI in Games 6.4 (2014), pp. 355–366.

[3] Ionel-Alexandru Hosu and Traian Rebedea. "Playing atari games with deep reinforcement learning and human checkpoint replay". In: arXiv preprint arXiv:1607.05077 (2016).