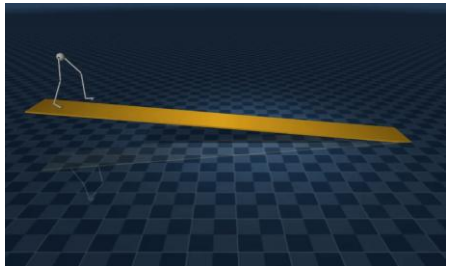
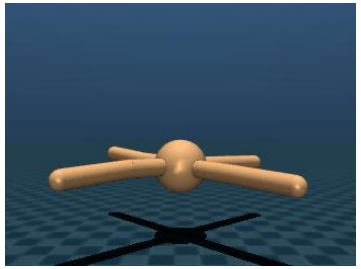
Overview Challenges



Introduction

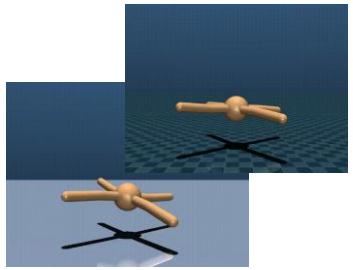
Evolutionary Strategies
Passive Dynamic Walker



Challenge 1

Submission Deadline:
19.03.2026

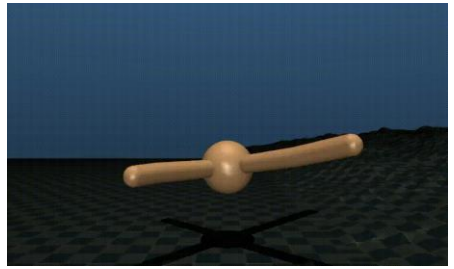
Evolving Neural Controllers
Ant – Flat Terrain



Challenge 2

Submission Deadline:
02.04.2026

Multi-objective Optimization
Ant – Ice and Flat Terrain



Challenge 3

Submission Deadline:
23.04.2026

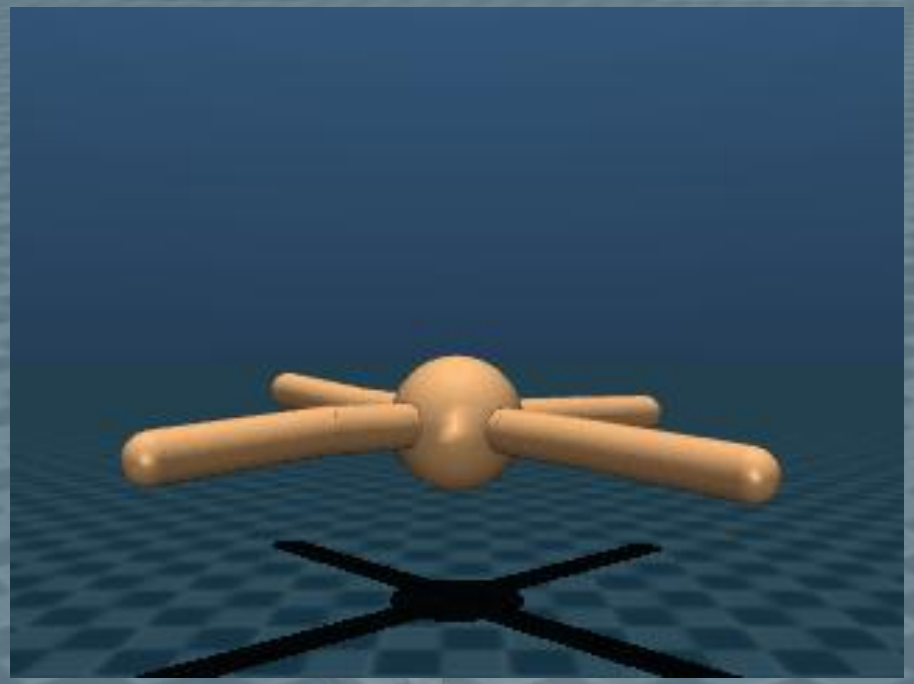
Evolving Robot Body
Ant – Hill Climbing

Final Project

Poster Presentation:
28.05.2026

Your Algorithm ☺
Multi-terrain Obstacles

Challenge 1
Objectives



- Gymnasium / Gym Convention for Learning Environments
- Neural Network Implementation (Feedforward, Oscillatory)
- Integration of State-of-the-Art Evolutionary Algorithm Libraries
- Comparison to Deep Reinforcement Learning

