INTELLIGENT AUTOPILOT SYSTEM FOR AIRCRAFT MANEUVERS IN X-PLANE 11

Rizwan, Oneeb, Krishna, Priya, Martin

Project Goal

Proposal:

- Implement RL Agent capable of following a given flight path
- Compare the performance of different RL models

Deliverables for the finals:

- Have (atleast) 3 RL models ready for navigating the aircraft through a predetermined flight path with clear visualization of suggested and actual flight paths.

Progress So Far...

Implementation of following algorithms in continuous action space:

1. REINFORCE

1. Deep Deterministic Policy Gradient

1. Proximal Policy Optimization

Current Problem Definition

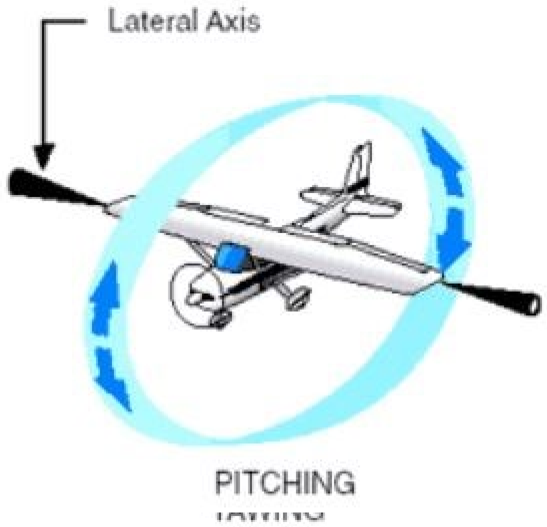
The aircraft will be spawned at 3600m and the target is to descend to 2500m irrespective of the final attitude.

We chose the 8 most relevant observation space parameters and these include:

1. Indicated Airspeed
2. Vertical Velocity
3. Altitude
4. Pitch
5. Roll
6. True Heading
7. Angle of Attack
8. Sideslip Angle

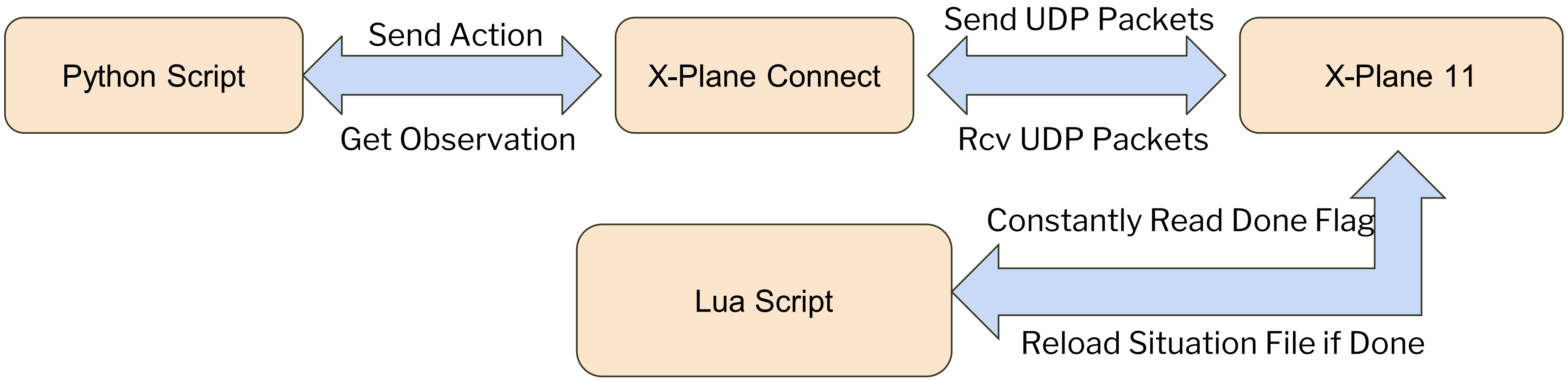
The action space was limited to 4 actions which include:

1. Latitudinal Stick (to control the elevator / pitching motion)
2. Longitudinal Stick (to control the ailerons / rolling motion)
3. Rudder Pedals (to control rudder / yawing motion)
4. Throttle



Environment Setup

- X-Plane 11's API allows data access through UDP sockets.
- NASA XplaneConnect plugin allows data to be read from/ written to the simulator without having to program sockets yourself.
- It provides several functions to interact with the simulator like sendCTRL(), getCTRL(), getDREFs() etc.
- There is 1 core functionality which is missing from the plugin is the ability to reload the configured flight (situation file).
- To build this functionality we used another plugin called FlyWithLua and wrote the script in Lua to reload the situation file whenever the aircraft crashes or an episode finishes.



