



Georgian  
College

Start



# LUNAR LANDER

Subject : REINFORCEMENT LEARNING

Submit by : NILKANTH AMIN



# INTRODUCTION

In the realm of machine learning, reinforcement learning stands out as a captivating paradigm that empowers agents to learn intricate decision-making strategies by interacting with dynamic environments. This project embarks on a journey into the realm of reinforcement learning through the lens of a captivating challenge: training an agent to navigate the complexities of space travel and safely land a spacecraft on the Moon's surface. The heart of this endeavor lies within the "LunarLander-v2" environment, a virtual simulation that mirrors the intricate task of guiding a spacecraft while judiciously managing fuel and avoiding catastrophic crashes. Through a step-by-step exploration of code, methodologies, and techniques, this project offers an immersive opportunity to unravel the foundations of reinforcement learning, comprehend the nuances of agent training using the Proximal Policy Optimization (PPO) algorithm, and witness the agent's evolution from novice to adept lunar navigator. Join us as we dive into this captivating realm, where code meets creativity, and machine intelligence conquers extraterrestrial challenges.

---



# SETUP AND DEPENDENCIES

## Package Installation:

- The first step involves installing essential packages and dependencies. The following commands are executed to ensure the availability of required tools:

```
!apt install swig cmake
!pip install -r https://raw.githubusercontent.com/Nilkanth14/Lunar-Lander-f
!sudo apt-get update
!sudo apt-get install -y python3-opengl
!apt install ffmpeg
!apt install xvfb
!pip3 install pyvirtualdisplay
```



# SETUP AND DEPENDENCIES

## Virtual Display Setup:

- A crucial component for headless operation and rendering is a virtual display. This is achieved using the pyvirtualdisplay library. The code snippet below illustrates how the virtual display is initiated:

```
from pyvirtualdisplay import Display

virtual_display = Display(visible=0, size=(1400, 900))
virtual_display.start()
```



# SETUP AND DEPENDENCIES

## Importing Libraries:

- Various libraries play a pivotal role in this project. Notable imports include those from the `huggingface_sb3`, `stable_baselines3`, and `gymnasium` packages. These libraries equip the project with the tools required for RL agent training and evaluation.

## Creating the Lunar Lander Environment:

- The project centers around the "LunarLander-v2" environment, created using the `gymnasium` library's `make` function. This environment simulates the spacecraft landing task, forming the foundation for agent training.

## Exploring the Environment:

- A critical step in understanding the environment's dynamics is taking random actions within it. The code demonstrates this exploration process, wherein actions are sampled from the action space, and the environment's responses are observed.

## Environment Characteristics:

- To comprehend the nature of the Lunar Lander environment, its observation and action spaces are explored. The dimensions of these spaces and sample observations/actions are presented.

## Vectorized Environment Creation:

- In preparation for training acceleration, a vectorized environment is created using the `make_vec_env` function. This environment is employed for agent training to enhance efficiency.
-



# EXPLORING THE LUNAR LANDER ENVIRONMENT

- **Foundations of Exploration:**
  - Exploration forms the bedrock of an RL agent's learning journey. It entails allowing the agent to interact with the environment by taking actions, thereby gaining firsthand experience of the outcomes.
- **Random Action Execution:**
  - During the exploration phase, the agent executes actions randomly sampled from the action space. These actions are not guided by any learned policy but rather serve as the agent's initial foray into the environment.
- **Observing State and Outcome:**
  - With each action, the environment responds by transitioning to a new state. This state encapsulates relevant information about the agent's position, velocity, orientation, and more within the lunar descent context.
  - The environment provides feedback in the form of a reward, which quantifies the agent's performance based on its actions. Positive rewards are associated with favorable outcomes, such as controlled descent, while negative rewards signify undesired behaviors like crashes.
- **Termination Conditions:**
  - The exploration process continues for a predefined number of steps. The environment signals the termination of exploration through two primary conditions:
    - Termination: The spacecraft has either safely landed or crashed, leading to the end of the current episode.
    - Truncation: If exploration reaches a time limit, the episode is truncated. This typically occurs when the agent takes too long to perform an action.
- **Resetting the Environment:**
  - Once an episode concludes, either through termination or truncation, the environment is reset to its initial state. This ensures that the agent has a fresh start for its subsequent exploratory endeavors.
- **Training Agent's Initial Observations:**
  - Through this exploration process, the agent garners initial insights into how its actions influence the environment. These observations lay the foundation for learning and formulating effective strategies.