Challenge 1
Gymnasium Convention

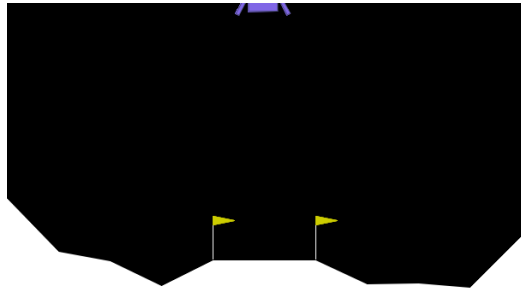
Gymnasium / Gym Convention for Learning Environments

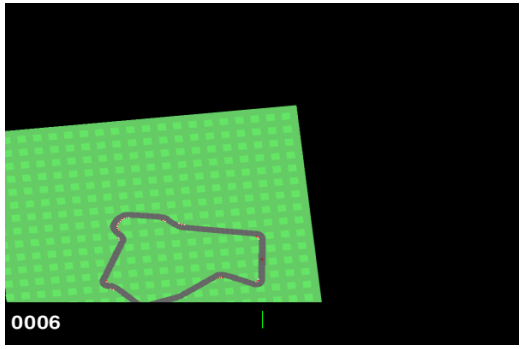
Gymnasium is the de-facto standard library for any reinforcement learning / embodied learning research and application.

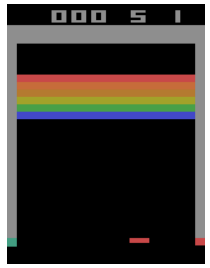
Standard for Learning Environments

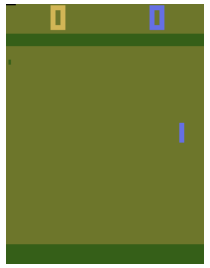
```python
import gymnasium as gym

# Initialise the environment
env = gym.make("LunarLander-v3", render_mode="human")

# Reset the environment to generate the first observation
observation, info = env.reset(seed=42)
for _ in range(1000):
    # this is where you would insert your policy
    action = env.action_space.sample()

    # step (transition) through the environment with the action
    # receiving the next observation, reward and if the episode has terminated or truncated
    observation, reward, terminated, truncated, info = env.step(action)

    # If the episode has ended then we can reset to start a new episode
    if terminated or truncated:
        observation, info = env.reset()

env.close()
```

