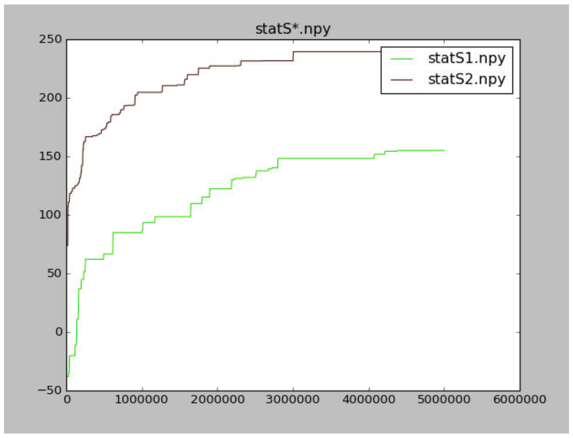
Exercise 4

Task description

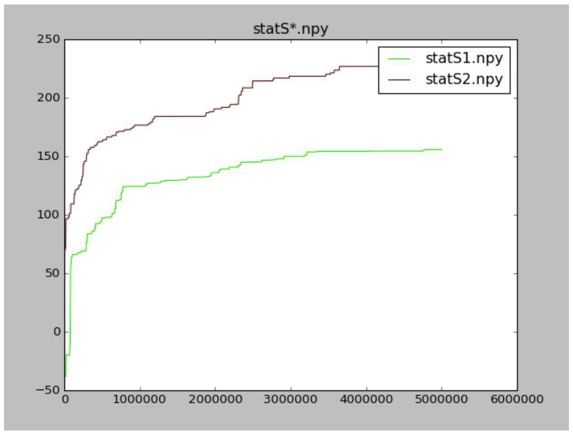
Evolve the robots with the original reward functions and then compare the behavior of robots evolved with the original and revised reward functions, e.g. in the case of the hopper and halfcheetah. Could you explain why the original rewards functions are not suitable for evolutionary strategies?

Results

On Fig. 1 you can see the results of training two models on the original and custom rewards. Seed 1 was intended for halfsheetah model, the seed 2 was for hopper model.



(a) Original reward



(b) Custom reward

Figure 1: Results of exercise 4

Hopper

The task of hopper model (see Fig. 2) is: Make a two-dimensional one-legged robot hop forward as fast as possible. In original version as reward function using a position or robot on x-axis. But this type of reward does not take into account the potential energy of the robot. The potential energy of the jump is more important than just the position of the robot, since a dynamic system can operate on internal energy. From a long-term perspective, the custom reward model is better suited to this task, since it uses the fundamental properties of the motion of dynamical systems. For longer timesteps, it should converge to the best result faster.

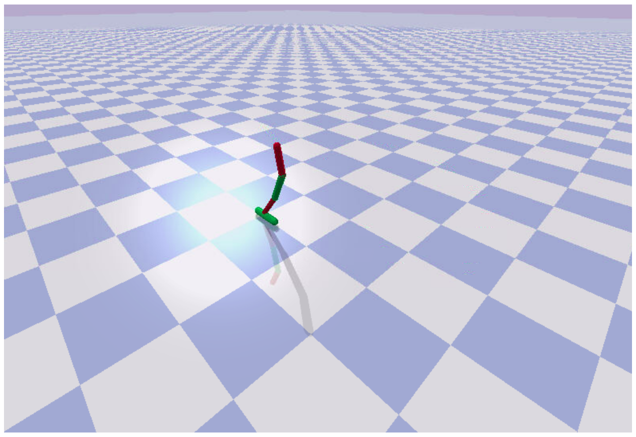


Figure 2: Hopper

Halfsheetah

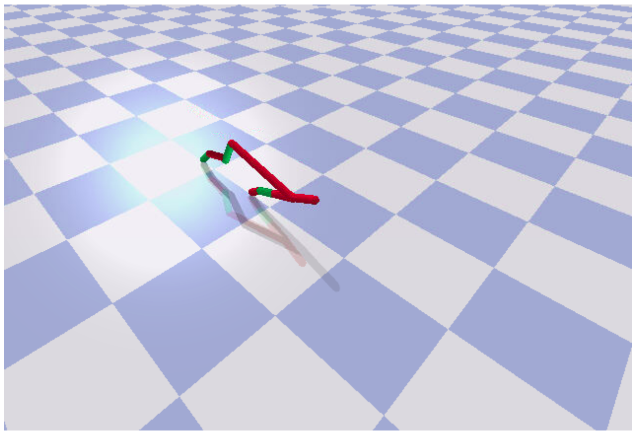


Figure 3: Halfsheetah

For the halfsheetah model (see Fig. 3), the situation is the same as for the hopper. In the original rewind, only the position of the robot relative to the old position (the speed of movement along the x axis) is taken into account, but the energy component is not taken into account.