GYM XPlane

- Open AI Gym does not have any environment for X-Plane 11.
- We implemented our own functions following Gym guidelines.
- These include step, compute_reward, check_terminal_state etc.
- This essentially provides a level of abstraction between our agent and the simulator and make it simple to code the RL agents.

Reward Function

+6000 for every step in successful range

–√(|current_altitude – target_altitude|) for each step outside target zone

–100,000 for crashing

–20,000 for finishing outside the successful range

REINFORCE Algorithm

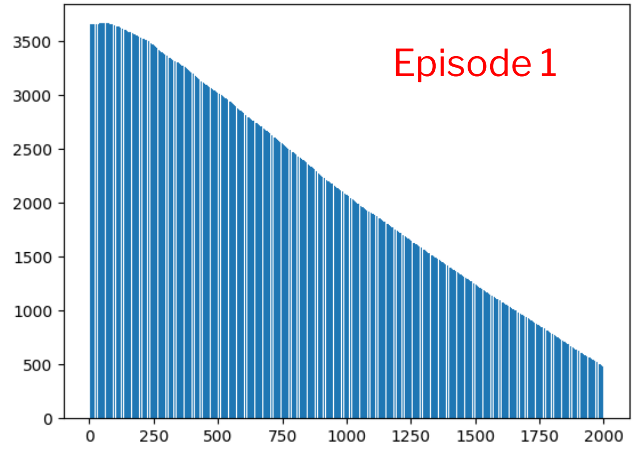
- A policy gradient approach to solve problems dealing with continuous action spaces.
- Network is modelled to output the parameters (mu, sigma)for 4 normal distributions corresponding to four actions in the action space.
- Actions are sampled from these distributions and the corresponding rewards from the next states are then used to compute loss and perform backpropagation.

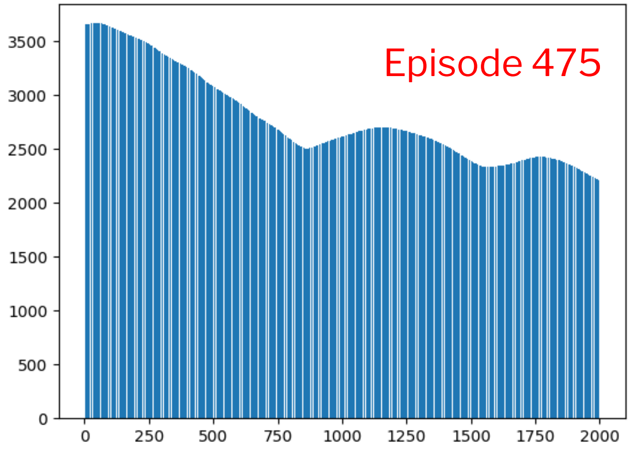
REINFORCE Algorithm

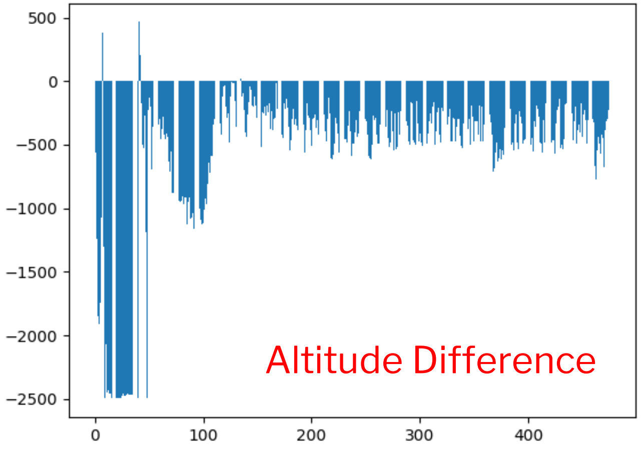
- Performed training by varying hyperparameters and activation functions.
- The best performance we got was using the following parameters:
  - NN Architecture:
    - Input_layer : 8 → No Activation Function
    - H1_layer : 256 → ReLU
    - H2_layer: 256 → ReLU
    - Output Layer: 4 (mu) No Activation Function
      : 4 (sigma) → ELU + 0.001
  - α = 0.001, γ: 0.9

