RL AUTOPILOT

USING X-PLANE 11

Project Team
Rizwan | Oneeb | Krishnateja
Martin | Rengapriya

Tuesday 30 November, 2021

Project Timeline

Week 10

- PPO Training
- DDPG Implementation
- Reward fine-tuning

Week 11

- DDPG Tuning & Training
- PPO Training & Testing
- Research other agents to explore
- Finish Project website

Week 12

- DDPG Training Continued
- Implementation of new agent
- Improve visualization

Week 13

- Training of new agent
- Comparative Analysis
- Identify critical weak points

Week 14

- Conclude any code-facing work
- Prepare Final Presentation
- Finalize demos

Week 15

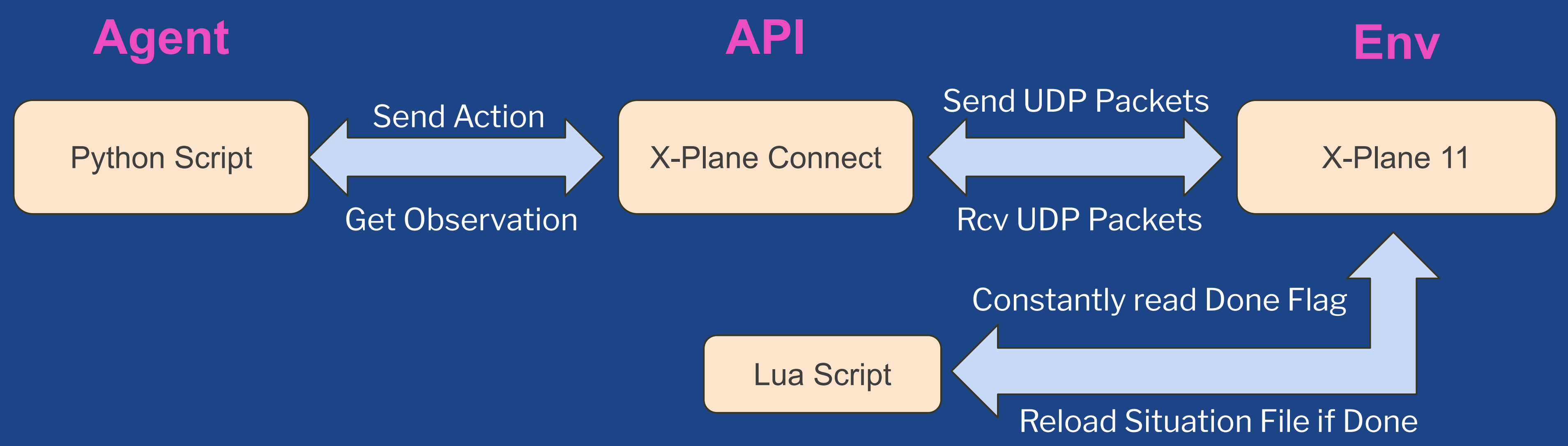
- Final Presentation
- Finish Project Documentation
  - EDD
  - Technical Paper

Environment

1. X Plane Simulation
2. RL Environment
3. Reward Function

Simulation Setup

- X-Plane 11's API allows data access through UDP sockets.
- NASA XplaneConnect plugin was used to facilitate communication
- Added "situation reset" functionality using a Lua script



Environment

1. X Plane Simulation
2. **RL Environment**
3. Reward Function

RL Environment

- The aircraft is spawned at an initial altitude and the goal is to descend to a given target altitude 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

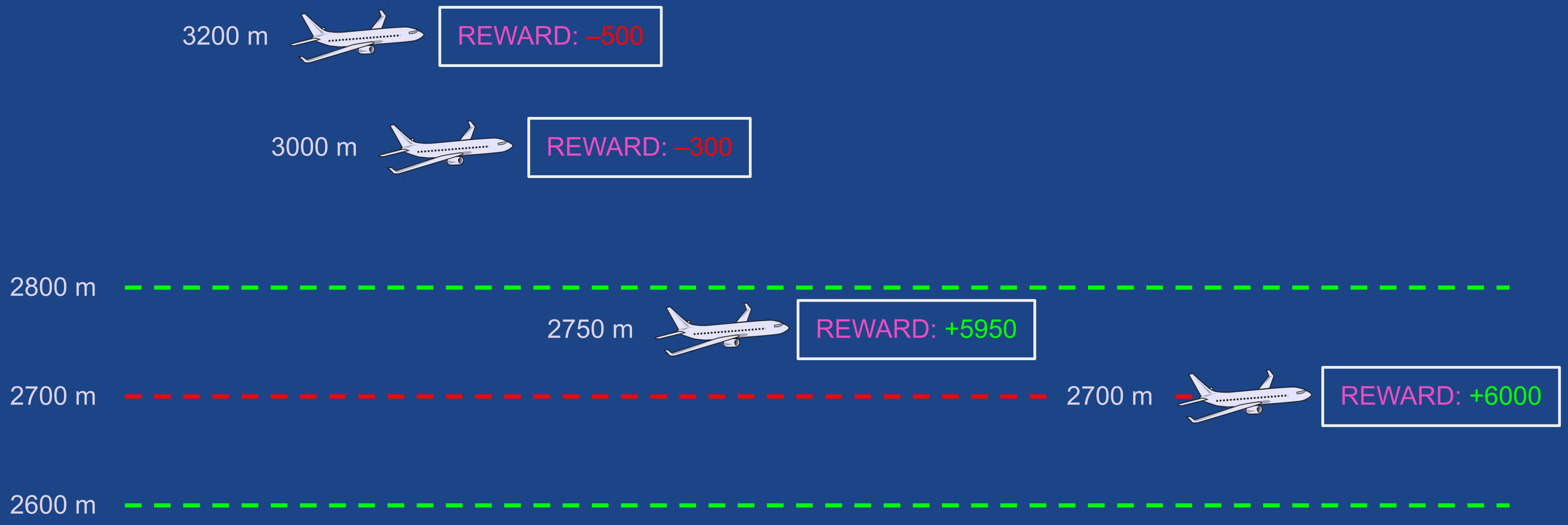
1. X Plane Simulation
2. RL Environment
3. Reward Function

Reward Function

+6000 for every step in successful range

–(|current_altitude – target_altitude|) for each step

–100,000 for crashing



RL Agents

RL Agents

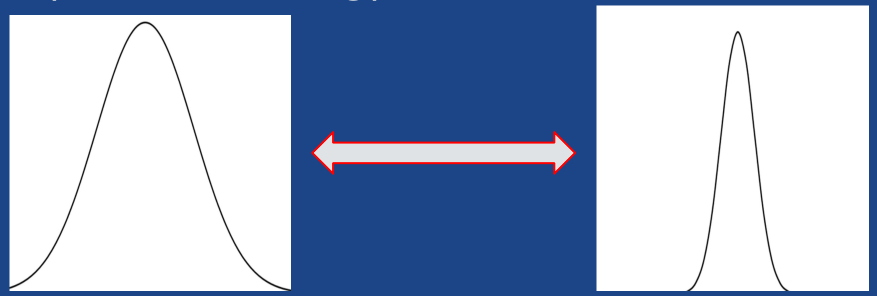
1. REINFORCE
2. Proximal Policy Optimization (PPO)
    - Continuous
    - Categorical
3. Deep Deterministic Policy Gradient
    - Original
    - Quick Ending
        - Target Start
4. Soft Actor-Critic

REINFORCE

- Simplistic method, which allowed us to familiarize ourselves with the world of Policy Gradients.
- REINFORCE is extremely computationally inefficient. We were almost guaranteed no results.
- At the Midterm we had decided not to continue training our model.