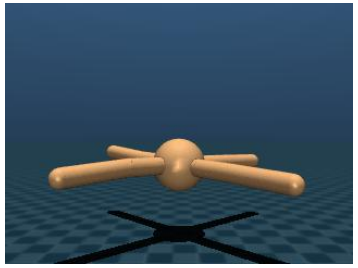
Benchmark Problems







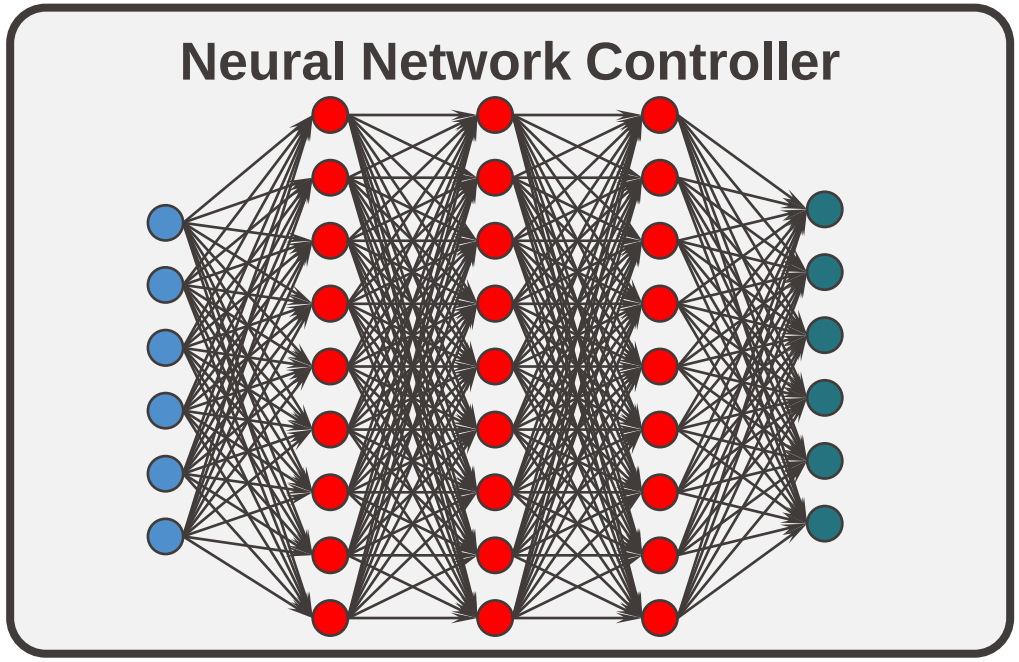




Challenge 1
Neural Network Implementation

Neural Network Implementation (Feedforward, Oscillatory)

Gradient-free neural networks are easy to implement!



Equations:

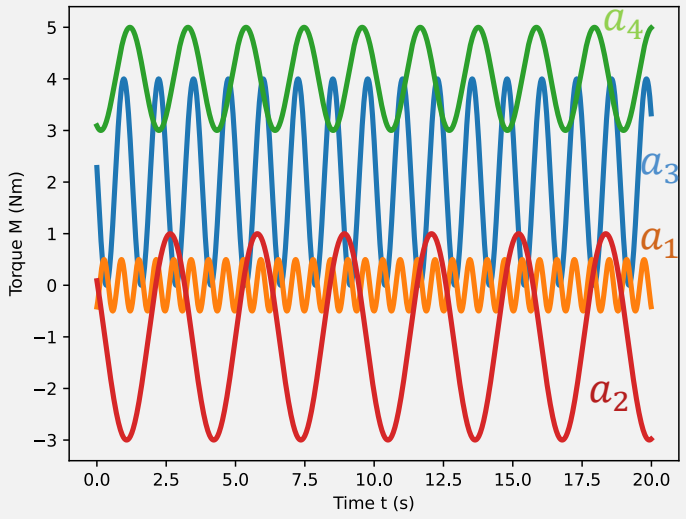
$$x^0 = \text{sensor observation}$$
$$x^{l+1} = \tanh\left(W^l \cdot x^l + b^l\right)$$
$$\text{action} = x^L$$

Challenge 1
Neural Network Implementation

Neural Network Implementation (Feedforward, Oscillatory)

With evolutionary algorithms you do not need a differentiable or feedforward network structure!

Oscillatory Controller



Equations:

$$a_1^t = w_{1,1} \cdot \sin\left(w_{1,2} \cdot t - w_{1,3}\right) + w_{1,4}$$
$$\vdots$$
$$a_7^t = w_{7,1} \cdot \sin\left(w_{7,2} \cdot t - w_{7,3}\right) + w_{7,4}$$

Challenge 1
Evolutionary Algorithm Libraries

Integration of State-of-the-Art Evolutionary Algorithm Libraries

The **ask-tell** interface is a popular convention for evolutionary algorithms, applied by many state-of-the-art libraries.

```python
class PowerfulEvolutionaryAlgorithm:
    def ask(self):
        # Generates a new population of solutions given internal statistics
        # updated by tell()
        ...
        return solutions

    def tell(self, solutions, scores):
        # Receives previous populations (solutions) and fitness (scores)
        # and updates the internal statistics. One of this internal statistics
        # could be e.g., mean-vectors or an archive of parents.
        ...
```

`ask()` → ask for new population based on current internal heuristics of algorithm

`tell()` → tell algorithm the fitness valvues of current solutions and updates internal heuristics

Challenge 1
Evolutionary Algorithm Libraries

Integration of State-of-the-Art Evolutionary Algorithm Libraries

pycma

pycma
Maintainer: Nikolaus Hansen (INRIA)
Features: Highly optimized **CMA-ES** implementation.
https://github.com/CMA-ES/pycma

evosax
Maintainer: Robert T. Lange (TU Berlin)
Features: GPU-accelerated and jit-compiled implementation of 30+ evolution strategies.
https://github.com/RobertTLange/evosax

