Simulation of Racing game using Reinforcement Learning

Rekik Ahmed

Berrejeb Amen Allah

Bouazizi Mouataz

Zid Mohamed Amin

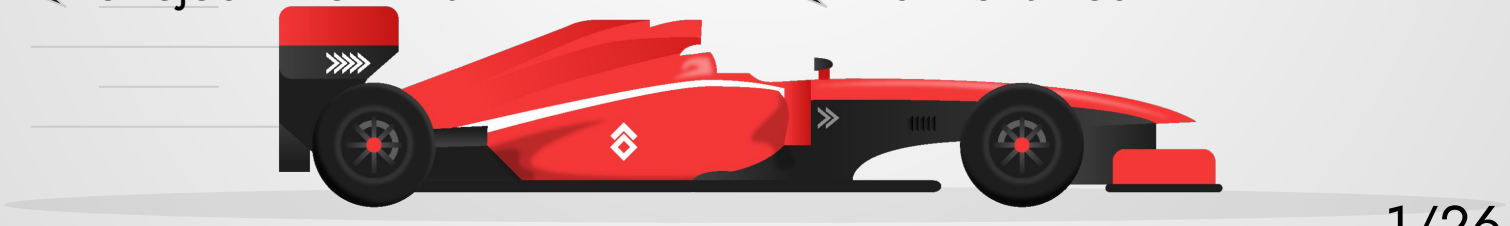
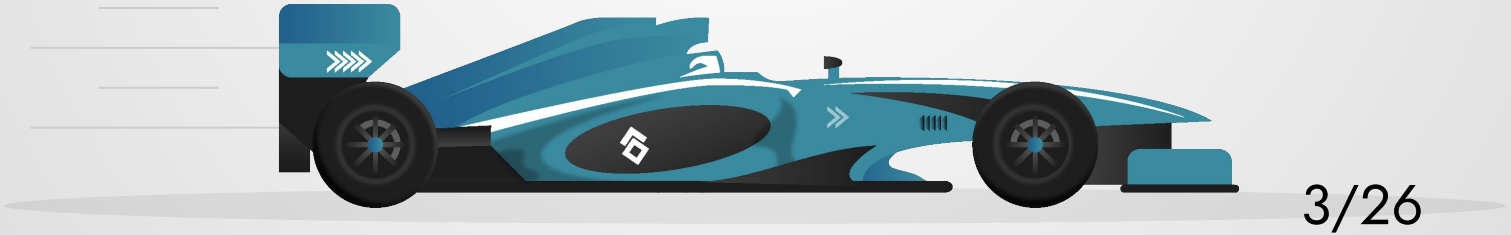


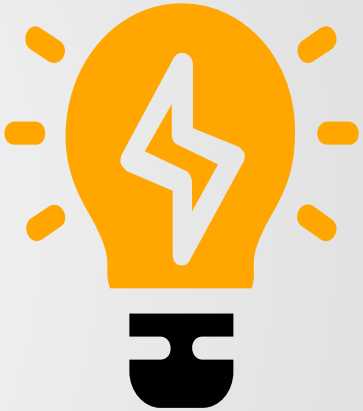
TABLE OF CONTENTS

01 Reference Paper



Introduction

- ***The foundation of our project is built upon the seminal paper "Playing Atari with Deep Reinforcement Learning" by Volodymyr Mnih, et al. This paper, published in 2013, marked a significant milestone in the field of reinforcement learning and served as the inspiration for our work.***

