**1. Introduction** A short paragraph explaining what is the aim of this notebook, and a specification of the team involved (at least: picture, name, surname, email)

2. **Case analysis**

You explain clearly how an example that is already implemented works. This includes at least an explanation of:

**In the default:**

a) what are the rewards

- The target arrives to the goal = +5.0

- Every frame that the target isn’t in the goal = (-1f / agentParameters.maxStep)

b) what are the states

Only has 1 state

c) how is, in general, the training implemented

With the movement of the agent, it gives a reward.

If casually, the agent push the target to the goal, the reward will be positive.

If the agent push the target faster to the goal, the reward will be higher, and the agent will learn a fastest way to put the target into the goal

**In the improved one from Map2 and Map3:**

a) what are the rewards

- The target arrives to the goal = +5.0f

- Every frame that the target isn’t in the goal = (-1f / agentParameters.maxStep)

- If the agent is at a max distance of 3.0f with the target, every frame = + 0.001f

- If the target is at a max distance of 1.0f with a wall, every frame = -0.01f

b) what are the states

Only has 1 state

c) how is, in general, the training implemented

With the movement of the agent, it gives a reward.

*To do a faster brain, there is a reward if the agent is near the target*

If the agent moves near the target, it will have a little reward, so it will be more possible that the target can move the block.

If casually, the agent push the target to the goal, the reward will be positive.

If the agent push the target faster to the goal, the reward will be higher, and the agent will learn a fastest way to put the target into the goal

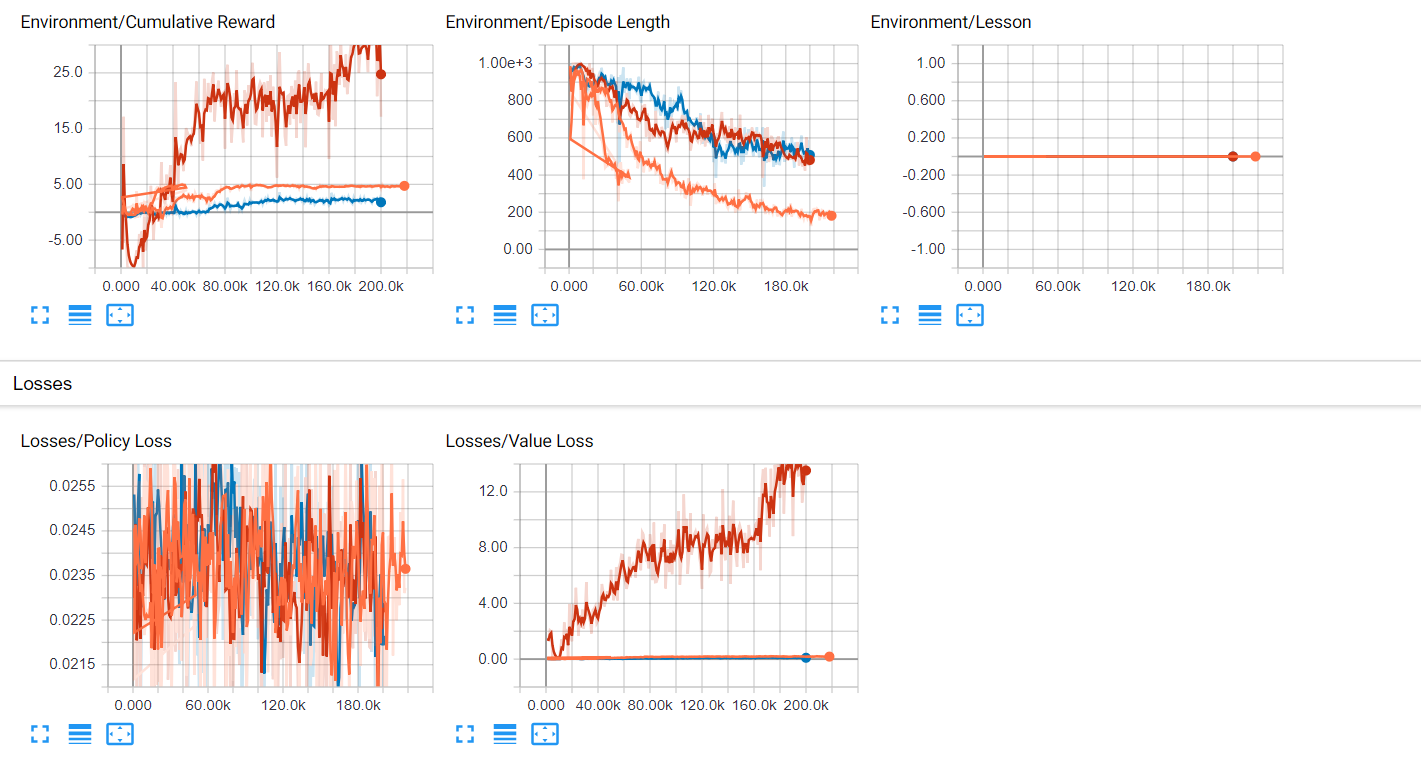
*In the default case, there was a problem, that sometimes, the block gets stuck into a corner. To evade this, in this case there is a negative reward if the target is near a wall.*