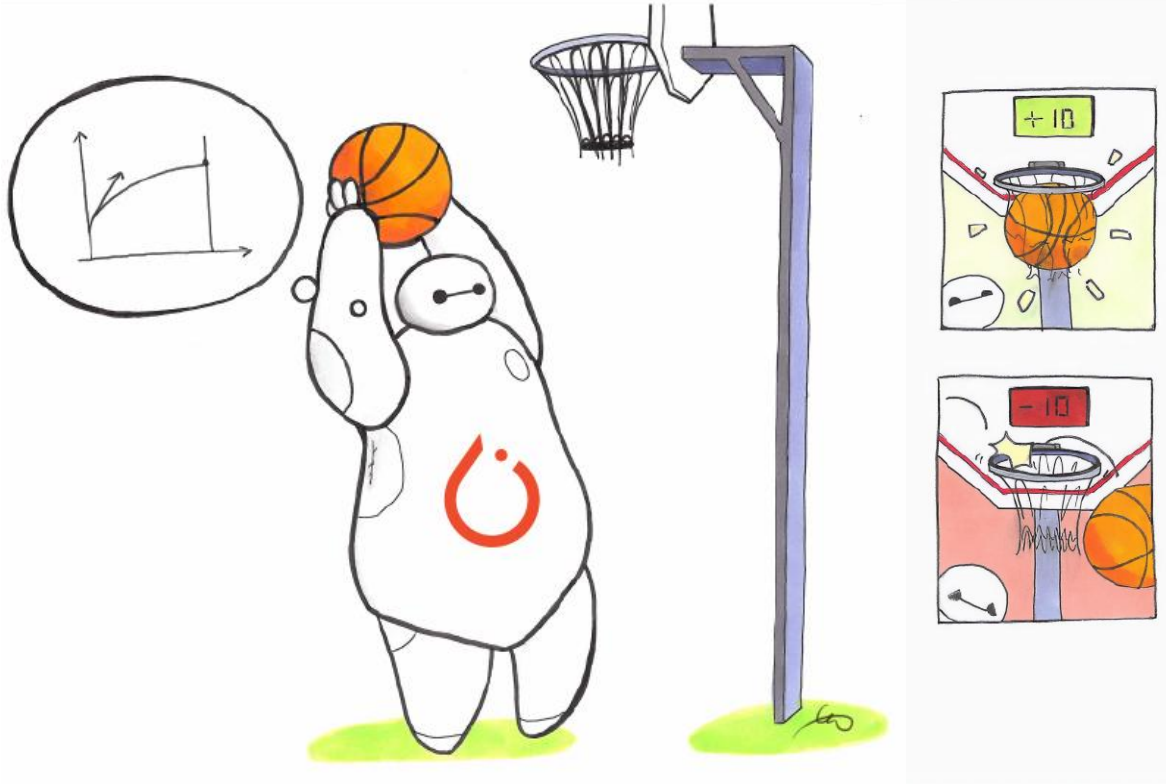
pyribs

pyribs
Maintainer: Interactive and Collaborative Autonomous Robotics lab (USC)
Features: High quality implementation of Quality Diversity algorithms.
https://github.com/icaros-usc/pyribs/

QDax
Maintainer: Adaptive & Intelligent Robotics Lab (Imperial)
Features: GPU-accelerated and jit-compiled implementation of QD algorithms.
https://github.com/adaptive-intelligent-robotics/QDax