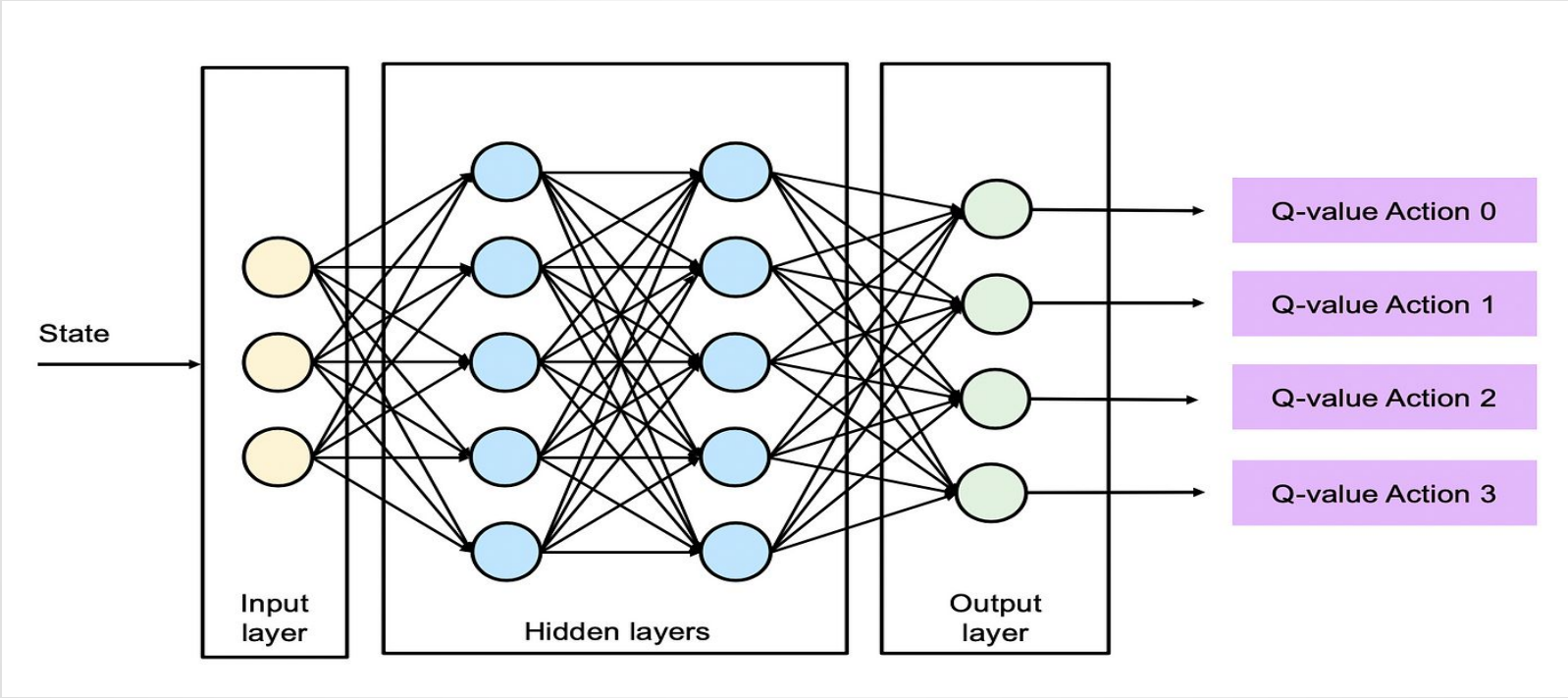


Key takeaways from the Reference Paper

- ***Deep Q-Networks (DQN) :***

  The paper introduced the concept of Deep Q-Networks, a neural network architecture that combines deep learning with reinforcement learning. DQN demonstrated remarkable success in learning to play various Atari 2600 games from raw pixel data.

DQN based trajectory design



- ***Experience Replay :***

  Experience Replay is a crucial component of the reinforcement learning approach discussed in "Playing Atari with Deep Reinforcement Learning." This algorithm enhances the stability and learning efficiency of deep reinforcement learning models, such as the Deep Q-Network (DQN).

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equa
  end for
end for

Let's break down the key steps of the Experience Replay algorithm:

1)Initialize Replay Memory :
Initialize a replay memory, denoted as D, with a capacity of N

2)Initialize Q-Function :
Initialize the action-value function Q with random weights