Method

The NN takes observation space as input and output (μ, σ) for the normal probability distribution for each continuous space action from which actions will be sampled (while training)

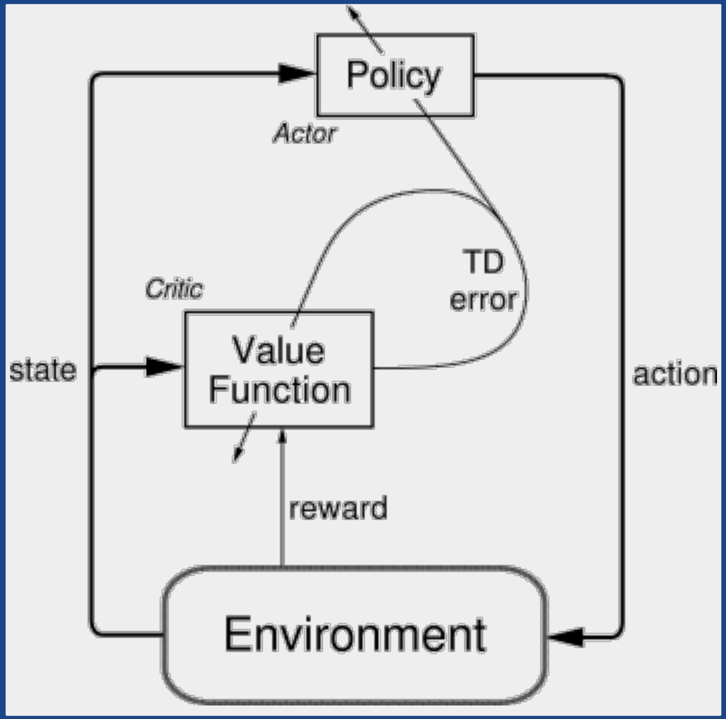


RL Agents

1. REINFORCE
2. Proximal Policy Optimization (PPO)
   - Continuous
   - Categorical
3. Deep Deterministic Policy Gradient
   - Original
   - Quick Ending
     - Target Start
4. Soft Actor-Critic

Proximal Policy Optimization (PPO)

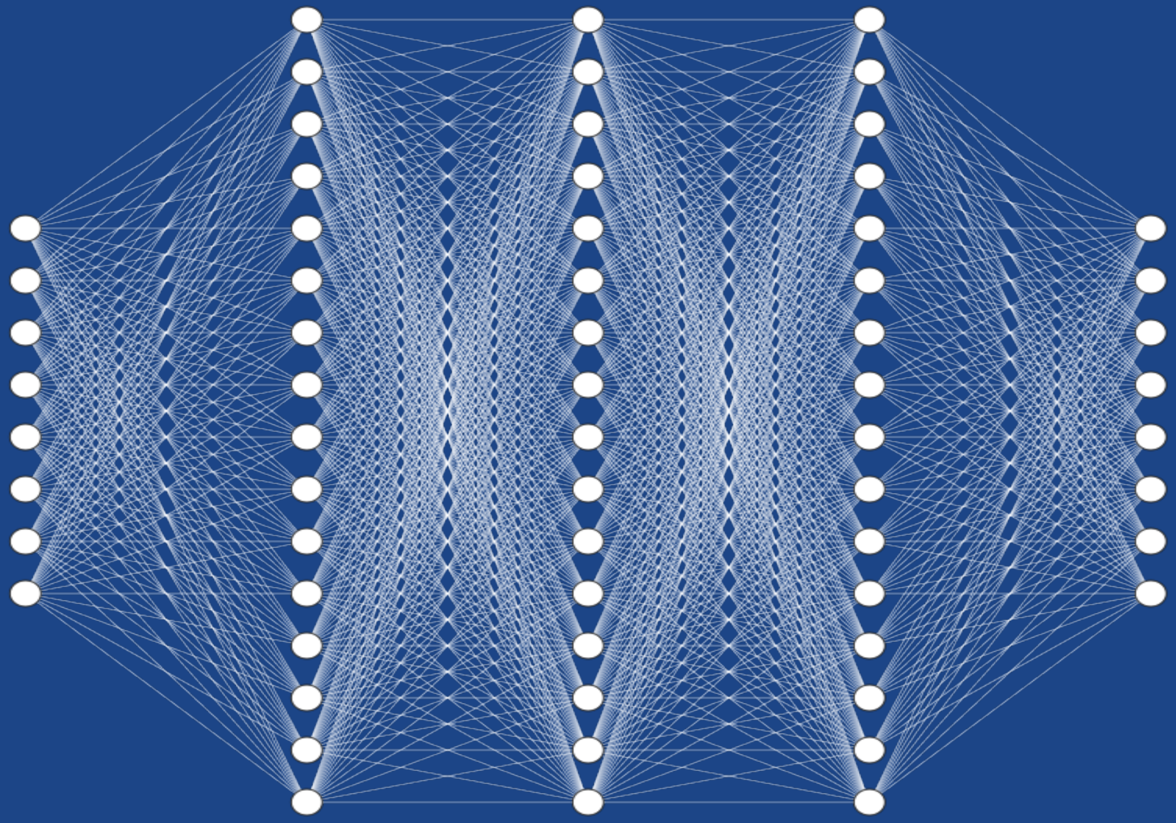
- Actor-Critic method based algorithm to solve problems dealing with continuous action spaces.
- 2 networks are trained in parallel.
- Actor network outputs the normal distribution parameters for sampling actions. Critic network approximates the value function for the current state.
- Rewards and value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