Challenge 1
Reinforcement Learning

Comparison to Deep Reinforcement Learning

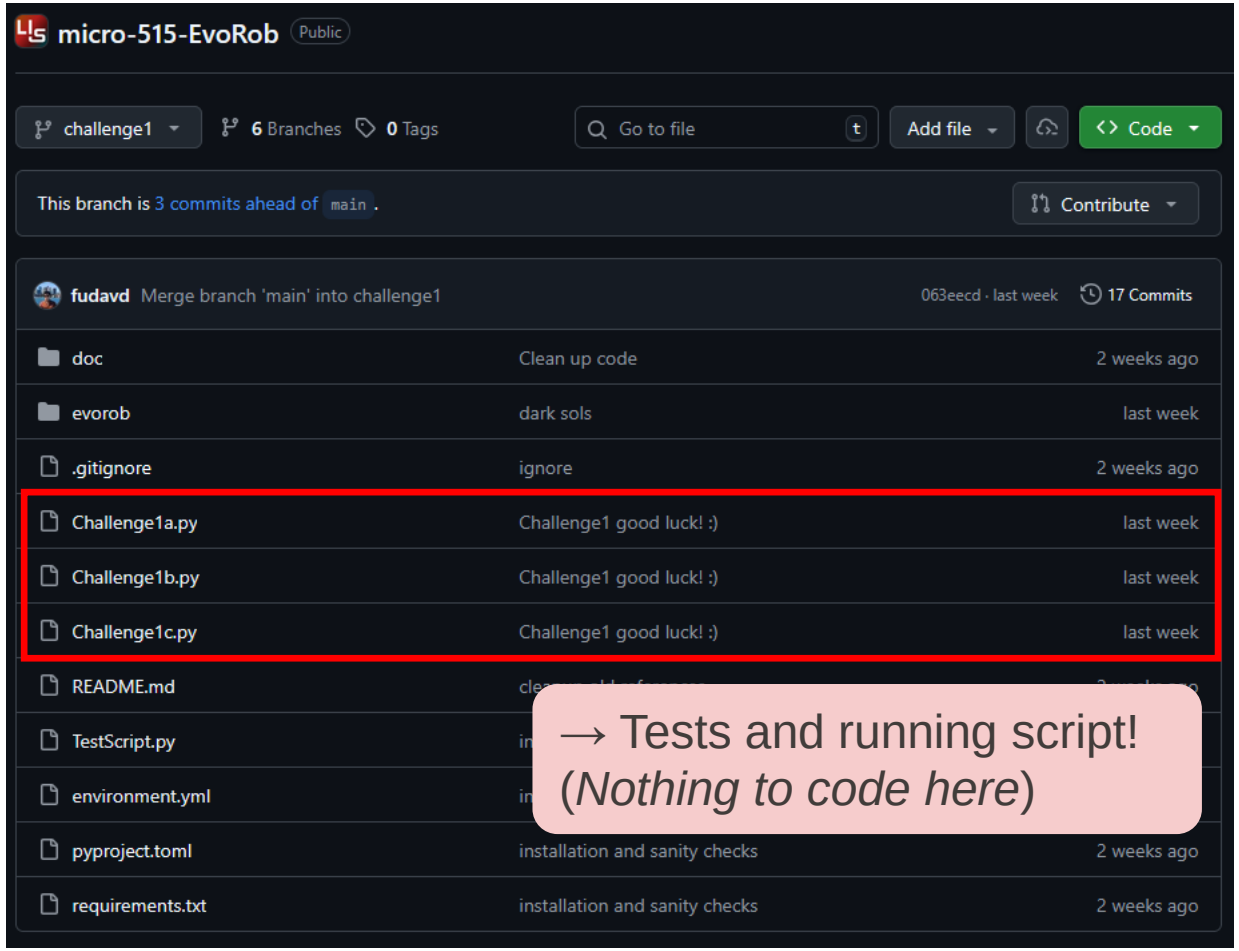


We can use your learning environment also with state-of-the-art reinforcement learning libraries, such as stable-baselines3.

https://github.com/DLR-RM/stable-baselines3

How does your evolutionary algorithm perform against **PPO**?