If the target is near a wall, the agent will have a little negative reward, because in future cases, the agent will not push the block near to walls and get the target stuck.

3. **Performance analysis**

You do a performance analysis of the training using TensorBoard, exploring what parameters are most critical to

The orange is with our code, the blue is with the default code and the red is the default code but with the time penalty at -10 (ten times bigger than the default) and the reward at 50 (ten times bigger than the default)



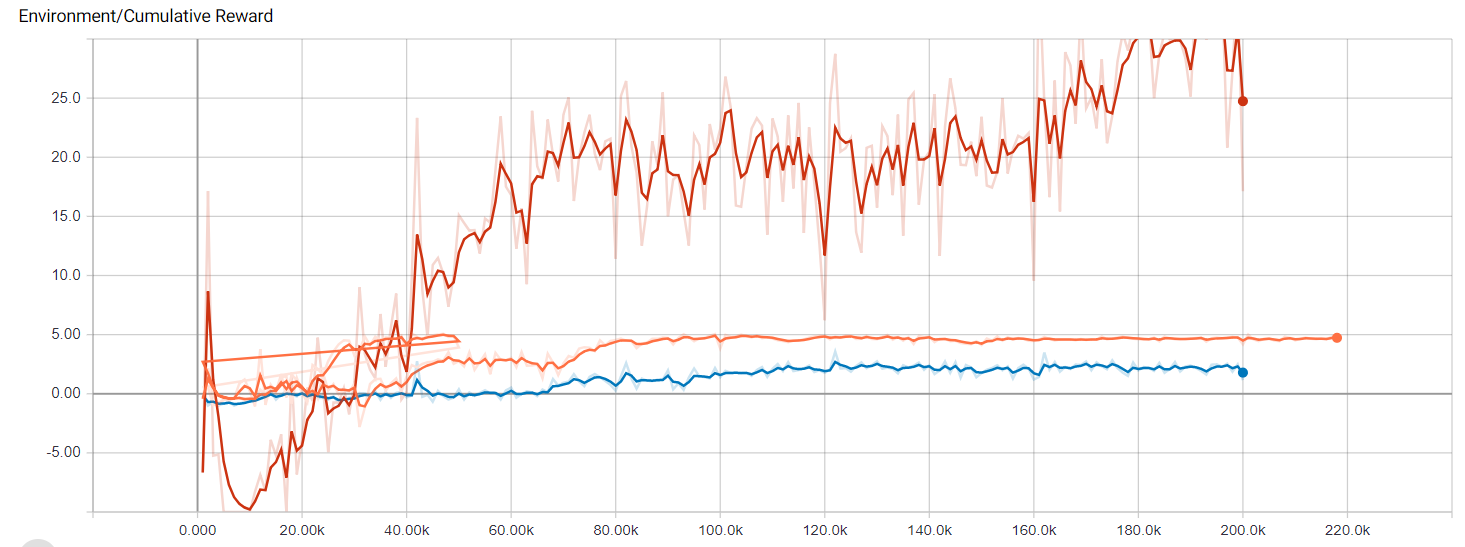
a) make it work

b) produce smooth results

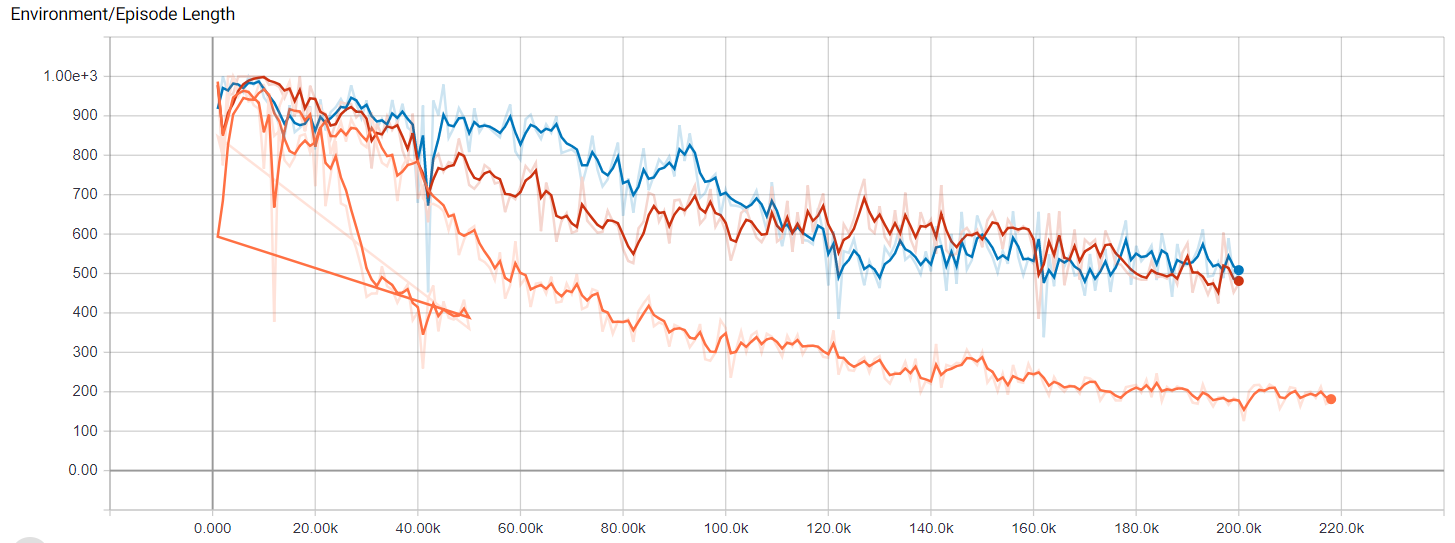
c) train fast.

In your analysis you should include a combination of parameters that does not work, and some that do work. You should also show some understanding of the role of the parameters, and try to explain why something works or doesn't.

