# MicroRacer - PPO

## Proximal Policy Optimization

Machine Learning

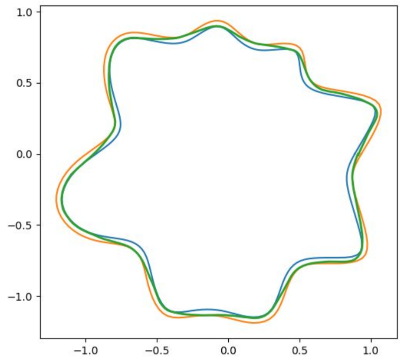
a.y. 2020/2021

*Salvatore Fiorilla* **931332**

*Marianna Corinaldesi* **946229**

# Project goal

Dive into Deep Reinforcement Learning by implementing PPO algorithm and evaluating its performances on MicroRacer environment



# In-depth topics

- Tensorflow
- Keras
- Numpy
- Policy based algorithms

# Challenges

- Math of PPO on continuous environment
  - Actions sampling on Truncated GD.
  - Adapting math of PPO properly.
- Debugging the agent
  - Use static track and lots of logs.
- Hardware limits
- Tuning hyperparameters
  - Experimental evaluation.

# Reinforcement Learning 1/2

$s_t$

$a_t$

$(s_{t+1}, r_t)$

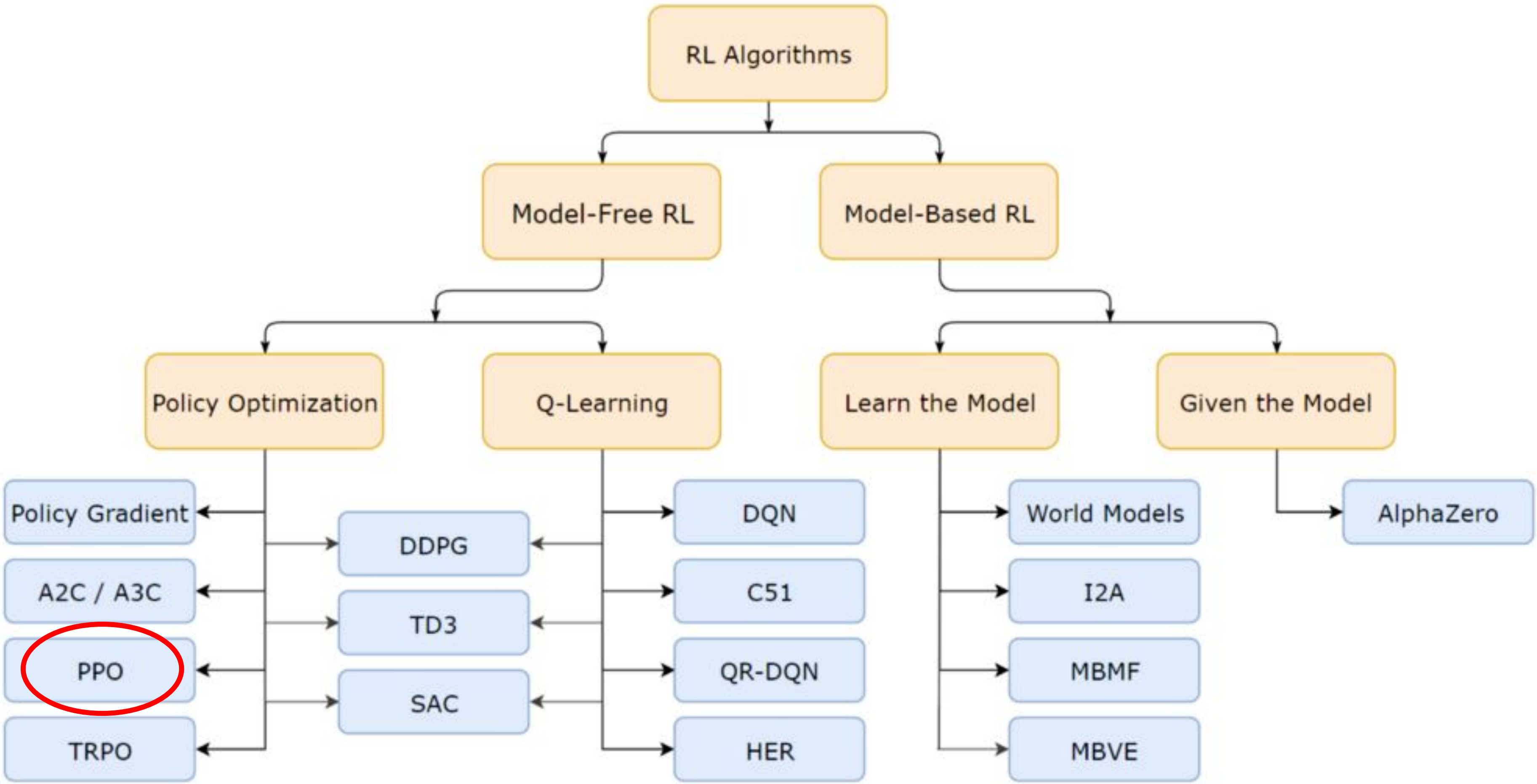
$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{j=0}^{T} \gamma^j r_{t+j+1}$$

