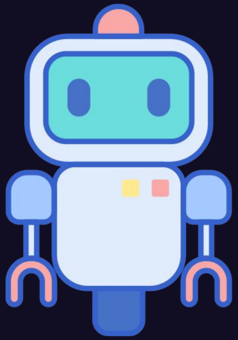
Reinforcement Learning Agents

Agents and Multi-Agent Systems



Group 2

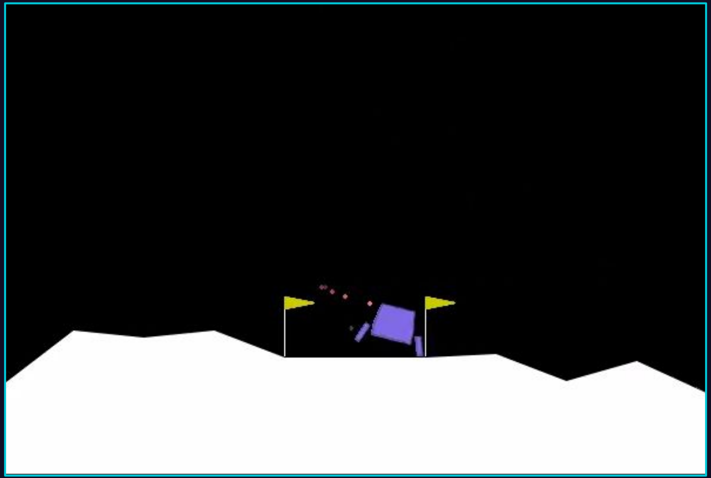
Diogo Silva (up202004288)
João Araújo (up202004293)

Problem Description

- Create a Reinforcement Learning Environment:
  - using an existing environment in Gymnasium → **Lunar Lander V2**
- Train the agents:
  - using existing algorithms in Stable baselines 3 → **PPO, A2C, DQN**
- Modify algorithms and environment's parameters:
  - fine-tune algorithms for environment:
    - hyperparametrization
    - training and validation
  - modify gravity, add wind, etc.

Environment Description - LunarLander-V2

- Rocket trajectory optimization problem
- **Agent**:
  - rocket/lander with 3 engines
  - fuel is infinite
- **Discrete** - engine either on/off (discrete actions)
- **Landing pad** - always at coordinates (0, 0)
- **Terrain** changes appearance with each episode
- Lander starts at the top in the center:
  - with random initial force applied to it
- **Episode** finishes if:
  - lander crashes (contacts moon surface)
  - lander gets outside of viewport
  - lander is not awake (not moving nor colliding)



Environment Description

Parameters	
Gravity	-10.0
Wind	Enabled
Wind Power	5.0
Turbulence	0.5

Action Space	
Do nothing	0
Fire left engine	1
Fire main engine	2
Fire right engine	3

Observation Space (8D vector)	
Coordinates	x, y
Linear velocity	vx, vy
Angle	radians
Angular velocity	radians/s
Left leg contact	boolean
Right leg contact	boolean

Rewards (per step)	
Closer to landing	↑
Slower movement	↑
More angle tilt	↓
Leg (each) ground contact	+10
Side engines firing	-0.03
Main engine firing	-0.3
Rewards (for episode)	
Landing safely	+100
Crashing	-100
Minimum points for solution	200

Algorithms

- <u>Commonly used algorithms that support a discrete environment</u>

- **PPO** (**P**roximal **P**olicy **O**ptimization):
  - optimizes the policy to maximize expected reward
  - prevents large policy updates to stabilize training
  - uses ***G**eneral **A**dvantage **E**stimation* (**GAE**) for stable training

- **A2C** (**A**dvantage **A**ctor-**C**ritic)
  - combines value-based and policy-based approaches
  - critic evaluates the actions by estimating value functions
  - uses an *Advantage Function* to update policy and value functions

- **DQN** (**D**eep **Q**-**N**etwork)
  - uses *Q-Learning* to estimate action-value function
  - approximates the *Q-Function* with deep neural networks
  - stabilizes training by breaking correlations

Training Environment

- Using the same **environment parameters** all along
- Perform **hyperparametrization**:
  - on each algorithm - until **100k** steps (**16** different combinations)
  - relevant variables tuned:
    - *learning_rate* - current progress remaining
    - *gamma* - discount factor
    - *gae_lambda* - factor for trade-off of bias vs variance (PPO, A2C)
    - *ent_coef* - entropy coefficient for loss calculation (PPO, A2C)
    - *tau* - update rate of network (DQN)
    - etc.