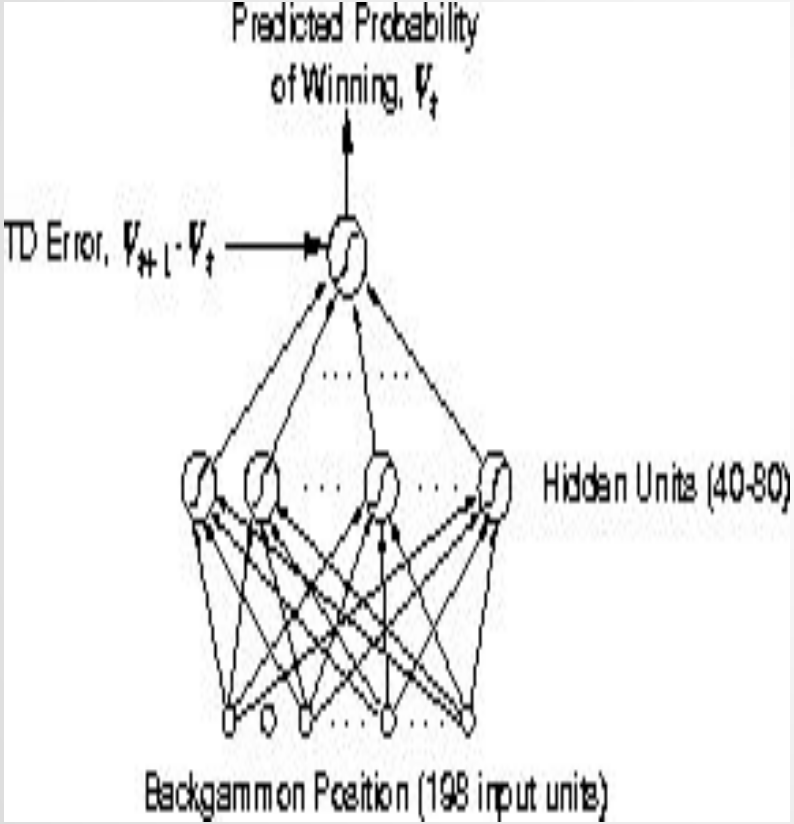
3)Episode Loop(for episode =1 to M) :
For each episode, initialize a sequence of states s1={x1}, and preprocess it into φ1=φ(s1)

4)Time Step Loop(for t=1 to T)

5)Experience Replay

6)End of Episode Loop :
Repeat the above steps for each episode within the total number of episodes M

Tessaro's TD-Gammon Architecture

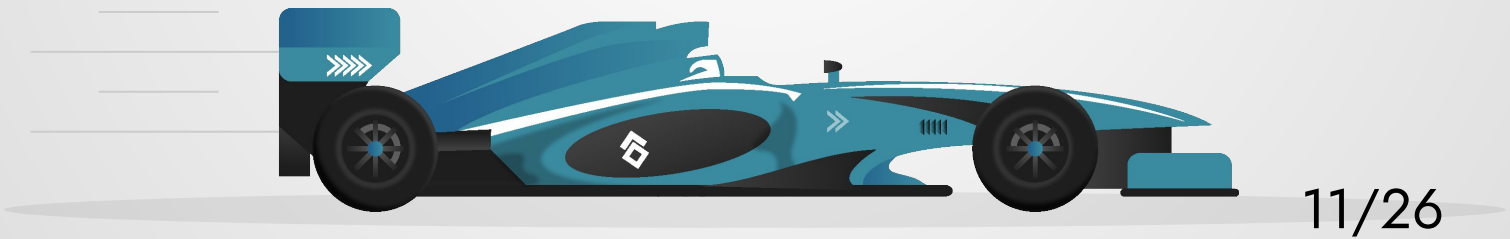


While the domain of racing games is significantly different from backgammon, the principles of using neural networks for value estimation and TD learning for reinforcement learning remain applicable.

- ***Inspiration of our Project :***

  The "Playing Atari with Deep Reinforcement Learning" paper served as a crucial starting point for our project. We drew inspiration from its key concepts and innovations, applying them to the domain of racing game simulations.