Challenge 1
Exercise Structure

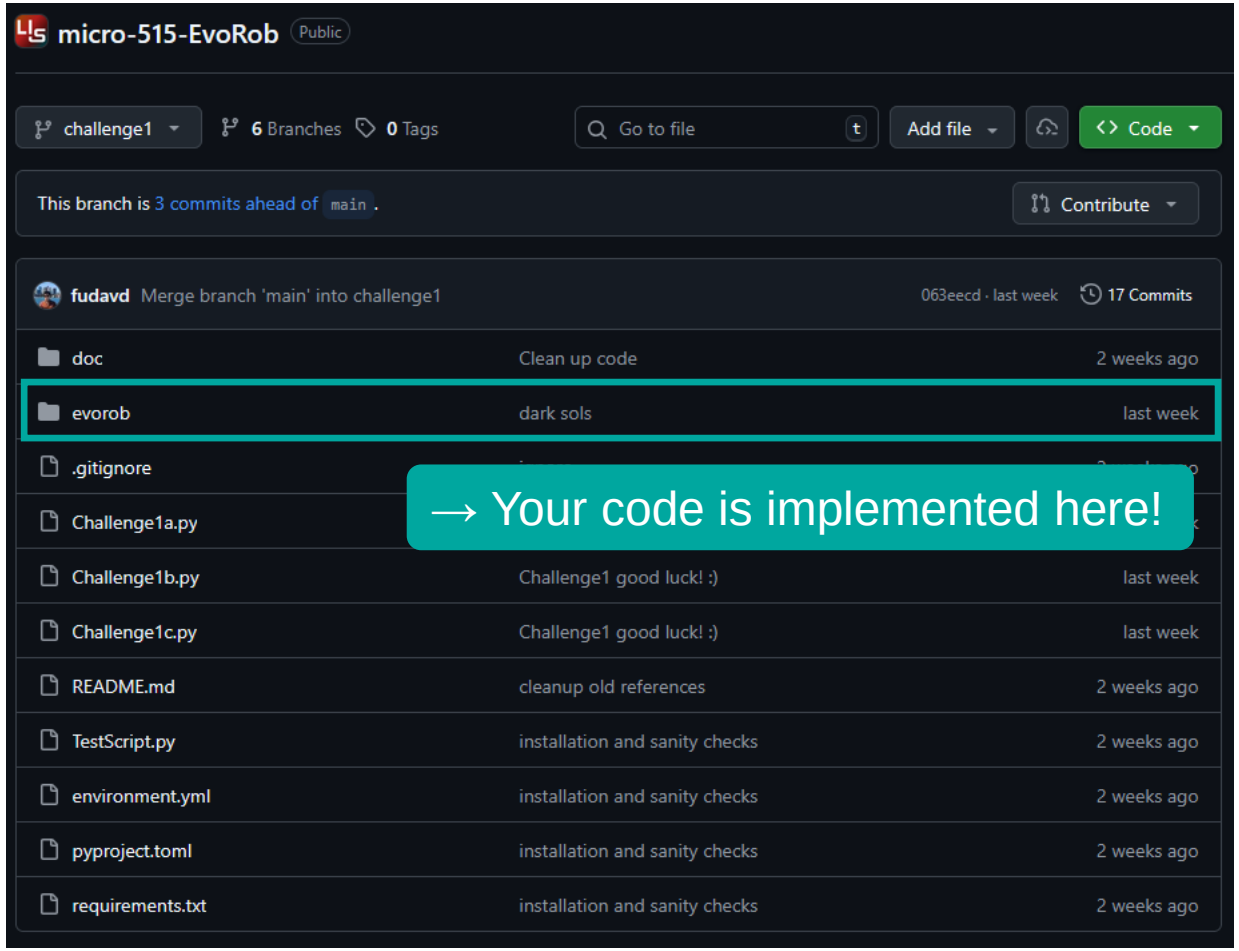


Content - Challenge 1a:
- Implement Learning Environment for Ant
- Implement Neural Controller
- Integrate Evolutionary Algorithm Library of your choice!

Content - Challenge 1b:
- Implement Oscillatory Controller

Content - Challenge 1c:
- Compare against Reinforcement Learning

Challenge 1
Exercise Structure



To-Do's - Challenge 1a:
- evorob / world / envs / ant_flat.py
- evorob / world / robot / controllers / mlp.py
- evorob / algorithms / ea_api.py

To-Do's - Challenge 1b:
- evorob / world / robot / controllers / sinoid.py

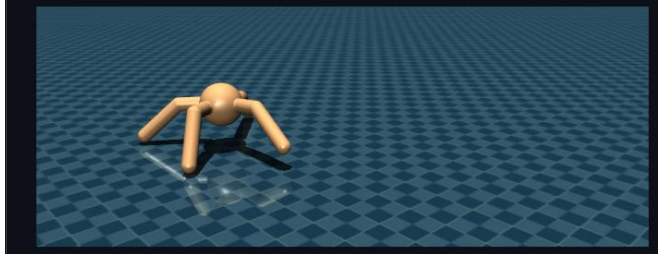
To-Do's - Challenge 1c:
nothing to do here

Challenge 1
Where to find help?