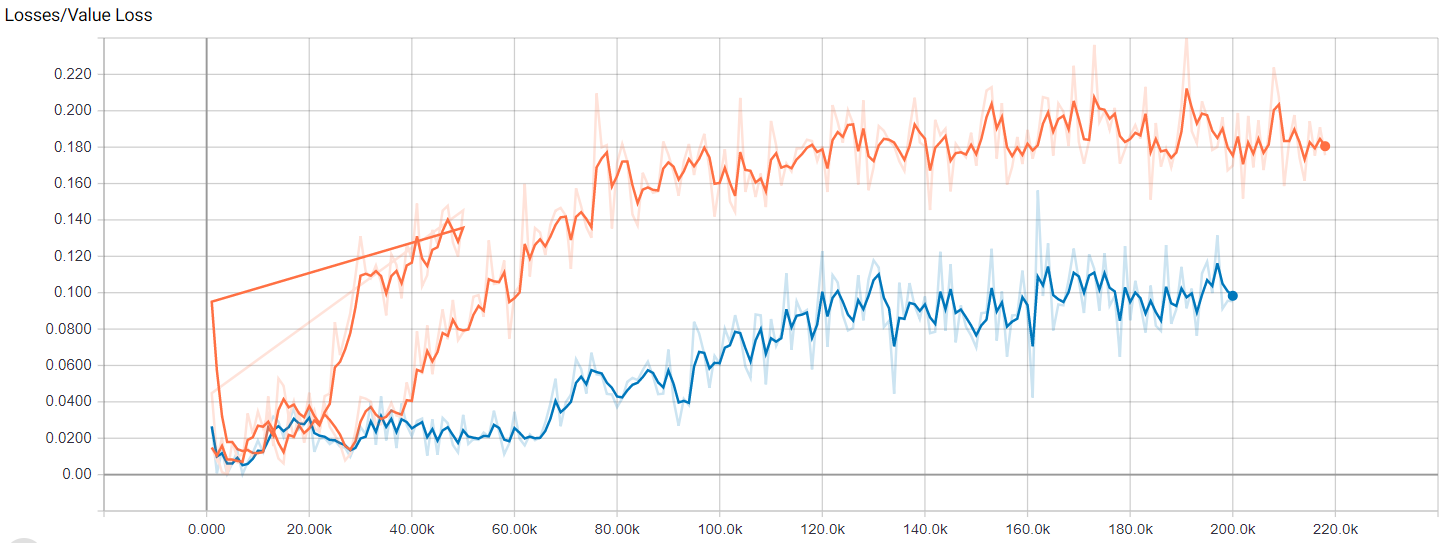
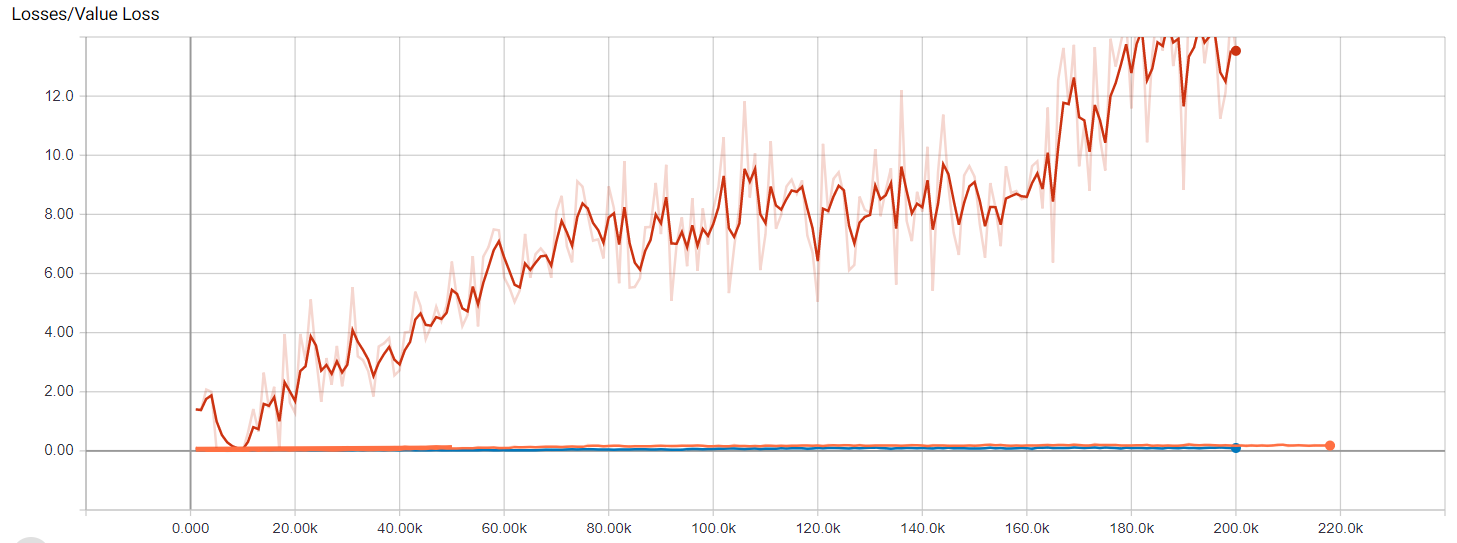
The orange has a better cumulative reward in long term, and the worst is the red by far



The orange is the one that have shorter episode length, so its better. The red and the blue are very similar.