PPO Continuous

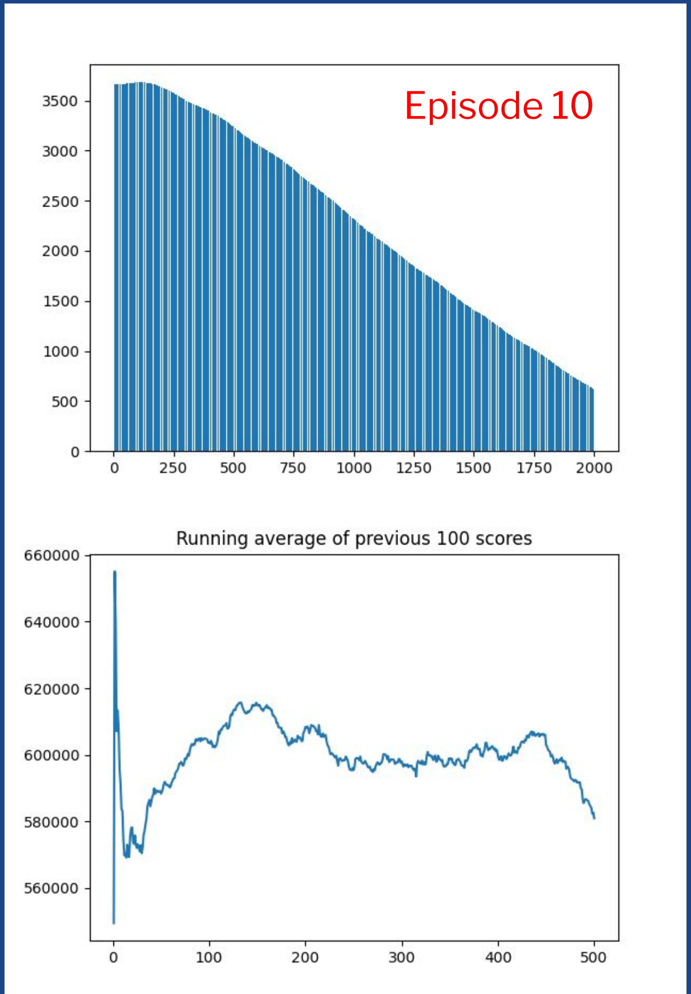
- We used the same network from REINFORCE as our actor network and for value network as well.
- The output layer has 8 units for predicting Mu and Sigma for 4 actions.

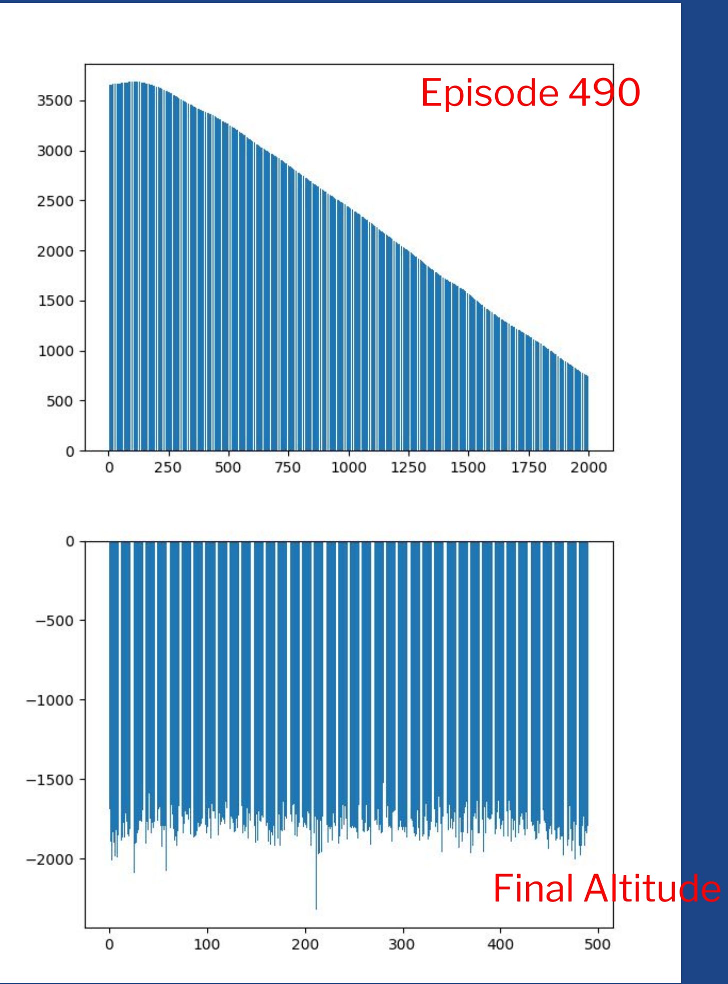
Architecture: 8 x 256 x 256 x 256 x 8
Activation: ReLU
Output Activation: Tanh for Mu
: Sigmoid for Sigma
LR: 3 x 10^-3



PPO Continuous

- 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.





RL Agents

1. REINFORCE
2. Proximal Policy Optimization (PPO)
   - Continuous
   - Categorical
3. Deep Deterministic Policy Gradient
   - Original
   - Quick Ending
     - Target Start
4. Soft Actor-Critic

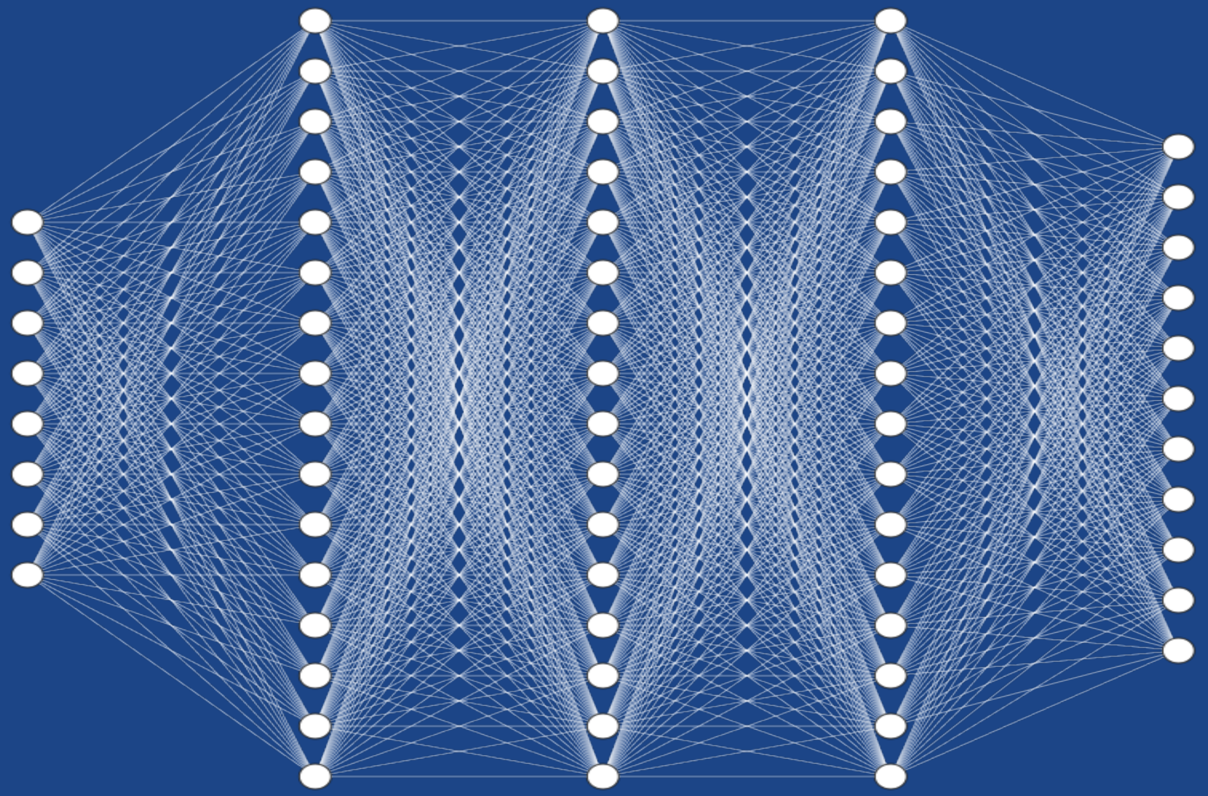
PPO Categorical

- To make the underlying model simpler, the action space was discretized into 11 equal graduations.
- The redefined action spaces:
  - Latitudinal Stick, Longitudinal Stick, Rudder Pedals: [-0.5, 0.5]
  - Throttle: [0.5, 1.0]
- 4 separate agents for 4 actions.
- Replay memory buffer was increased to store 100000 steps
- Mini batches of size 5000 were used.