$$V_\pi(s) = \max_a Q(s,a)$$

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s = s_t] = \mathbb{E}_\pi\left[\sum_{j=0}^{T} \gamma^j r_{t+j+1} \,|s = s_t\right]$$

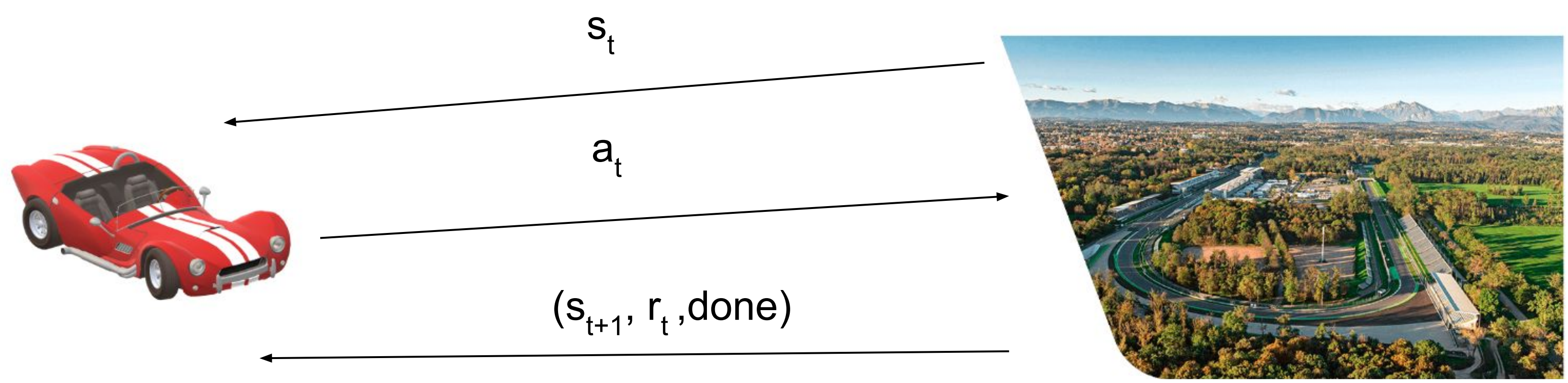
$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{j=0}^{T} \gamma^j r_{t+j+1} \,|S_t = s, A_t = a\right]$$

# Reinforcement Learning 2/2




Reinforcement Learning taxonomy as defined by OpenAI [Source]