02 Agent training

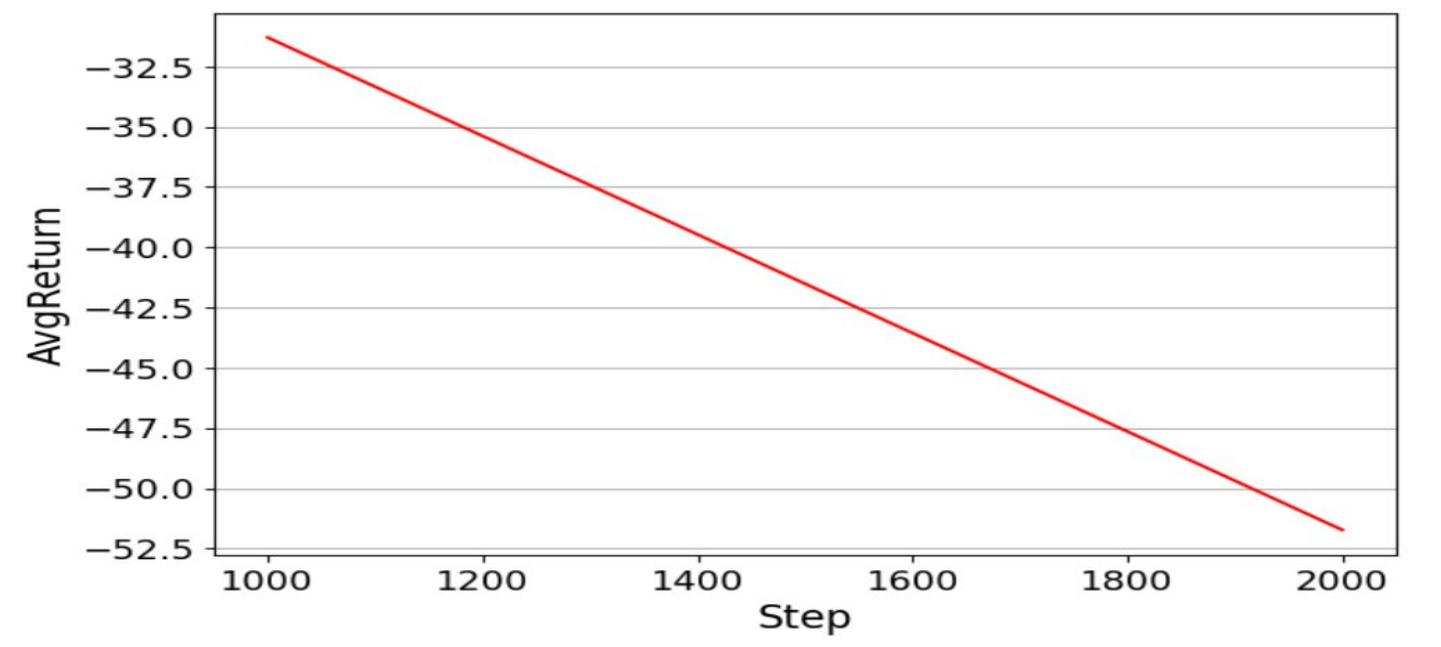


Training DQN Agent

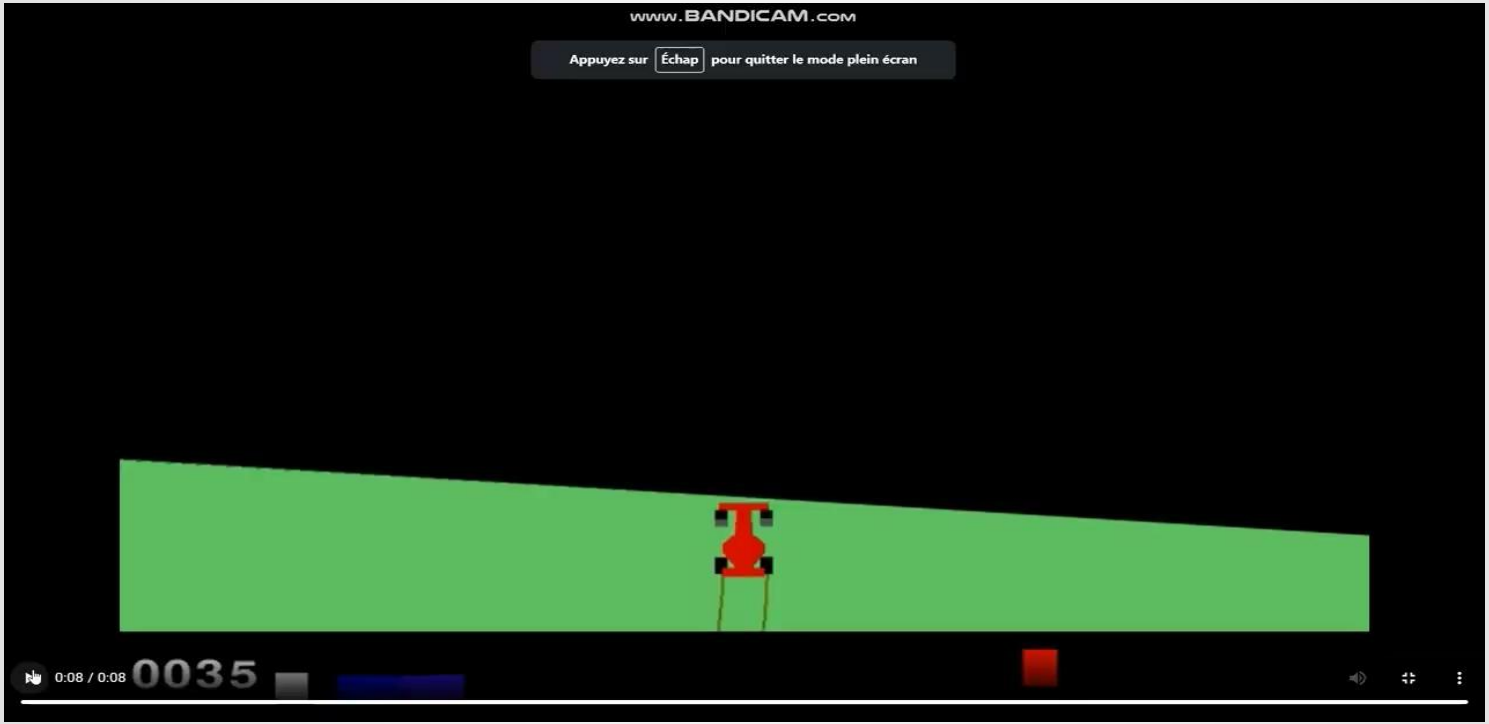
In our project, we trained a Deep Q-Network (DQN) agent, teaching it to make decisions through interactions with its environment. We continued training until the agent had reached a total of 2 Millions interactions.