PPO Categorical

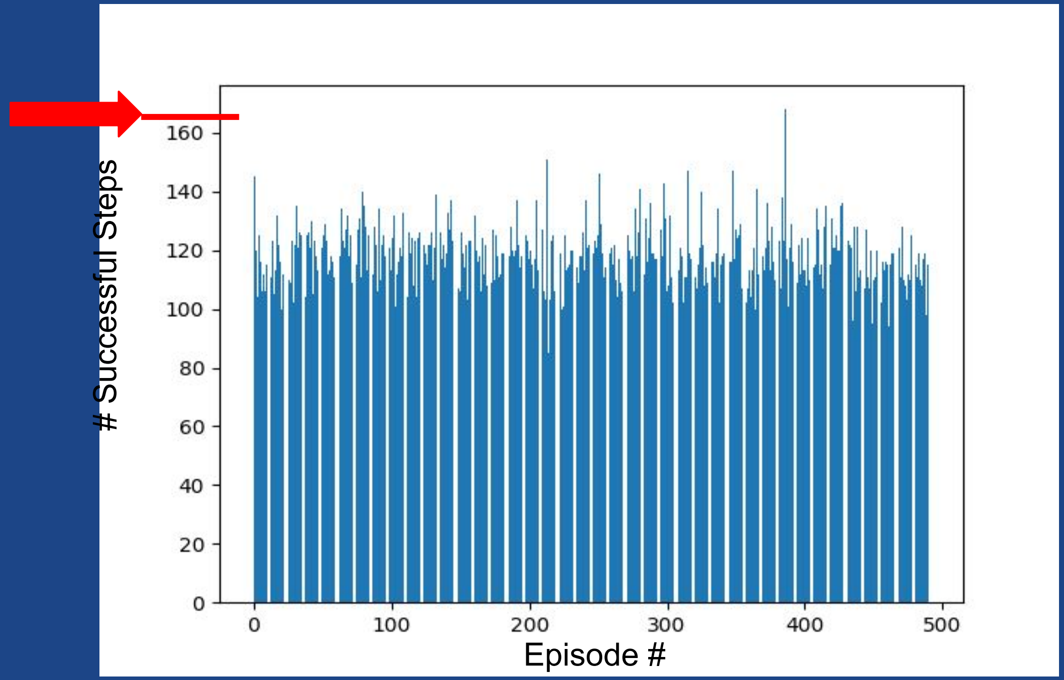
- Same networks were used for Actor and Critic except for the output layer.
- Actor outputs a pdf over 11 discretized action values.
- Critic outputs a single value function for the state

Architecture: 8 x 256 x 256 x 256 x 11
Activation: ReLU
Output Activation: Softmax
LR: 3 x 10^-3

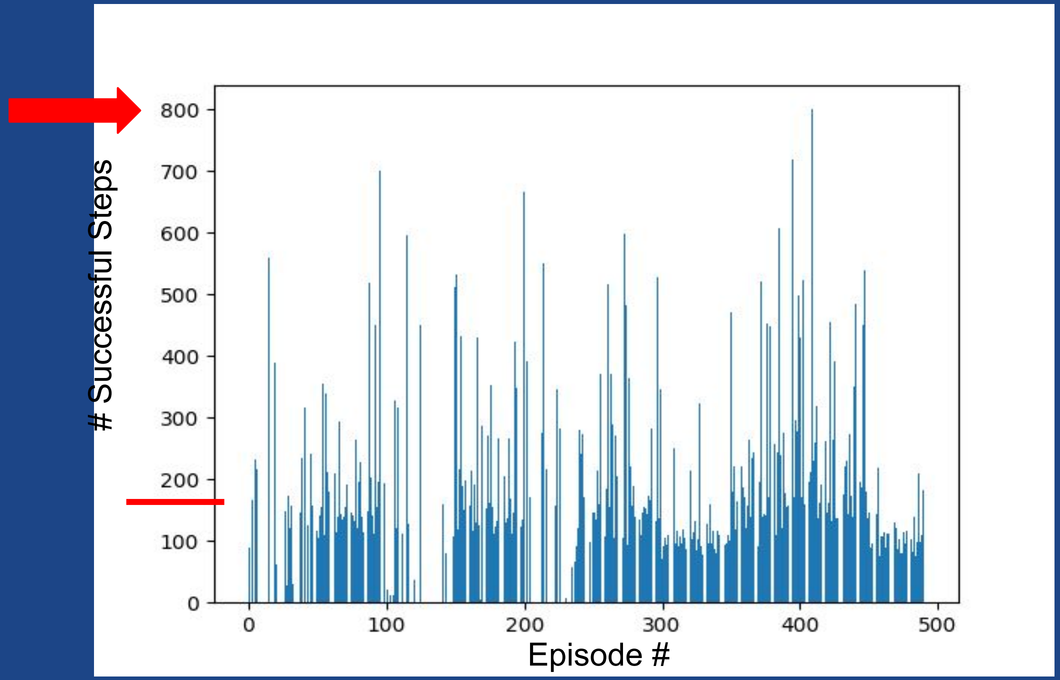


PPO Categorical

- Discretizing the action space showed a marked improvement from the PPO with continuous action space.
- Number of successful steps after 500 episodes.