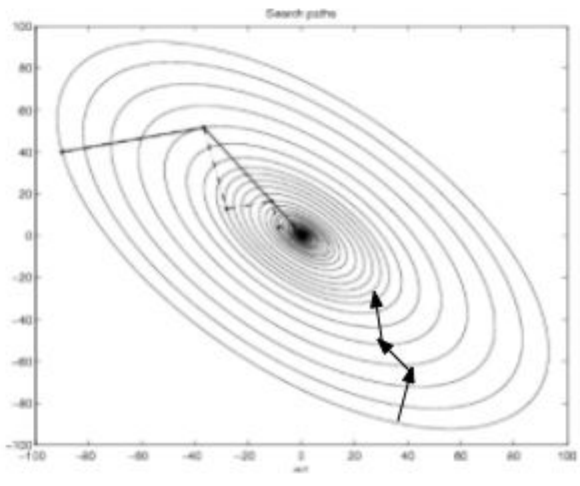
# Policy Based Model-Free



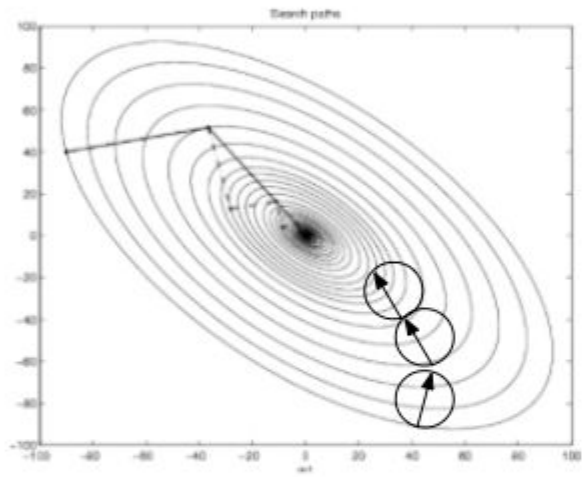

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t)$$

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\right].$$

# TRPO




LINE SEARCH METHOD




TRUST REGION METHOD

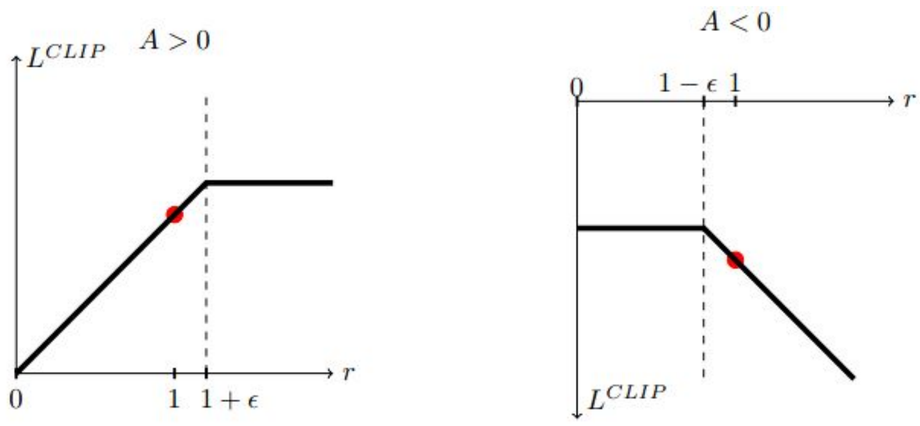
$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[ \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[ r_t(\theta) \hat{A}_t \right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