The red have a lot more of loss value, it’s the worst by far. The orange and the blue have a similar loss value

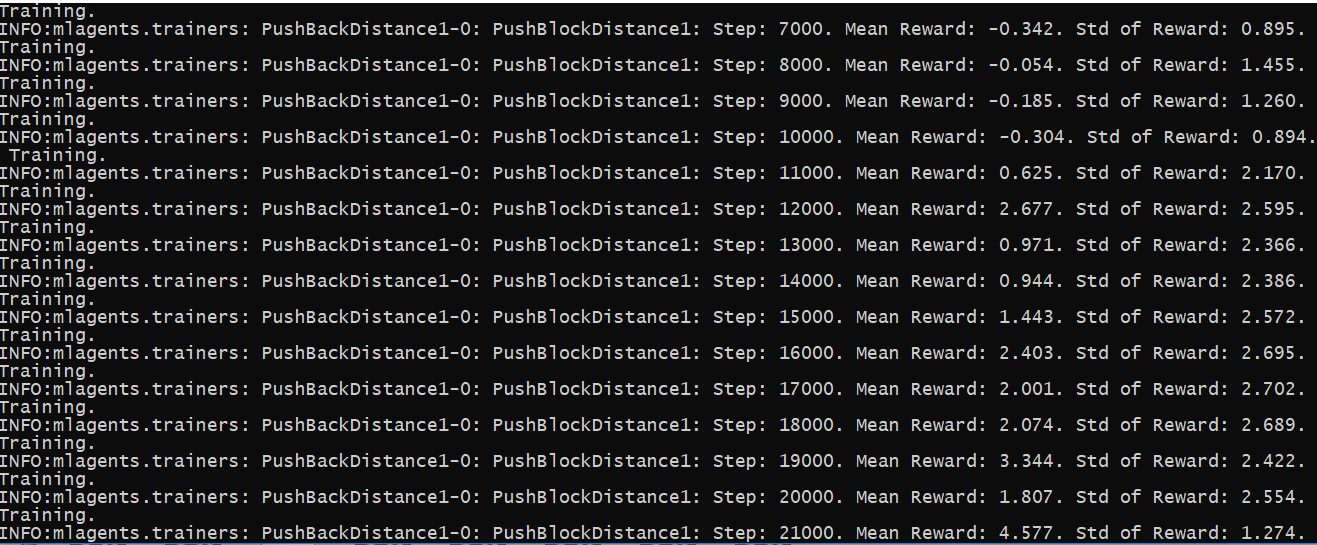


Cases:

Brain without learning to Brain learned

Caso 1: Sin nada, como viene

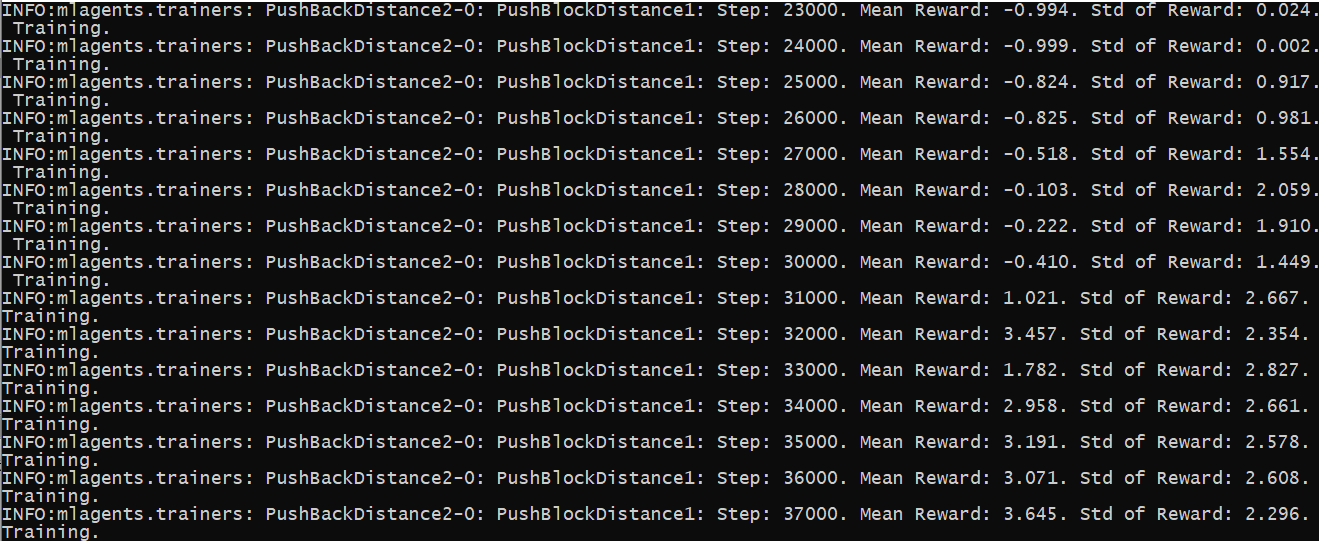
Caso 2: With distance in 0.05 and reward of 0.5 / agentParameters.maxStep