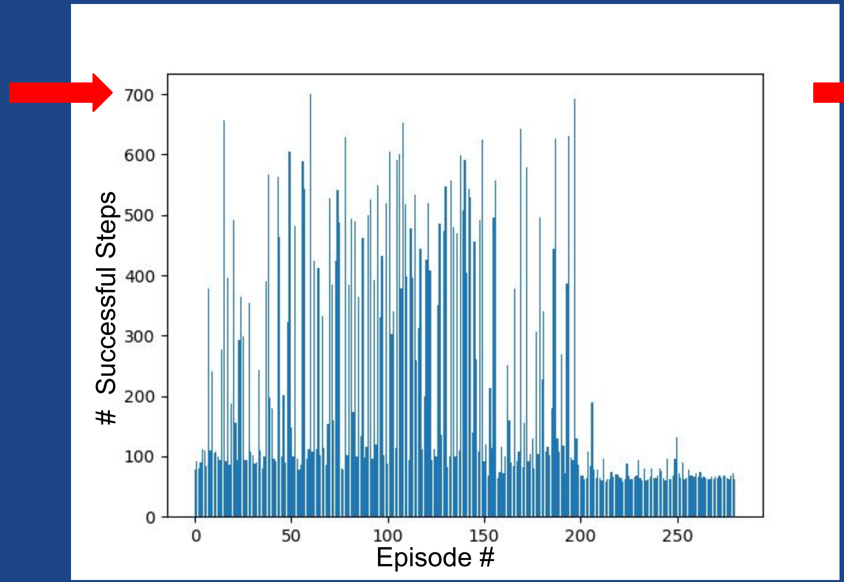
Continuous Action Space



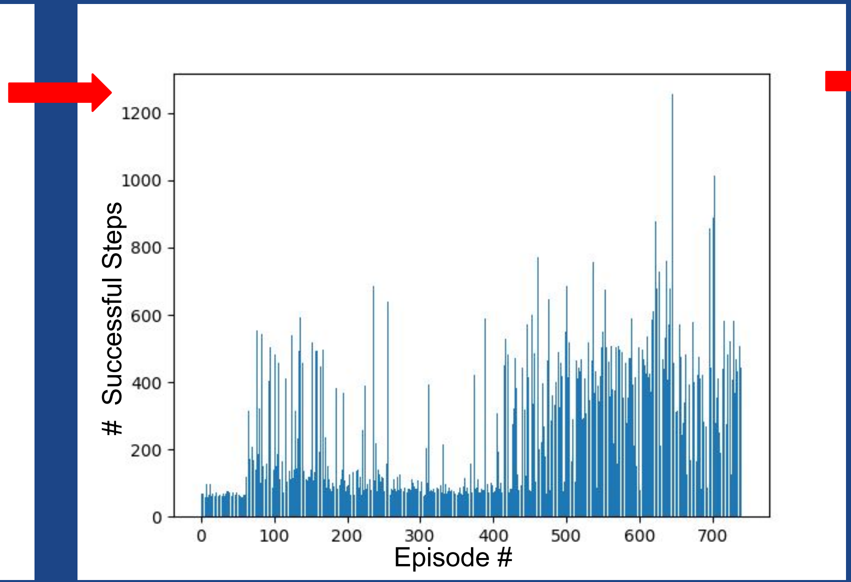
Discrete Action Space

PPO Categorical

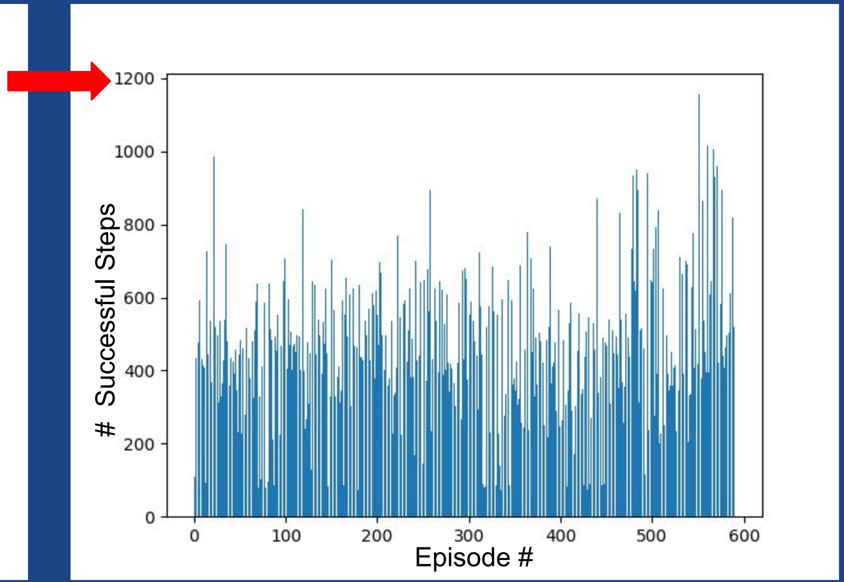
- During the start there were some crashes but then it learnt pretty quickly.
- During some of the episodes the aircraft stayed inside the target zone for longer than half of the time.



Episodes 0 - 280



Episodes 281 - 1030



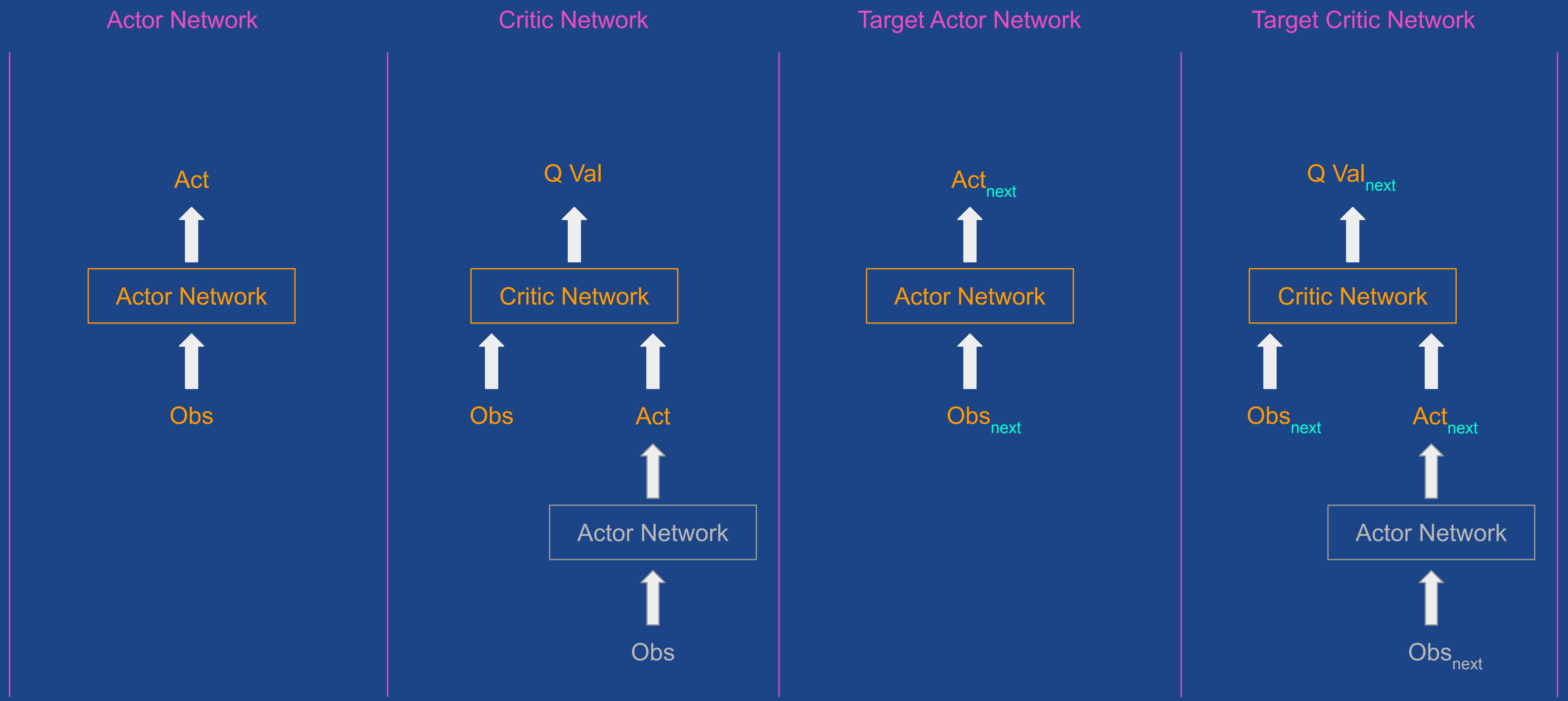
Episodes 1031- 1640

RL Agents

1. REINFORCE
2. Proximal Policy Optimization (PPO)
    - Continuous
    - Categorical
3. Deep Deterministic Policy Gradient
    - Original
    - Quick Ending
        - Target Start
4. Soft Actor-Critic