# PPO

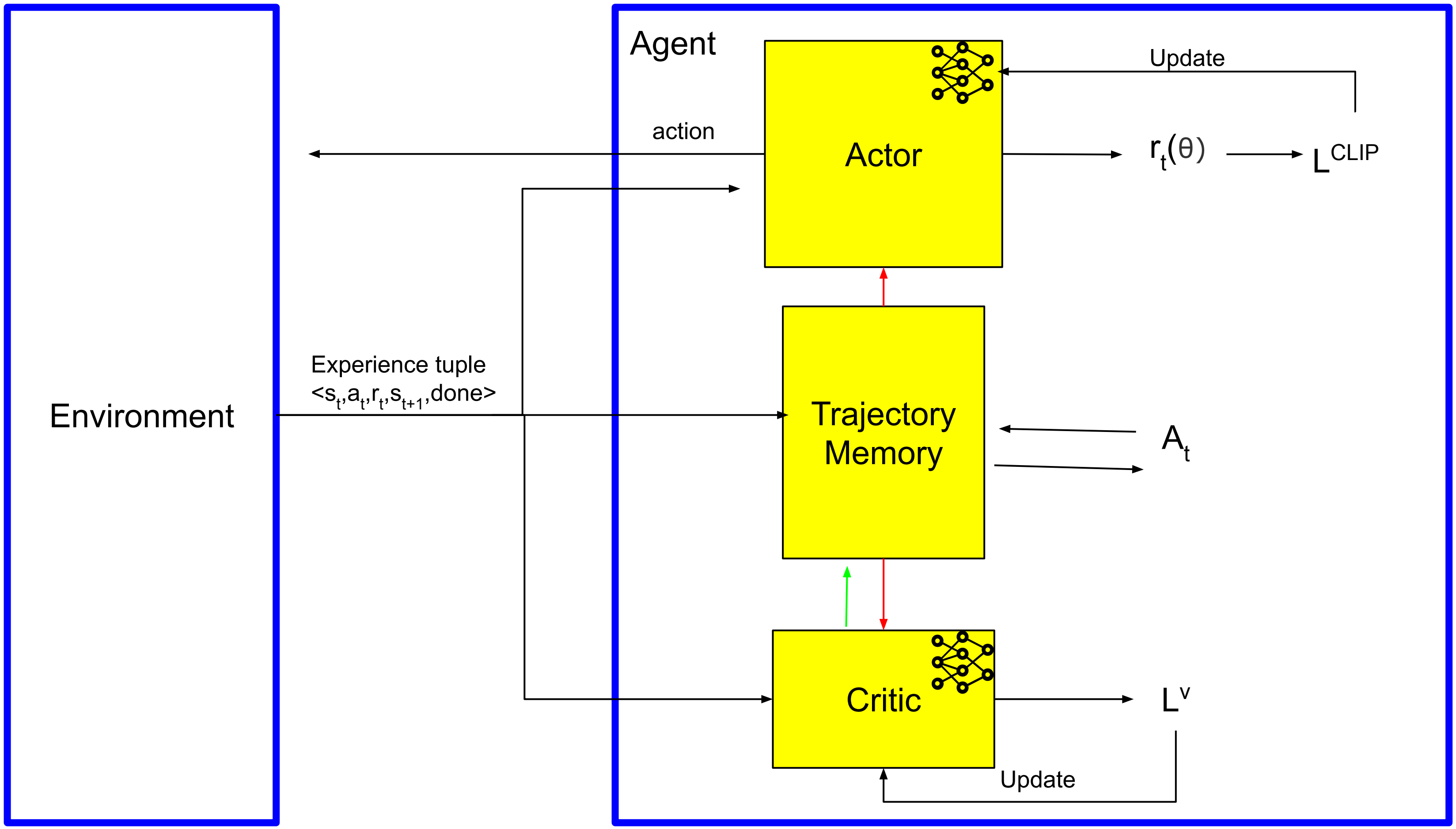
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$




$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

# Deep into code

Actor starts with initial state $s_0$ which is obtained from environment.reset() stmt.