# PPO ALGORITHM CONFIGURATION

- **Introducing Proximal Policy Optimization (PPO):**
  - Proximal Policy Optimization is a widely-used reinforcement learning algorithm known for its stability and effectiveness in training agents.
  - PPO optimizes policy networks, which determine the agent's actions in response to observations, by iteratively improving policy updates.
- **Algorithm Configuration Components:**
  - **Policy:** The choice of policy architecture shapes the agent's decision-making strategy. In this context, we employ the Multi-Layer Perceptron (MLP) policy.
  - **Environment:** The environment, often referred to as the task or challenge, is the "LunarLander-v2" environment in our case.
  - **Batch Size:** Determines the number of experiences sampled from the environment for each update. Larger batch sizes might provide more stable updates.
  - **Gamma (Discount Factor):** Captures the agent's preference for immediate rewards over delayed rewards. A higher gamma emphasizes long-term rewards.
  - **Lambda (GAE Lambda):** The Generalized Advantage Estimation lambda balances the emphasis between bias and variance when estimating advantages.
  - **Entropy Coefficient:** Influences the agent's exploration strategy by introducing randomness. A higher entropy coefficient encourages exploration.
  - **Verbose:** Setting verbosity to 1 provides progress updates during training for insights into the agent's learning trajectory.
- **Hyperparameter Tuning:**
  - Configuring hyperparameters, such as batch size, gamma, and lambda, is a nuanced process that impacts the algorithm's performance.
  - Optimal hyperparameter values can vary based on the environment and problem complexity. Iterative tuning is often essential for achieving desired outcomes.
- **Stability and Convergence:**
  - PPO is known for its stability and the balance it strikes between exploration and exploitation. The algorithm avoids drastic policy updates to prevent policy divergence.





# EVALUATING THE TRAINED AGENT

- **Evaluation Environment Setup:**

- To gauge the agent's effectiveness, an evaluation environment is set up. This environment mirrors the Lunar Lander challenge, providing a controlled environment to assess the agent's learned policy.

- **Mean and Standard Deviation of Rewards:**

- The performance of the trained agent is quantified through rewards—a numerical measure of its success in accomplishing the task.
- The `evaluate_policy` function measures the agent's performance over a set number of evaluation episodes within the evaluation environment.
- The result consists of the mean and standard deviation of the rewards earned by the agent during these episodes.

- **Interpreting Mean Reward:**

- The mean reward reflects the agent's average success in accomplishing the task over the evaluation episodes.
- A higher mean reward signifies that the agent is making successful decisions that yield favorable outcomes in navigating the lunar descent.

- **Understanding Standard Deviation:**

- The standard deviation captures the variability in the agent's performance across evaluation episodes.
- A lower standard deviation suggests consistent and reliable decision-making, while a higher standard deviation implies more variability in outcomes.

- **Deterministic Evaluation:**

- In some cases, the agent's actions are deterministic—consistent across episodes. This eliminates randomness in decision-making, yielding more predictable outcomes.

- **Performance Insights:**

- The evaluation metrics provide insights into the agent's adaptability, robustness, and overall performance. A well-trained agent should exhibit high mean rewards and low standard deviations.

- **Refinement and Iteration:**

- The evaluation phase often guides further iterations in training and hyperparameter tuning. If the agent's performance is suboptimal, adjustments can be made to enhance its learning process.





# SHARING THE MODEL ON HUGGING FACE

- With the agent's training and evaluation complete, the time has come to share its acquired knowledge with the wider community. This slide walks us through the process of packaging and sharing the trained Proximal Policy Optimization (PPO) model using Hugging Face's model hub. By understanding the steps to define model attributes, commit to a repository, and make the agent's prowess accessible, we foster collaboration and innovation.

## **The Power of Hugging Face's Model Hub:**

- Hugging Face's model hub serves as a central platform for sharing and discovering machine learning models. It enables seamless collaboration, knowledge dissemination, and experimentation.

## **Defining Model Attributes:**

- Before sharing the model, key attributes are defined. These include the model's name, architecture, environment ID ("LunarLander-v2"), and other relevant details that encapsulate the agent's learning journey.

## **Creating a Repository:**

- To make the model available on the hub, a repository is created. This repository houses the model's code, configurations, and artifacts that collectively represent the trained agent's capabilities.

## **Committing to the Repository:**

- The model, along with its associated code and artifacts, is committed to the repository. This marks the moment when the agent's wisdom becomes accessible to the global community of machine learning enthusiasts.



Nilkanth014 / **ppo-LunarLander-v2**


















 like
0

-  Reinforcement Learning
-  Stable-Baselines3
-  LunarLander-v2
-  deep-reinforcement-learning
-  Eval Results

 Model card
 **Files and versions**
 Community
 Settings


 Use in stable-baselines3

 main
ppo-LunarLander-v2
 1 contributor
History: 6 commits
+ Add file

 Nilkanth014 Upload PPO LunarLander-v2 trained agent 2f21b4e about 3 hours ago
 ppo-LunarLander-v2 Upload PPO LunarLander-v2 trained agent about 3 hours ago
 .gitattributes 1.52 kB  initial commit about 5 hours ago
 README.md 785 Bytes  Upload PPO LunarLander-v2 trained agent about 3 hours ago
 config.json 13.6 kB  Upload PPO LunarLander-v2 trained agent about 3 hours ago
 ppo-LunarLander-v2.zip  pickle 146 kB   Upload PPO LunarLander-v2 trained agent about 3 hours ago
 replay.mp4 182 kB  Upload PPO LunarLander-v2 trained agent about 3 hours ago
 results.json 163 Bytes  Upload PPO LunarLander-v2 trained agent about 3 hours ago



**Amin**

Nilkanth014

Profile

Account

Organizations

Billing

**Access Tokens**

SSH and GPG Keys

## Access Tokens

### User Access Tokens

Access tokens programmatically authenticate your identity to the Hugging Face Hub, allowing applications to perform specific actions specified by the scope of permissions (read, write, or admin) granted. Visit [the documentation](#) to discover how to use them.

RL Lunar lander **WRITE**

Manage ▾

hf\_jVcNbMDTjQJhbrTOrvzYGcHkdxlhRzTPvE

[Hide](#) 

New token