DDPG

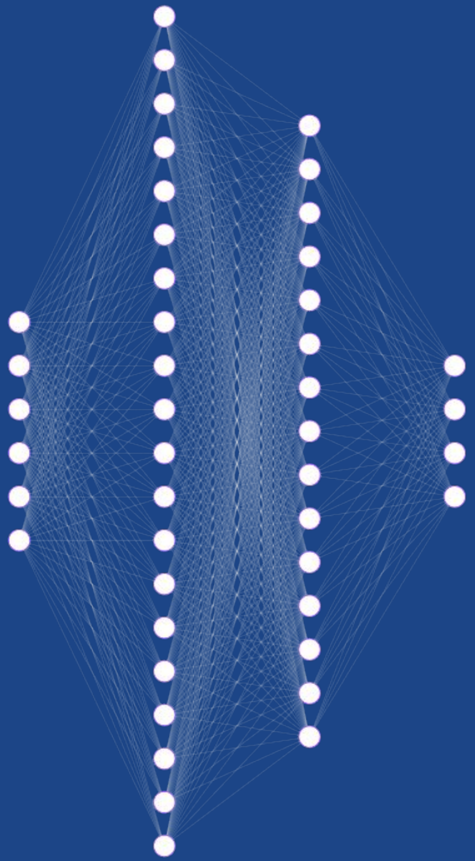
Setup


Actor Network
Critic Network
Target Actor Network
Target Critic Network
Act
Actor Network
Obs
Q Val
Critic Network
Obs
Act
Actor Network
Obs
Act next
Actor Network
Obs next
Q Val next
Critic Network
Obs next
Act next
Actor Network
Obs next