It learns in about 16.000 steps, more than the default.

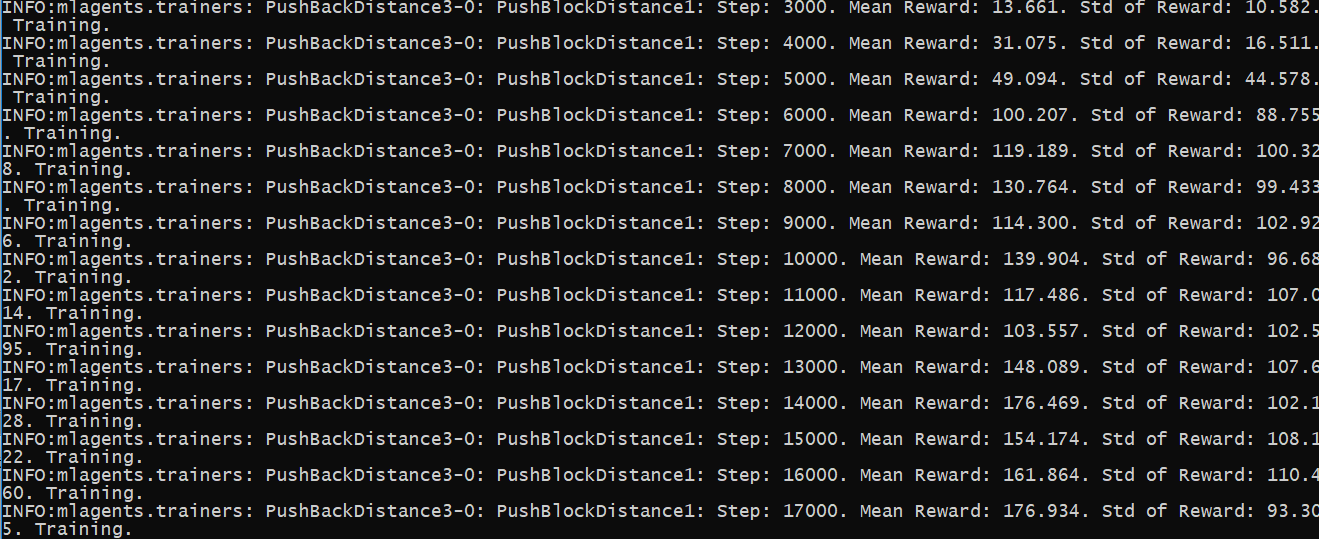
The distance is too short, so it needs to be bigger