# Benchmarks

We trained for

**200** epochs, **4096** steps per epoch and **50** update iteration

with hyperparameters

Actor_lr 3e-5

Critic_lr 1e-4

clip_value 0.1

early stopping 0.0225

**different "Actor - Critic" Networks' architectures**

# Benchmarks 1/4

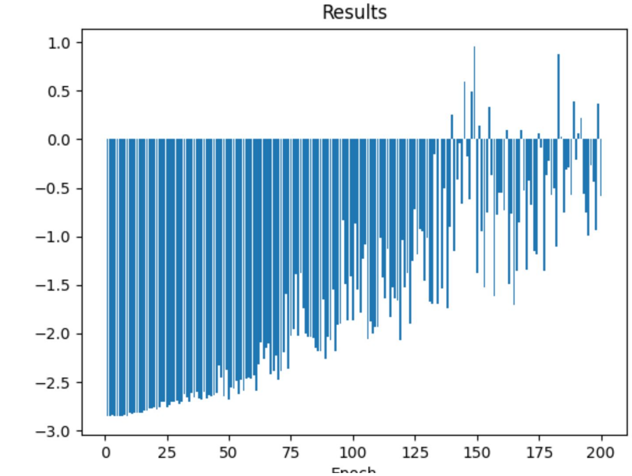
**CRITIC: 3 dense, ACTOR: 4 dense**

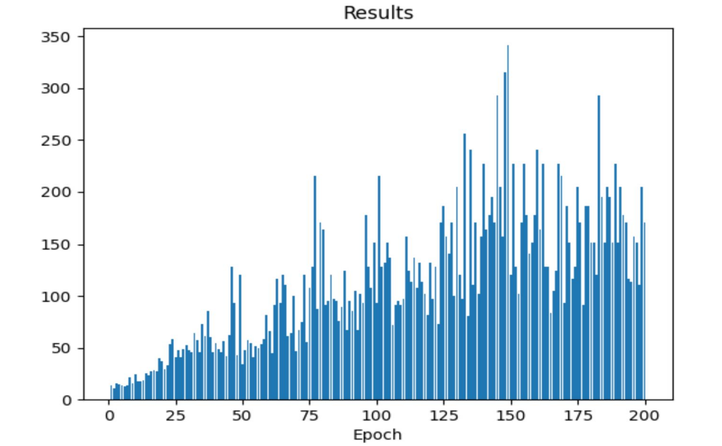
- 32 units
  - training time: **4h 15 m**
  - first completed path at epoch : **never completed**
- 128 units
  - training time: **1h 23 m**
  - completed : **18/100**
  - In **100** races:
    - mean length trajectories: **166**
    - mean rewards: **-0.52**
  - first completed path at epoch: **62**
- 512 units
  - training time: **2h**
  - completed : **9/100**
  - In **100** races:
    - mean length trajectories: **133**
    - mean rewards: **-0.65**
  - first completed path at epoch: **76**

# Benchmarks 2/4

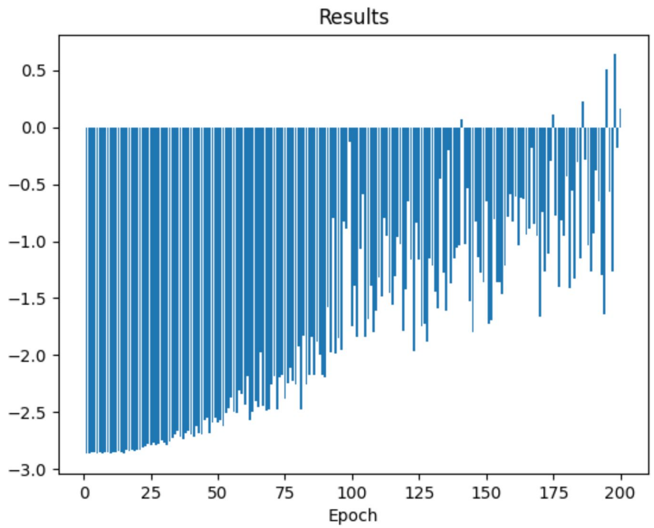
MEAN REWARDS

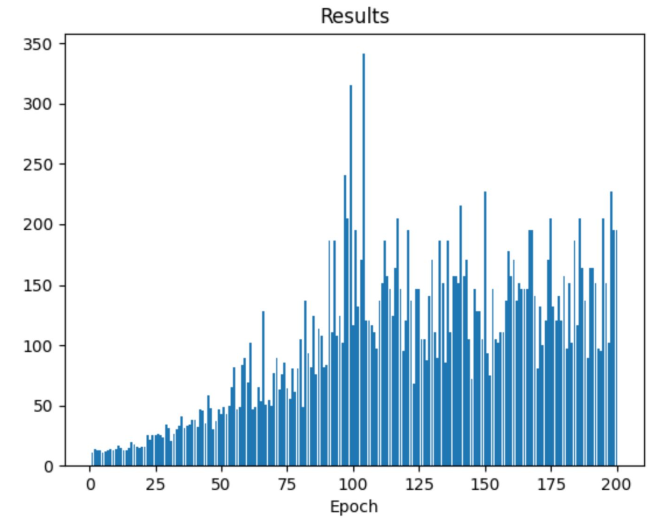
MEAN TRAJECTORIES LENGTH




128 units




512 units

# Benchmarks 3/4

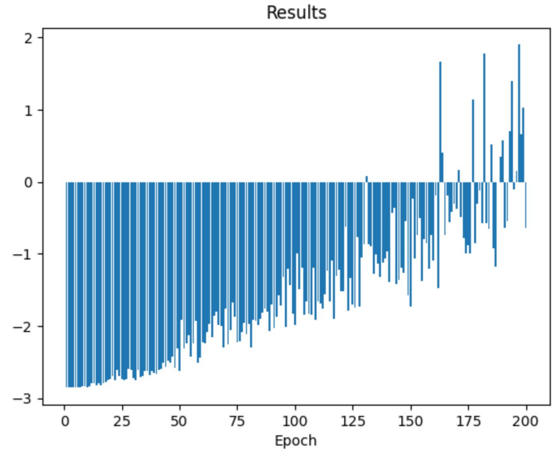
**CRITIC: 3 dense, ACTOR: two towers of 3 dense layers each**