DDPG

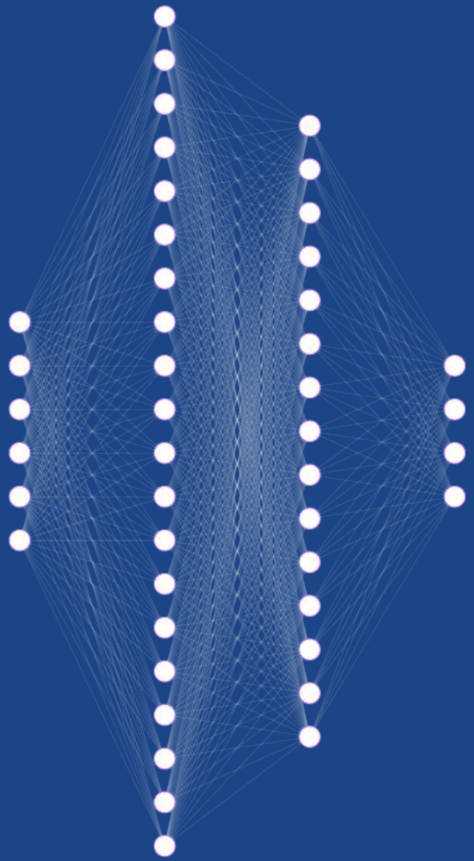
Setup

Actor Network



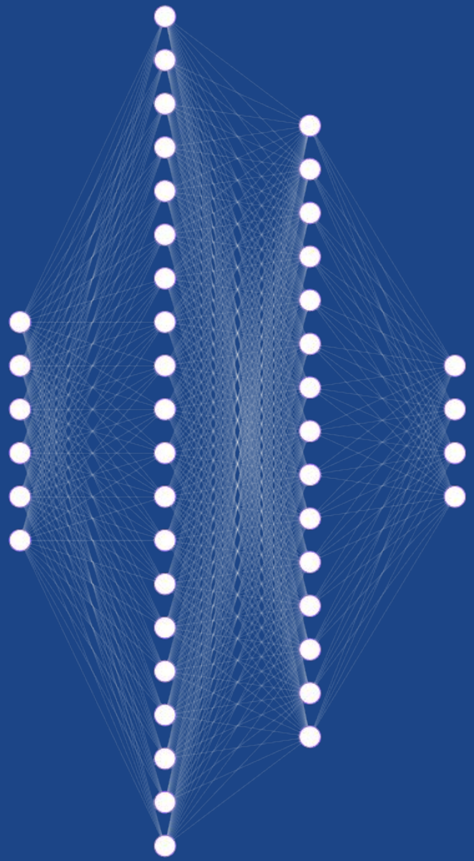
Architecture: 8 x 400 x 300 x 4

Critic Network



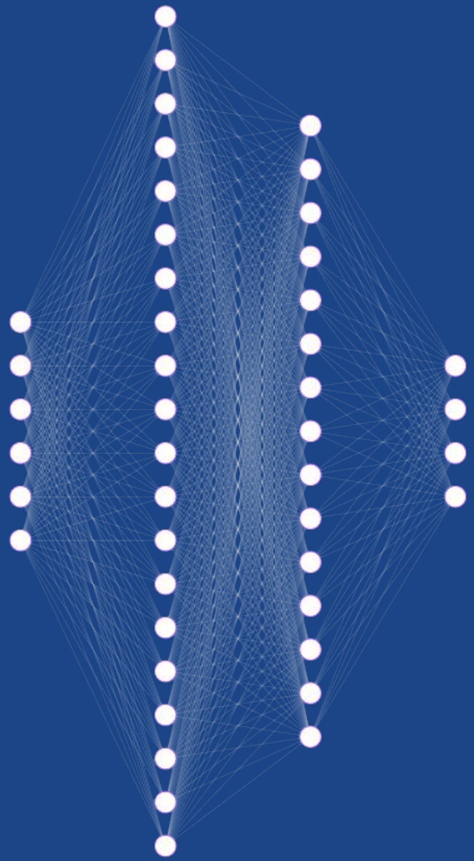
Architecture: 8 x 400 x 300 x 4

Target Actor Network



Architecture: 8 x 400 x 300 x 4
Tau: 0.001

Target Critic Network



Architecture: 8 x 400 x 300 x 4
Tau: 0.001

DDPG

Hyper-parameters

Learn: **Every 20 Steps**

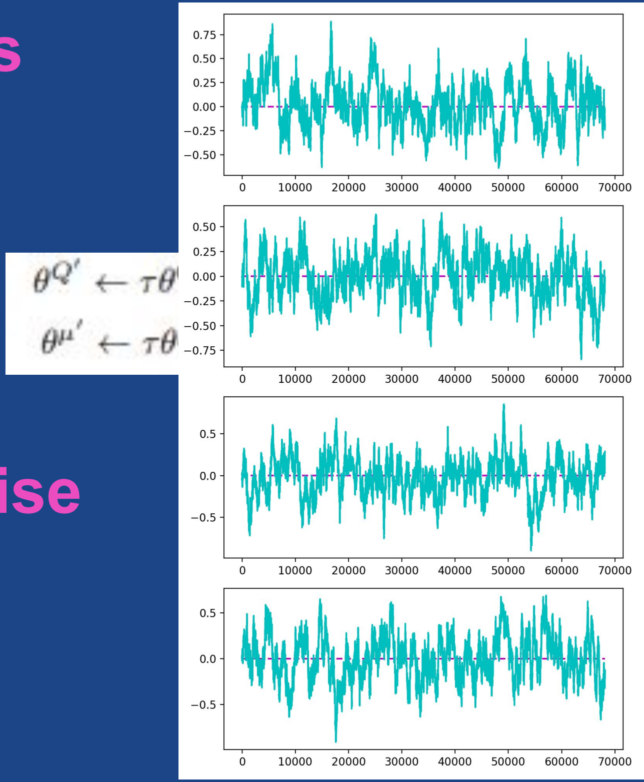
Learning Rates: **2.5×10^{-5}** (Actor, Target Actor), **2.5×10^{-4}** (Critic, Target Critic)

Max Replay Buffer Size: **1×10^{6} Steps**

Batch Size: **5000**

Target Network Soft Update: **0.001**

Exploration: **Ornstein-Uhlenbeck Noise**



Noise added to each action space item

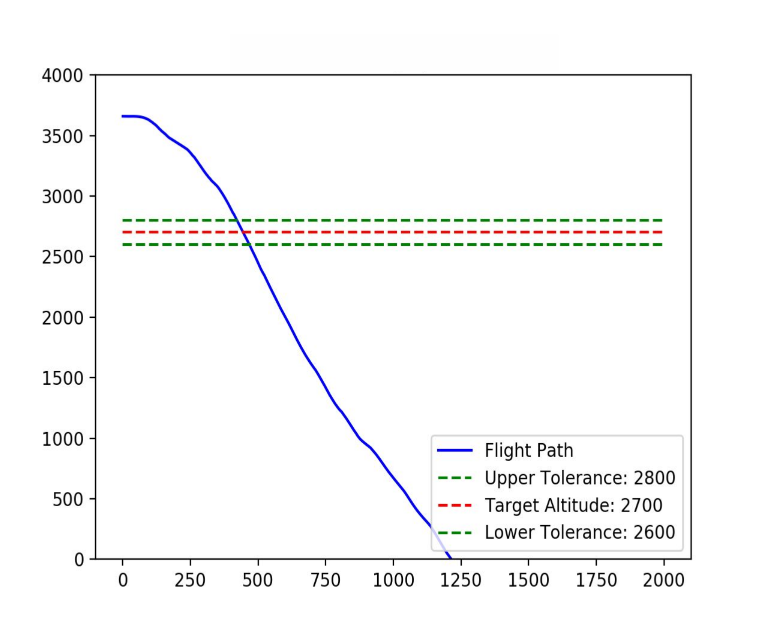
(tends to μ, as $t \rightarrow \infty$)

