**RESULTS OF CARTPOLE PROBLEM BUILT ON RL-STUDIO**

**Starting point:**  
  
gamma: 0.95  
epsilon\_discount: 0.9998

**GOAL** → keep the pole always up and in the center

* Reward = 1 if still alive (up), 0 otherwise
* Steps per run = 1000
* states

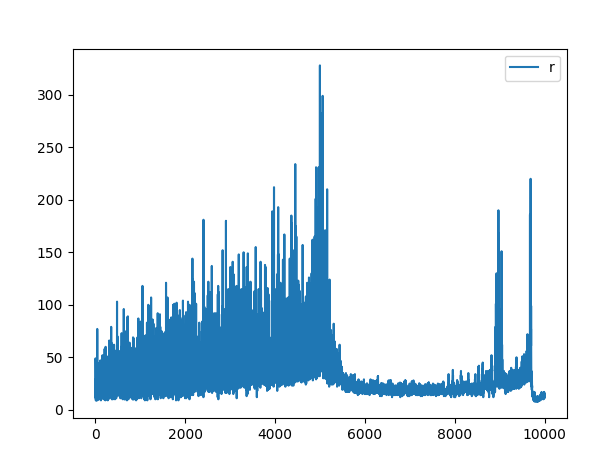
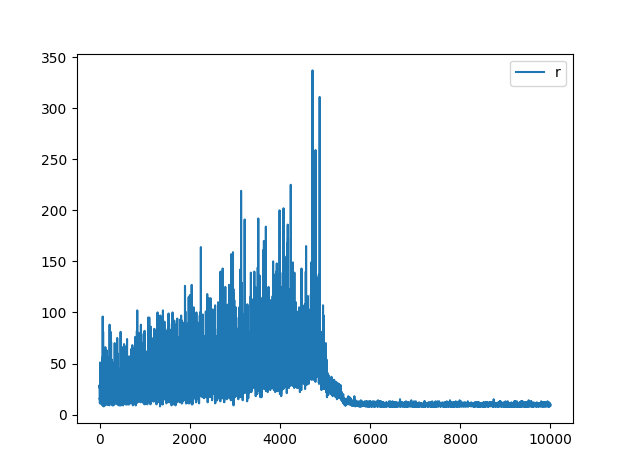
*| 0 | Cart Position | -4.8 | 4.8 |*  
*| 1 | Cart Velocity | -Inf | Inf |*  
*| 2 | Pole Angle | ~ -0.418 rad (-24°) | ~ 0.418 rad (24°) |*  
*| 3 | Pole Angular Velocity | -Inf | Inf |*

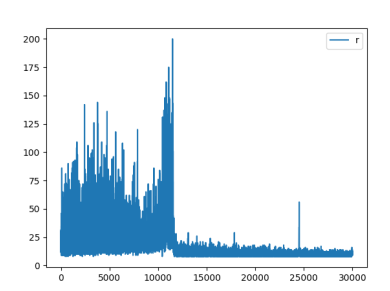
* Actions

*| 0 | Push cart to the left |*  
*| 1 | Push cart to the right |*

**Process**

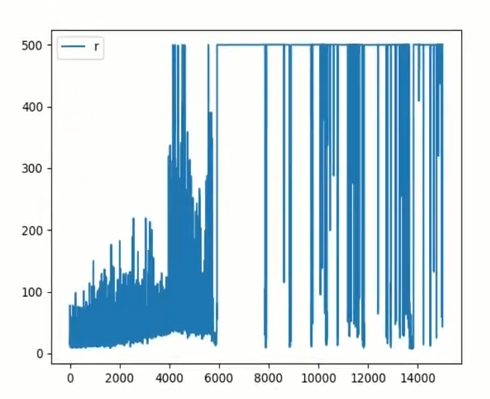
* First we observe that the robot was suffering from the catastrophic forgetting problem (https://en.wikipedia.org/wiki/Catastrophic\_interference). Typical of DQN algorithm beign overtrained, so we decided to make the episodes shorter (500 steps) with the same following result ilustrated in the following graph - cumulated reward per episode.





* Then we applied the epsilon discount in a different way.

Instead of multiplying it by a number between 0 and 1 each step, we decided to substract a low quantity instead, going from a logaritmic decrease to a linear decrease, which make the robot explore more at the beginning and less at the end and also stop learning before. We used this blog as inspiration (<https://blog.gofynd.com/building-a-deep-q-network-in-pytorch-fa1086aa5435>)



* It improved the results, but we still experienced the catastrophic forgetting randomly in some of our experiments, so we decided to test locking some of the experience\_replay buffer to unsuccessfull events stored during the initialization. Exactly, a 10% of the buffer, in this case, the results were always like the shown in the previous picture, achieving 100% of successfull episodes during inference time with the final trained brain.