README.md in Repository

Preview | Code | Blame

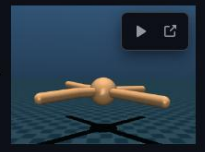
Challenge 1: Evolving Neural and Oscillatory Controllers with OpenAI Gym



In Challenge 1, you will evolutionary algorithms to evolve a neural controller to let an abstract four-legged robot learn to locomote on a flat surface as fast as possible.

Learning Goals

In this challenge, you will ...

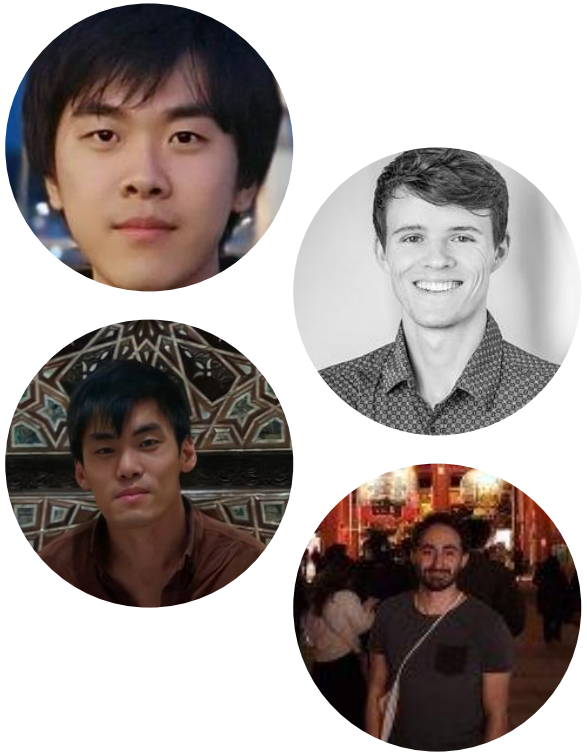


- implement a learning environment according to the widely used OpenAI Gymnasium convention Towers et al. 2024 for the high-dimensional benchmark environment `Ant`,
- interface the environment with a state-of-the-art evolutionary algorithm of your choice,
- design your own evolvable neural network controller and an oscillatory controller
- compare evolutionary optimization to deep reinforcement learning

TODO comments give hints

```
100
101     def _get_obs(self):
102         # TODO: Return observation as concatenation of:
103         # - position EXCLUDING x,y: self.data.qpos[2:].flatten() (13 values)
104         # - velocity: self.data.qvel.flatten() (14 values)
105         # This gives 27 total dimensions, making the task translation-invariant
106         # Hint: Use np.concatenate() to combine both arrays
107         raise NotImplementedError("TODO: Implement observation function")
108
109     def _get_rew(self, x_velocity: float, action):
110         # TODO: Implement reward function with three components:
111         # 1. forward_reward = x_velocity * forward_reward_weight (weight=1.0)
112         # 2. healthy_reward = healthy_reward_weight (weight=1.0)
113         # 3. ctrl_cost = ctrl_cost_weight * sum of squared actions (weight=0.5)
114         # Final reward = forward_reward + healthy_reward - ctrl_cost
115         # Return: (reward, reward_info_dict)
116         raise NotImplementedError("TODO: Implement reward function")
117
118     def _get_termination(self):
119         # TODO: Robot should terminate when:
120         # - Any value in state is not finite (check with np.isfinite(state).all())
121         # - Torso height (state[2]) is below 0.26 or above 1.0
122         # Return True if NOT healthy (i.e., should terminate)
123         # Hint: Use self.state_vector() to get current state
124         raise NotImplementedError("TODO: Implement termination function")
```

MICRO-515 TAs



Challenge 1
Submission Details

Requirements

- the code of your **final controller** and the corresponding **evolved weights** (**important**: your controller needs be compatabile with the controller interface),
- the **video** rendering of your Ant,
- the **fitness graph** showing the evolution of fitness over generations,
- a textfile **README.md** describing shortly the selected algorithm, environment design and controller (max. 300 words),
- the generated evolution_score.txt file.

Submit these documents in a single zipped folder to **Moodle** at **Submission - Challenge 1** with the following naming convention:

`2026_micro_515_TEAMNAME_LASTNAME1_LASTNAME2.zip`

One submission per team is sufficient.