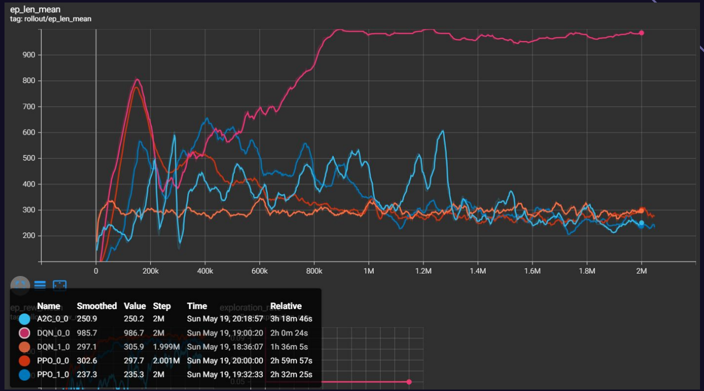
Models Used

- Take parameters of the best pair of results:
  - PPO models:
    - one with default parameters
    - one with higher *gae_lambda* and *ent_coef*
  - A2C models:
    - single one with default parameters
  - DQN models:
    - one with default parameters (*tau* → hard update)
    - one with higher *learning_rate* and opposite *tau* (soft update)
- Train models until **2M** steps
- Visual **validation** every 250k steps - to see agent's behaviours at the moment
- Final rewards above **200** → the environment has been solved!

▶ Training Evolution

Episode Statistics

- **Mean Episode Length:**
  - DQN with default parameters - odd increase (high default *tau* value → unstable training)
  - custom PPO/DQN - rapid reduction/leveling
- **Mean Episode Reward:**
  - PPO algorithms - quicker to find episode solution and stabilizing
  - A2C - solution found but some fluctuations
  - DQN algorithms - no signs of approaching the solution



Mean Episode Length

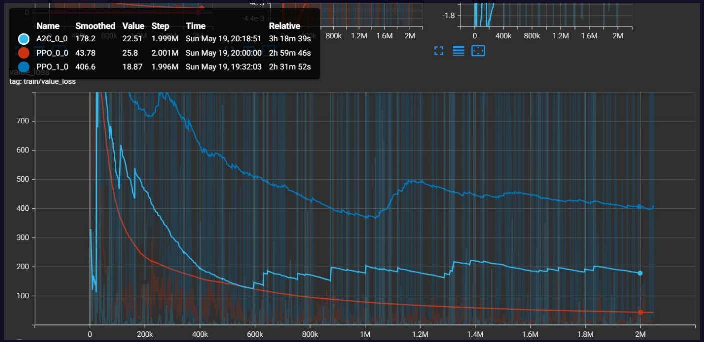
Mean Episode Reward

▶ Training Evolution

Other Statistics

- **Entropy Loss:**
  - randomness in policy's actions:
    - being reduced throughout training
    - less exploration, more exploitation
  - PPO allowed for more exploration at early stages and slowly converted
- **Value Loss:**
  - predicted rewards vs actual observed returns:
    - improves (decrease) during training
  - PPO and A2C with default parameters better at predicting than modified PPO