Caso 3: With distance in 3.0f and reward of 0.5 / agentParameters.maxStep



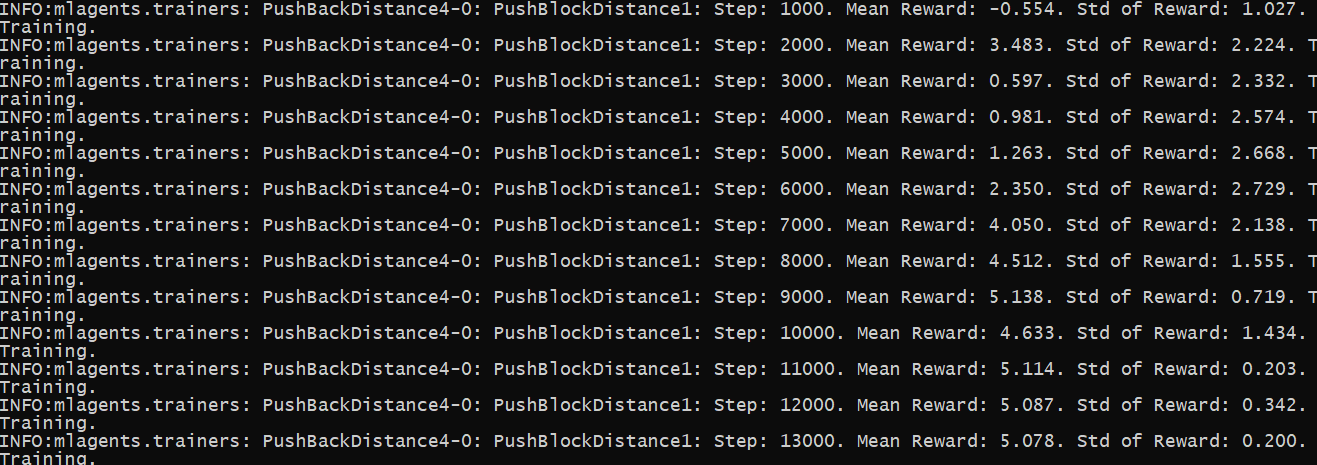
It learns in about 32.000 steps, a lot more than in other cases.

Caso 4: With distance in 3.0f and reward of 0.05



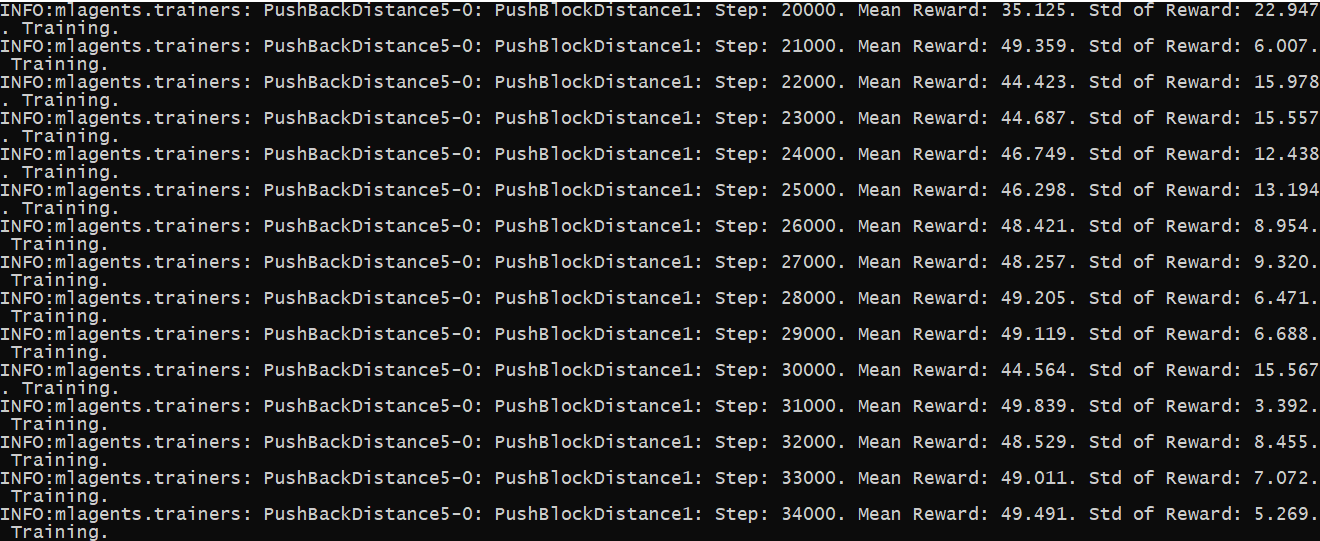
In this case, the agent learns in 14.000 steps, better than the other ones, but not as the default one

Caso 5: With distance in 3.0f and reward of 0.001 and reward to put the target into the goal of 5.f