Challenge 1
Leaderboard

https://lis-epfl.github.io/micro-515-EvoRob/

EPFL

MICRO-515 · EVOLUTIONARY ROBOTICS

Challenge Leaderboard

TEAMS	CHALLENGES	TOP SCORE	AVERAGE	MAX POSSIBLE
10	3	287.0	256.6	300

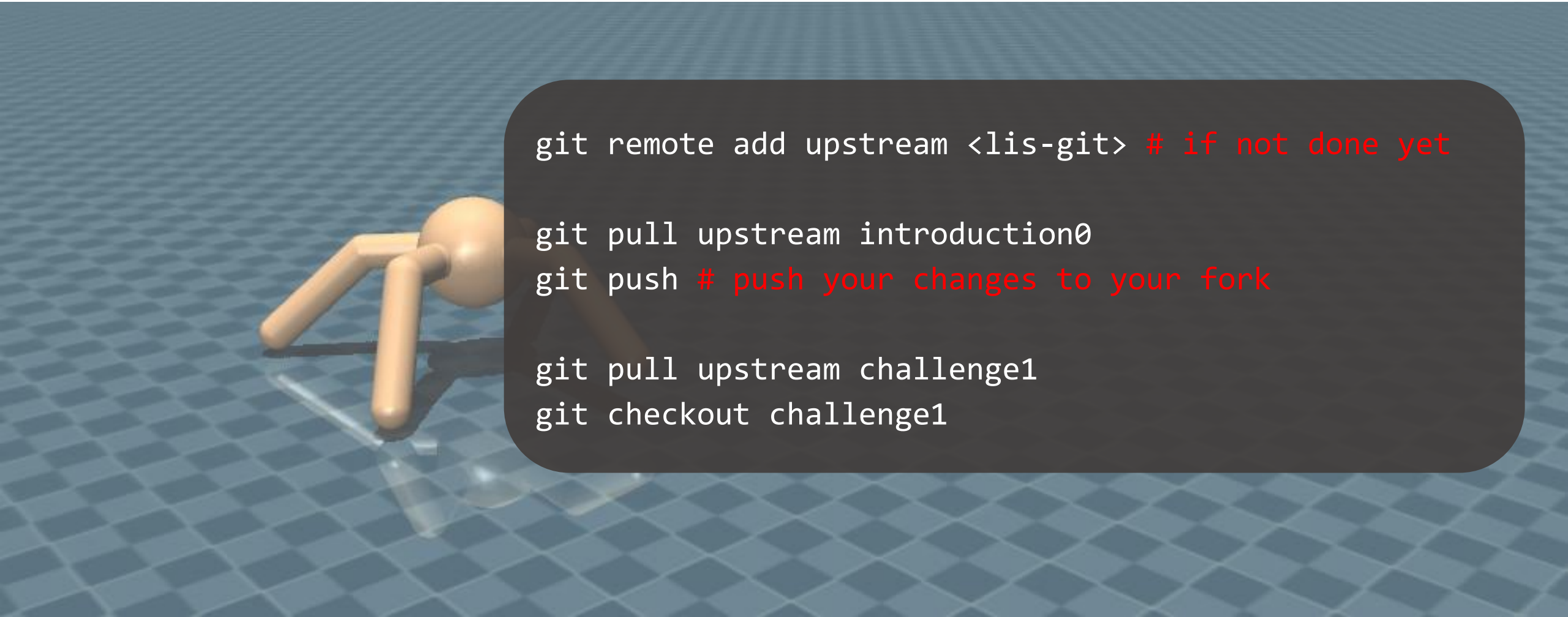
Search team...

⊞ Overall | Challenge 1 | Challenge 2 | Challenge 3

#	TEAM	CHALLENGE 1	CHALLENGE 2	CHALLENGE 3	TOTAL (NORM.)
1	NanoBot Crew	4990	330	810	287 / 300
2	Iron Wolves	4823	310	872	284.6 / 300
3	Helix Drive	3880	305	893	266.6 / 300
4	ByteForce	3950	275	901	260 / 300
5	Delta Robotics	3120	340	830	254.6 / 300
6	Sigma Dynamics	3600	290	868	253.8 / 300
7	Quantum Gears	4450	255	790	251.9 / 300

Challenge 1

Update your current repository version



```
git remote add upstream <lis-git> # if not done yet

git pull upstream introduction0
git push # push your changes to your fork

git pull upstream challenge1
git checkout challenge1
```