To monitor its progress, we evaluated the agent's performance every huge numbers of steps. This approach showcases the potential of DQN agents in solving real-world problems through experiential learning

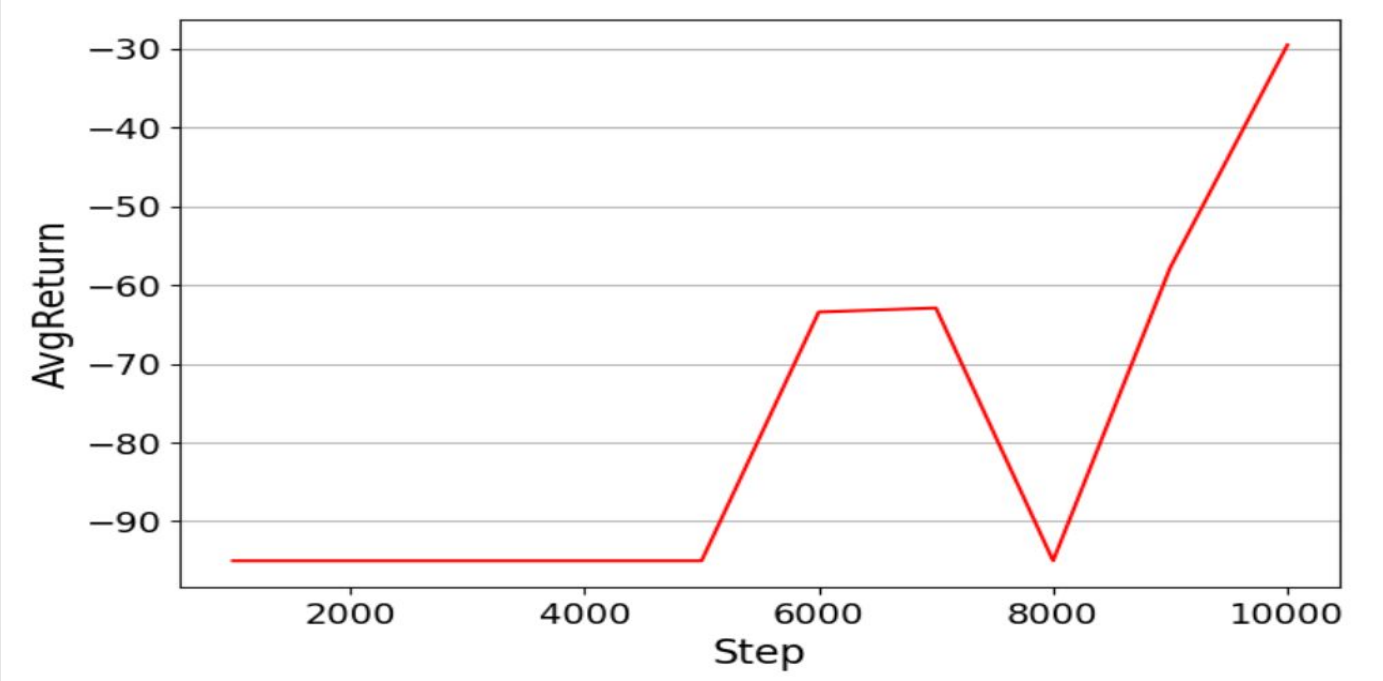
2000 steps


AvgReturn
−32.5
−35.0
−37.5
−40.0
−42.5
−45.0
−47.5
−50.0
−52.5
1000
1200
1400
1600
1800
2000
Step