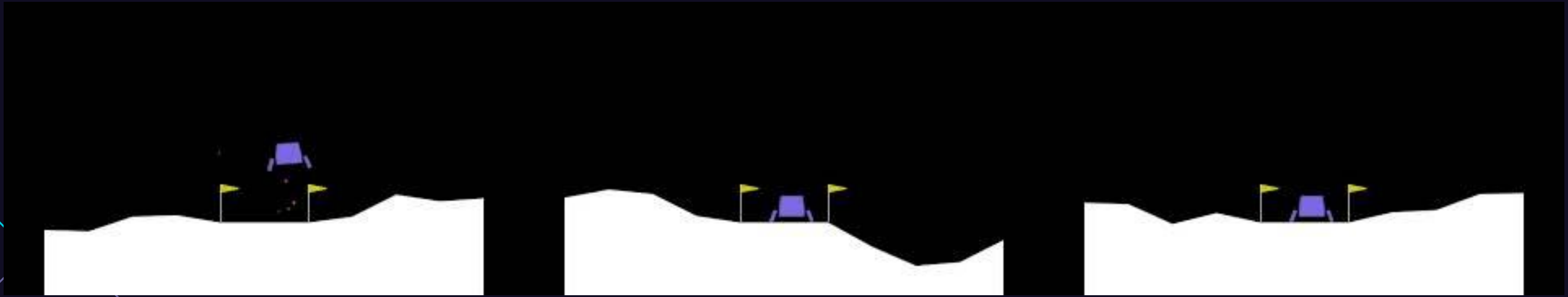
Entropy Loss



Value Loss

Validation - PPO

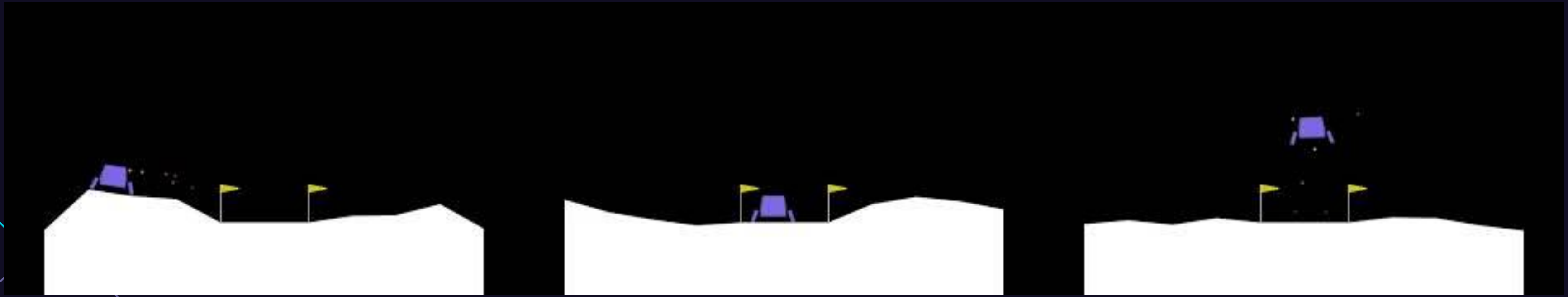
- **500k steps**: quickly presents a good degree of success, despite some imperfections
- **1M steps**: fulfills the episodes effectively, showing it can adapt rapidly
- **2M steps**: continues to complete each episode in an efficient manner
- Good balance in policy update steps:
    - **stabilizes training** → steady and reliable improvements (as shown in the graphs)



500k steps | 1M steps | 2M steps

Validation - A2C

- **500k steps**: incapable of solving, wanders off a bit in the air and sideways
- **1M steps**: achieves success, but not in the most efficient way (excessive tilting and engine usage)
- **2M steps**: more efficient, despite some slight hesitations near landing
- **Synchronous update** of policy and value function:
    - less stable training → less thorough exploration of the state space



500k steps 1M steps 2M steps

Validation - DQN

- **500k steps**: never gets to land (constantly in flight), quite slow, often moves sideways
- **1M steps**: remains with long times, but shows an effort to approach the landing
- **2M steps**: intentionally moves towards the landing, but hovers there until timeout
- Used policy (ε-greedy) leading to **suboptimal decisions** (in this case):
  - slower movement, less angle tilt, closer to landing, avoid crashing → loop
  - difficulty in assessing long-term benefits

500k steps | 1M steps | 2M steps

Unforeseen Scenarios

- Testing out the best results from each algorithm on **LunarLander-V2** but...
  - **rougher environment:**
    - Wind Speed = 15.0 → **3x** the initial wind speed
    - Turbulence Power = 1.5 → **3x** the initial turbulence power
    - Gravity = -5.0 → **0.5x** the initial gravity
  - expectations based on training evaluation/validation:
    - PPO models should perform the best - easily adapts on similar conditions
    - DQN models should have a harsher time - suboptimal exploration
- ***Note** on an easier environment*:
  - **PPO** & **A2C** succeeded the environment with ease
  - DQN still demonstrated some level of struggle → not finding the correct course of action

Unforeseen Scenarios

- **PPO** - able to adapt to harsh conditions, rapidly contacts the ground and adjust from there
- **A2C** - manages to use engines well to fight the wind, takes a bit longer to contact the ground
- **DQN** - constantly in flight, fighting the turbulence, contacts landing pad but decides not to rest
- **Comparisons:**
  - PPO performed best, followed by A2C - both were able to succeed at several attempts
  - DQN was the worst by a large margin - virtually never landing correctly

PPO **A2C** **DQN**

Conclusion

- **Policy based**, **actor-critic** and **value based** methods:
  - demonstration of distinct characteristics/approaches within reinforcement learning
  - PPO with its penalty to the objective function ended up bringing the best results
- Significant importance of balancing exploration and exploitation:
  - learning process becomes more reliable
  - **excessive exploration** - slow learning and inconsistent performance
  - **excessive exploitation** - lack of robustness:
    - agent policy highly specialized to specific states/actions - hard to adapt in unseen situations