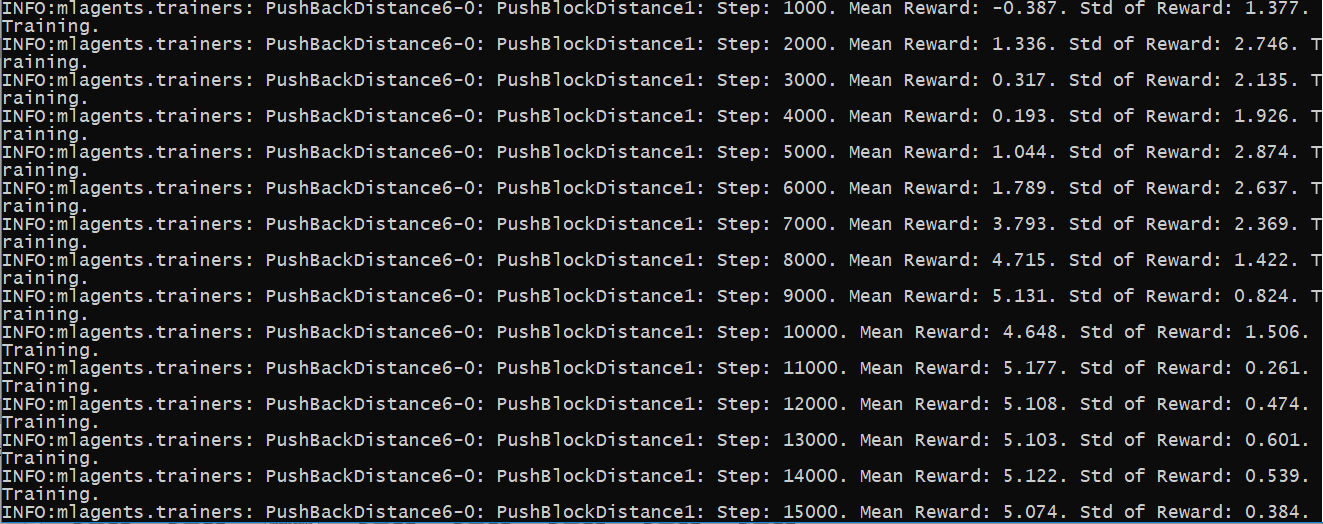
Here the agent learns in 11.000 steps, a bit better than the default

Caso 5: With distance in 3.0f, reward of 0.001 and reward to put the target into the goal of 50.f



In this case, the agent learns in 31.000 steps, very worts than the previous one

Caso 6: With distance in 3.0f, reward of 0.001, reward to put the target into the goal of 5.f and reward of -0.01f if the target is in 1.0f distance to a wall.



In this case, the agent learns in 9.000 steps, better than the default and the other ones. This is the fastest way to learn and the better

4. **New case proposal**

- How to train:

1. First, duplicate a brain, rename it if needed, and put the model to none

2. Put the brain in the academy and check the control box

3. Put the same brain into the GameObject Agent, inside PushAgentBasic, the part with Brain.

4. In the Anaconda Prompt, put activate ml-agents and press Enter

5. In the Anaconda Prompt put the direction of the folder ml-agents

6. Once you are inside that folder, put mlagents-learn config/trainer\_config.yaml --run-id=”NameOfTheLearning” --train

7. Press enter, wait a few seconds, and then, press the play button. With this, the agent will train and learn

**If you like to train from a launcher and not from the editor**

3.5 Create a .exe of the project

6. Once you are inside that folder, put mlagents-learn config/trainer\_config.yaml --env=FolderOfTheExecutable/Executable --run-id=”NameOfTheLearning” --train

- Rewards:

- The target arrives to the goal = +5.0f

- Every frame that the target isn’t in the goal = (-1f / agentParameters.maxStep)

- If the agent is at a max distance of 3.0f with the target, every frame = + 0.001f

- If the target is at a max distance of 1.0f with a wall, every frame = -0.01f

- States

- Only has the default state

Implementation:



In this map, there are a few walls inside the path. Two are vertical and they aren’t a big problem, but the other cuts the map in 2 parts, putting difficulty to put the target into the goal.