2000 steps

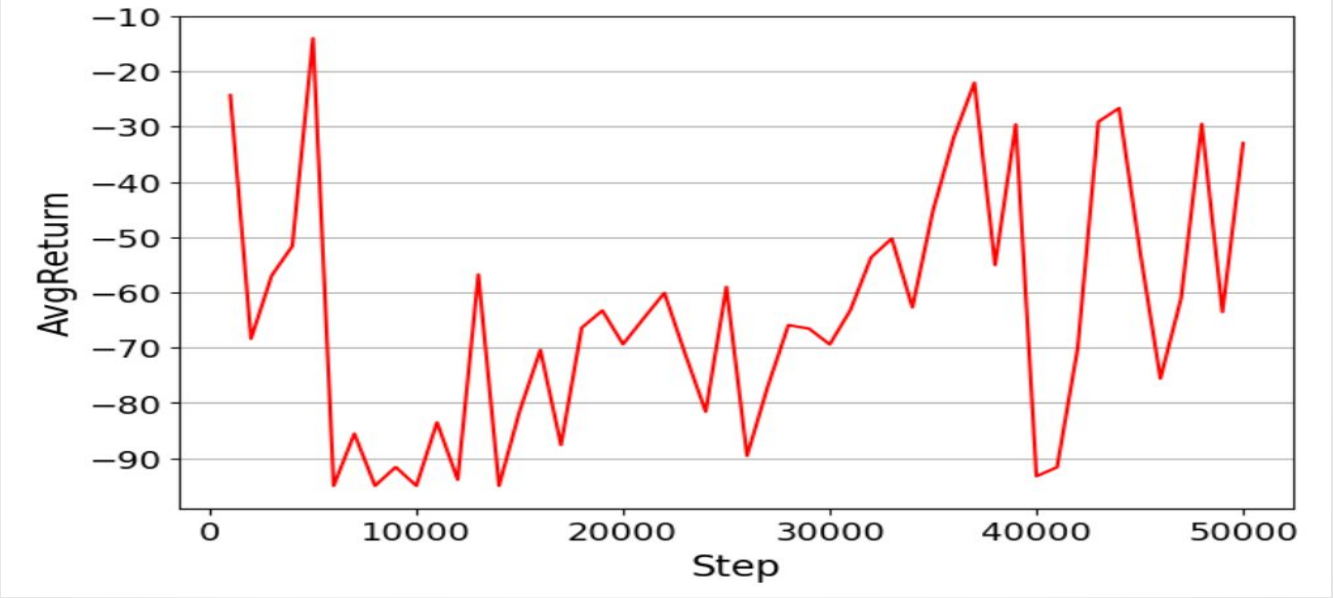

www.BANDICAM.com
Appuyez sur Échap pour quitter le mode plein écran
0:08 / 0:08
0035