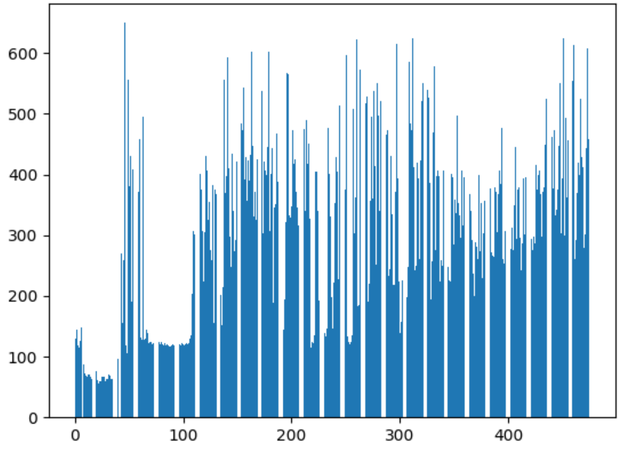
  - Results (500 Episodes): Promising! After crashing in the first couple of episodes, the aircraft started “loitering” close to the target altitude.

REINFORCE Results





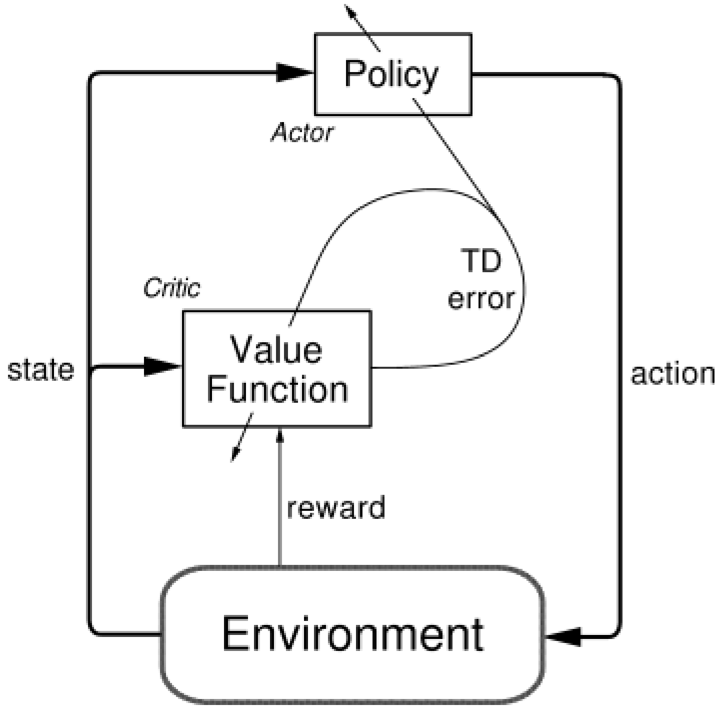




Successful Steps

Proximal Policy Optimization (PPO)

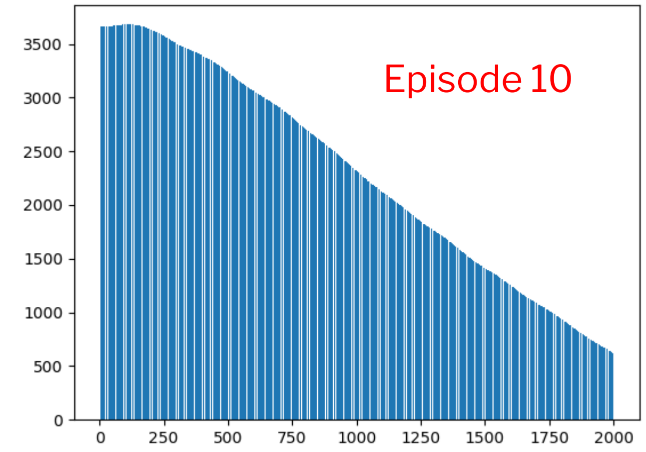
- Actor-Critic method based algorithm to solve problems dealing with continuous action spaces.
- 2 networks are trained in parallel.
- Actor network outputs the normal distribution parameters for sampling actions. Critic network approximates the value function for the current state.
- Rewards and value function are used to compute the advantage used in calculations for loss and backpropagation