- 32 units
  - training time: **killed after 2 hours of no learning**
  - first completed path at epoch: **never completed**
- 128 units
  - training time: **1h30 m**
  - completed: **22/100**
  - In **100** races:
    - mean length trajectories: **164**
    - mean rewards: **0.249**
  - first completed path at epoch: **54**
- 512 units
  - training time: **1h 50m**
  - completed: **11/100**
  - In **100** races:
    - mean length trajectories: **161**
    - mean rewards: **-0.219**
  - first completed path at epoch: **65**

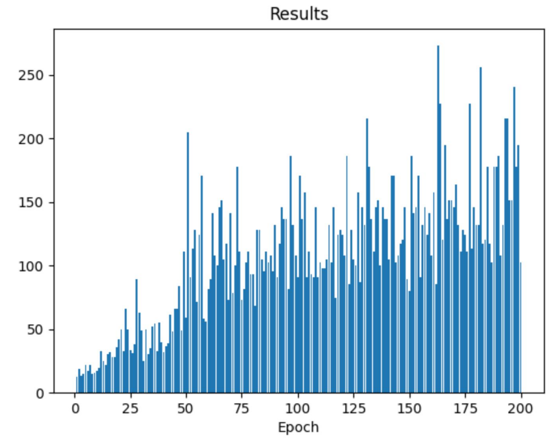
# Benchmarks 4/4

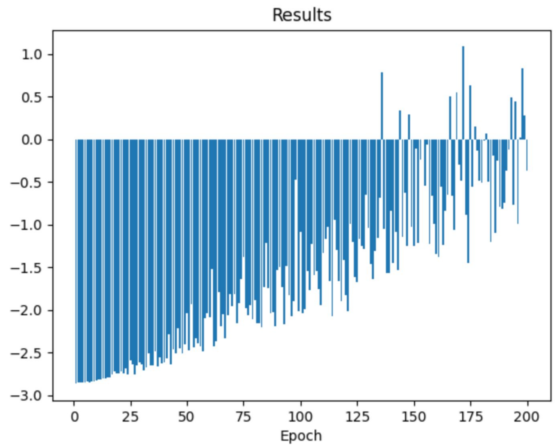
MEAN REWARDS

MEAN TRAJECTORIES LENGTH

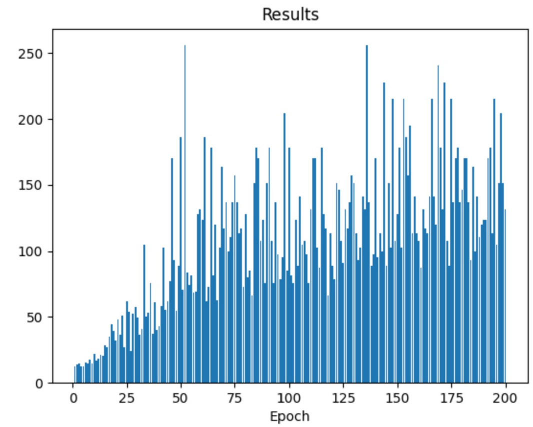



128 units




512 units



# Conclusion

Two tower solution is always better for the same units number.

The model performs generally better for 128 neurons than 512

link to test

# PPO vs DDPG

PPO get optimization, here the

comparison of the models on 100 races…

| | Mean Rewards | Mean Trajectories Length | Completed Path |
|---|---|---|---|
| PPO | 3.8460845947265625 | 228 | 64/100 |
| DDPG | 4.120152473449707 | 152 | 63/100 |

# Future Works

- Make a race between models
- Explore other DRL algorithms

# References

repo code

MariannaCor/MicroRacer_Corinaldesi_Fiorilla (github.com)

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Salvatore Fiorilla
Marianna Corinaldesi