10000 steps



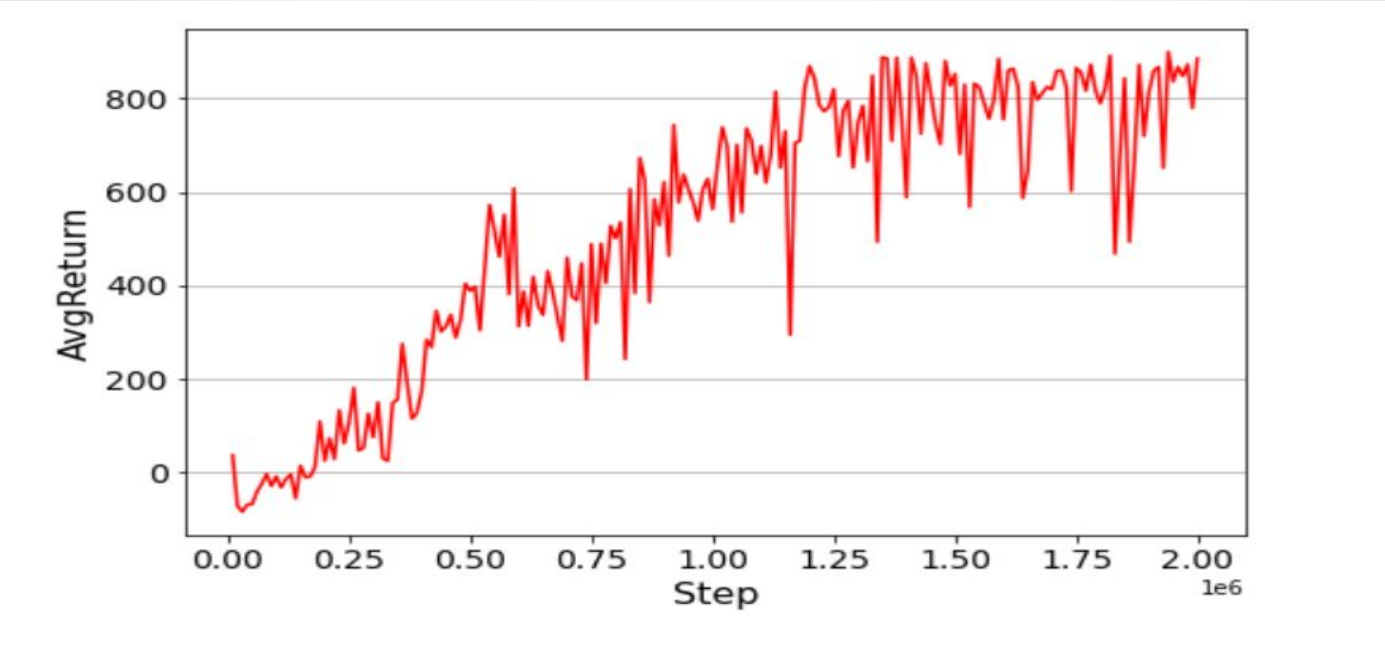
10000 steps

From 250000 to 300000 steps

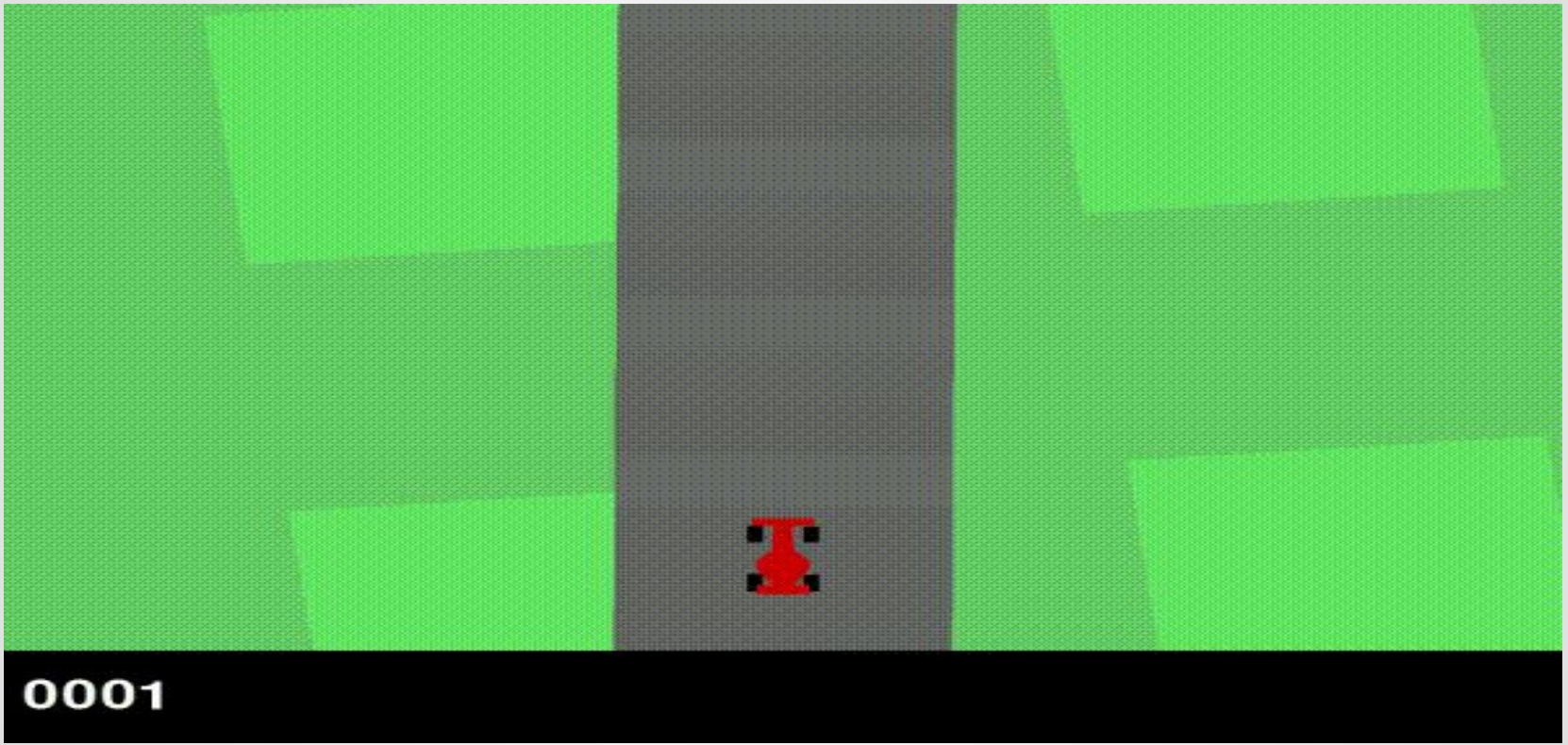


From 250000 to 300000 steps

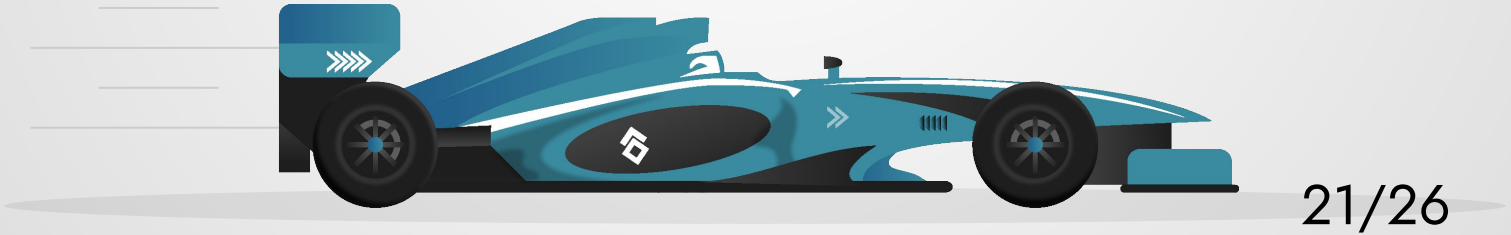
2 millions steps



Final Result



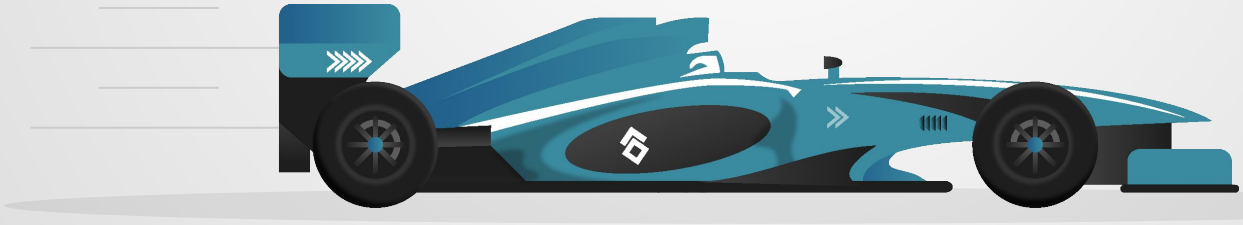
03 The project implementation



Code

- We are going now to explain the code for you
- We used several libraries like : Matplotlib , gym , pytorch , etc…

04 Conclusion and future improvement



Conclusion

- *Our project represents a significant step forward in the application of reinforcement learning techniques to the simulation of racing games. We've achieved several noteworthy outcomes and insights throughout this endeavor*
- *We also conclude that deep reinforcement learning is a very challenging task and need a huge number of working hours and computational resources !!!*

Future improvement

- *While we have made significant progress, there is always room for improvement and expansion. Here are some directions for future work :*

  - Optimized Training
  - Model Deployment using Web Development
  - Other Environments like : Frozen lake , Mars Lander
  - Real-World Applications

THANKS

DO YOU
HAVE
ANY QUESTIONS?