Exercise 5

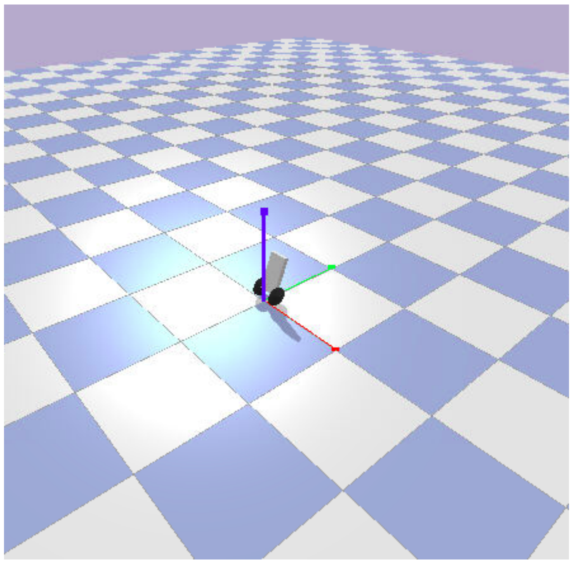
Task description

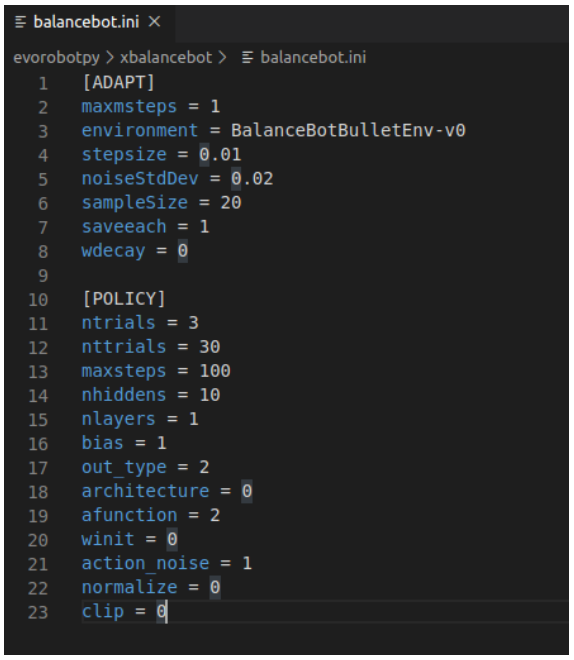
Creating and using a new Gym environment from scratch.

The problem concerns a simple wheeled robot that can be trained for the ability to balance. The model name is balance-bot.

Results

The results of the work done can be seen on Fig. 4


(a) Robot simulation



```
[ADAPT]
maxmsteps = 1
environment = BalanceBotBulletEnv-v0
stepsize = 0.01
noiseStdDev = 0.02
sampleSize = 20
saveeach = 1
wdecay = 0

[POLICY]
ntrials = 3
nttrials = 30
maxsteps = 100
nhiddens = 10
nlayers = 1
bias = 1
out_type = 2
architecture = 0
afunction = 2
winit = 0
action_noise = 1
normalize = 0
clip = 0
```

(b) .ini-file for training

Figure 4: Results of exercise 5

Exercise 6

Task description

Run 10 replications of the experiment from the evorobotpy/xdiscrim folder by using different seeds. To speed-up the training process you can run 2 instances of the program in parallel or eventually more, depending on the characteristics of your computer. Test and analyze the strategy displayed by best robot of each replication. Describe the strategies of the robots by grouping them in families. Try to explain why the robot of each family behave in that manner. Run other experiments by using a feed-forward neural architecture (without memory). Explain how the behavior of evolved robots differ from those evolved with the LSTM architecture (i.e. the Long Short Term Memory architecture).

Results

On the Fig. 5 you can see the results of exercise 6. Seeds from 1 to 10 is for LSTM Neural network. Seeds from 11 to 13 for feed forward neural network.

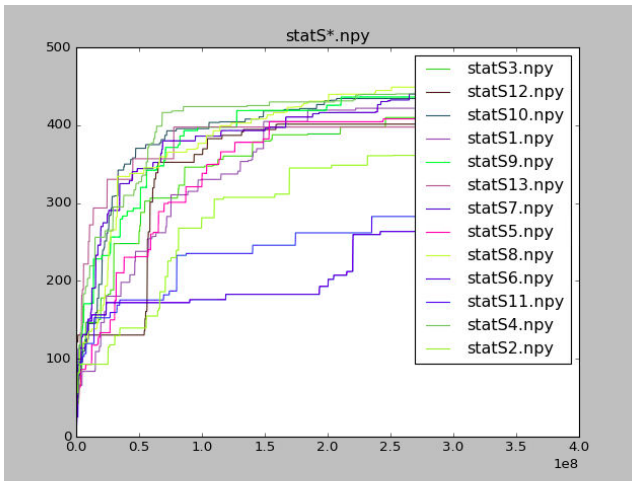


Figure 5: Results of exercise 6

Traditional neural networks do not have short-term memory. Therefore, for them, each new step does not take into account the previous state. After simulating the results, it was revealed that those cases that used LSTM neural networks - quickly and most smoothly reach the target point. The LSTM neural networks adapt more faster to the new goal function than feed forward neural network.