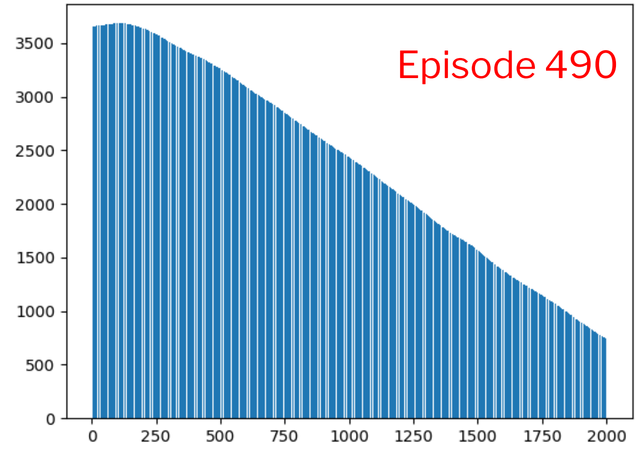


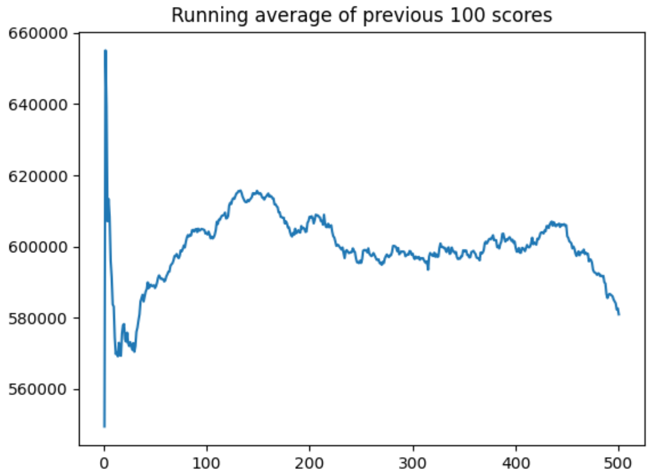
Proximal Policy Optimization (PPO)

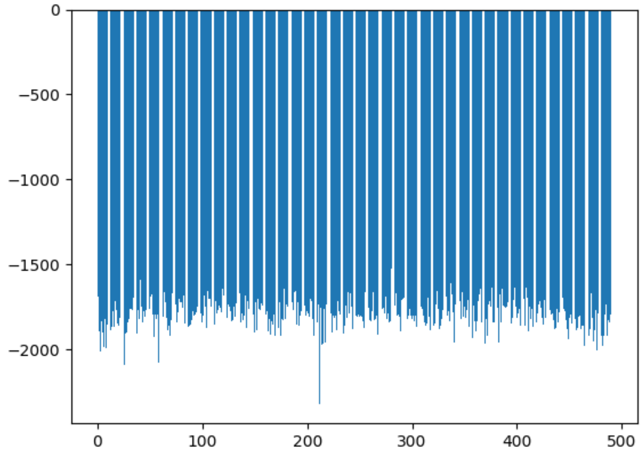
- We used the same network from REINFORCE as our actor network and for value network as well, except that at the output layer it has just 1 output for value function.
- During the first training session of 500 episodes, the results were not satisfactory.
- The average scores plateaued.
- The reason which we realised was that the buffer size that we were using was very small.

PPO Results









Successful
Steps

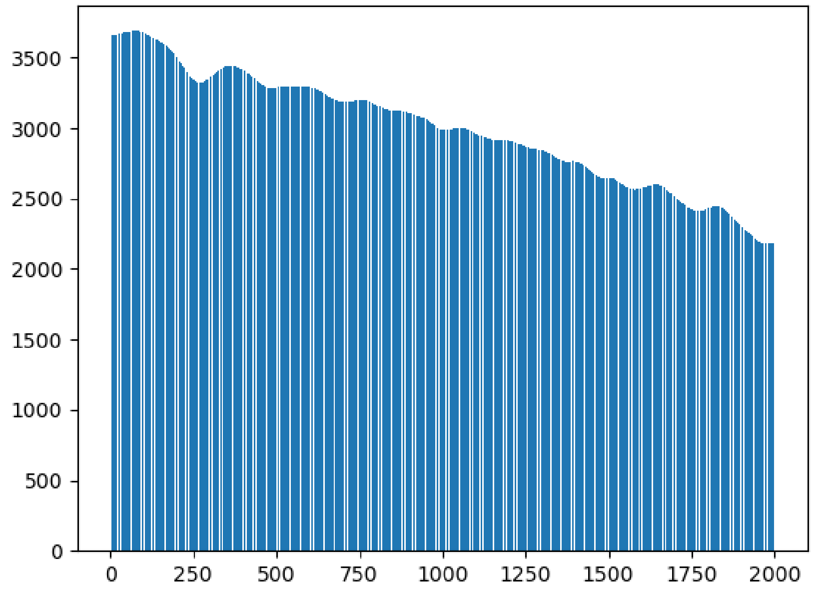
Deep Deterministic Policy Gradient

Network Structure:

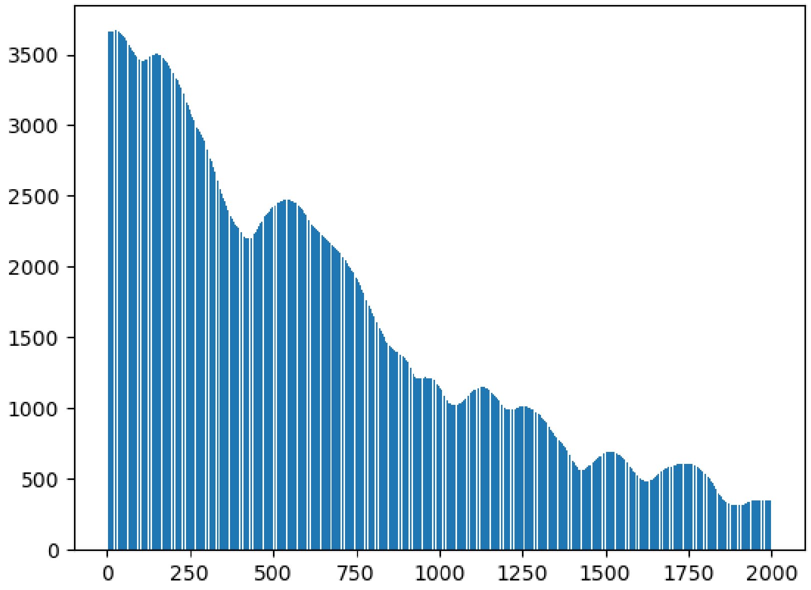
- Batch Size: 64
- Target Update Factor: 0.001
- 2 layers Actor:
  - learning rate: 0.0001
- 2 layers Critic:
  - learning rate: 0.001
- Memory buffer size: 1000,000

Deep Deterministic Policy Gradient

- Altitude after 200 episodes:



Some interesting episodes:




Episode 465

Future Tasks

Agents:

- Train PPO with large enough replay buffer for enough epochs so that the model can capture the patterns.
- Try out more complicated networks so that the model can pick on the complex underlying patterns.
- Make necessary changes to DDPG

Environment and Objective:

- Switch from a fixed target altitude approach to any given target altitude. This will enable the model to learn the generic flight level change behaviour instead of just descending to 2500m.
- Reducing the number of steps in each episodes and keeping the target altitude closer to the spawn altitude. This would help us run more episodes in the same time.

Individual Contributions

Muhammad Rizwan Malik:

Environment/Gym Setup, REINFORCE Debugging, PPO Implementation, Mid Term Presentation

Muhammad Oneeb Ul Haq Khan:

REINFORCE Implementation/Debugging, EDD, Technical Paper, Mid Term Presentation

Martin Huang:

DDPG Implementation, EDD, Mid Term Presentation

Krishnateja Gunda:

REINFORCE Implementation, EDD, Technical Paper

Rengapriya Aravindan:

DDPG, EDD, Technical Paper, Project Website methods

THANK YOU!