$\mu = 0$

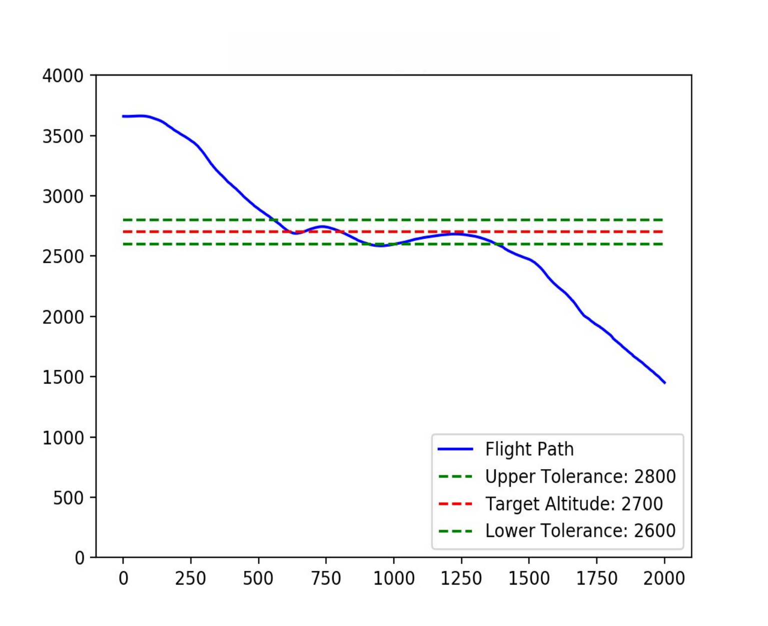
DDPG

Performance and Evaluation: Flight Trajectory

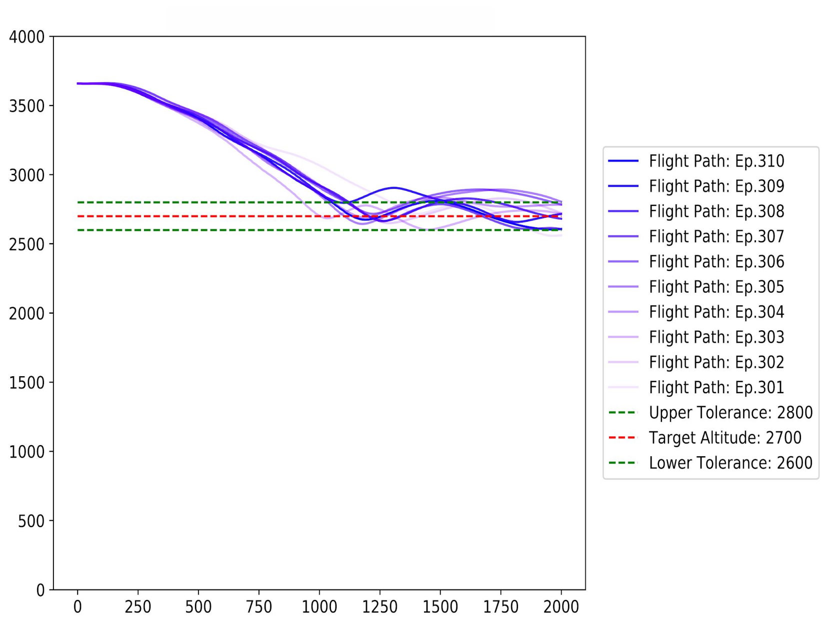
Episode 0-500



Episode 500-1000

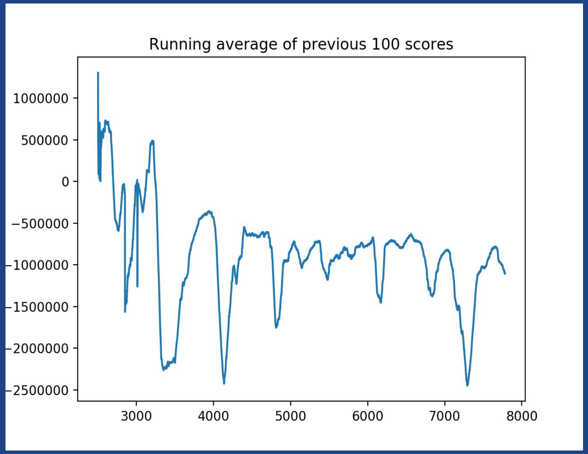


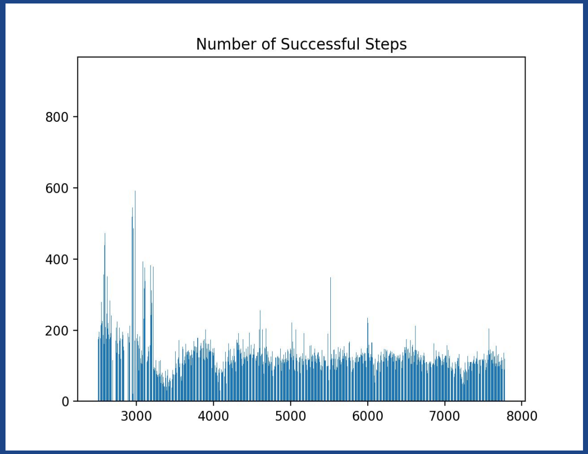
Episode 1500-2000



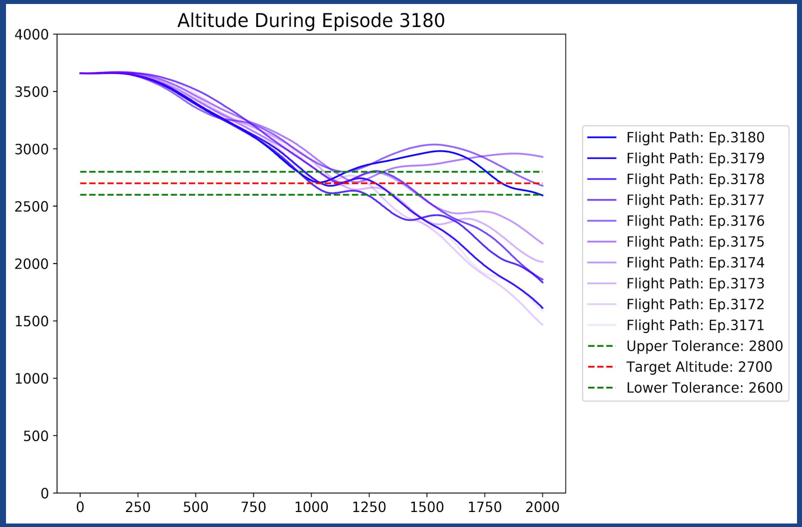
DDPG

Episode 4000 onwards, the agent did not show signs of progress.

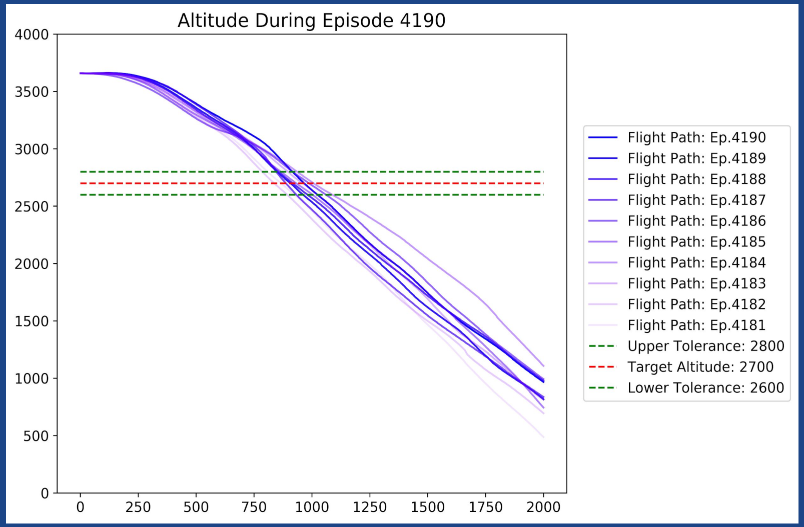




Episode 3171-3180



Episode 4181-4190



RL Agents

1. REINFORCE
2. Proximal Policy Optimization (PPO)
    - Continuous
    - Categorical
3. Deep Deterministic Policy Gradient
    - Original
    - Quick Ending
        - Target Start
4. Soft Actor-Critic

DDPG

Convinced that the structure of our reward OR episode was the problem

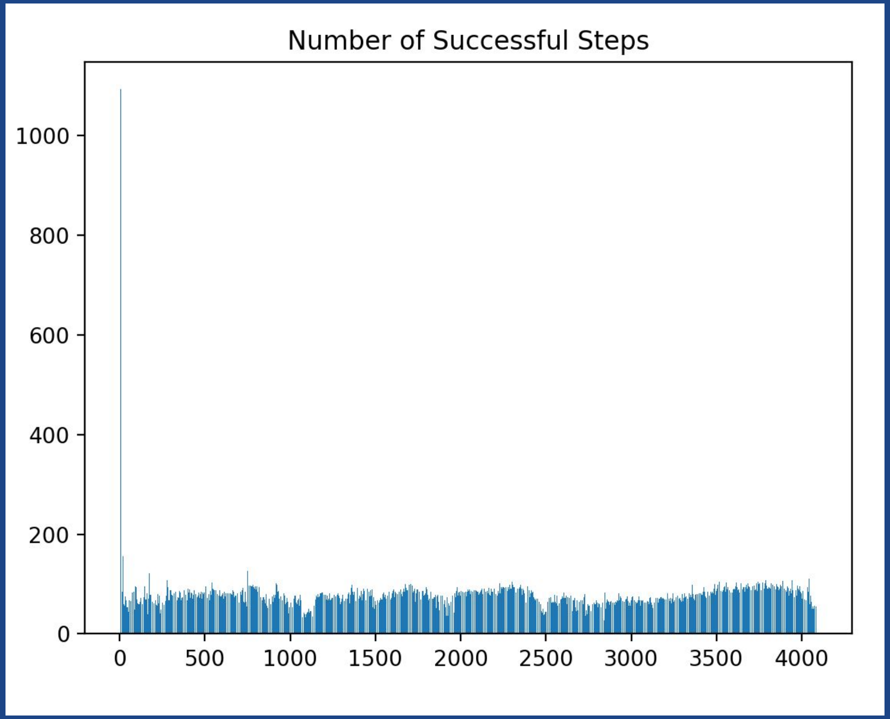
Δ Episode ends whenever plane leaves the target zone after entering it once.

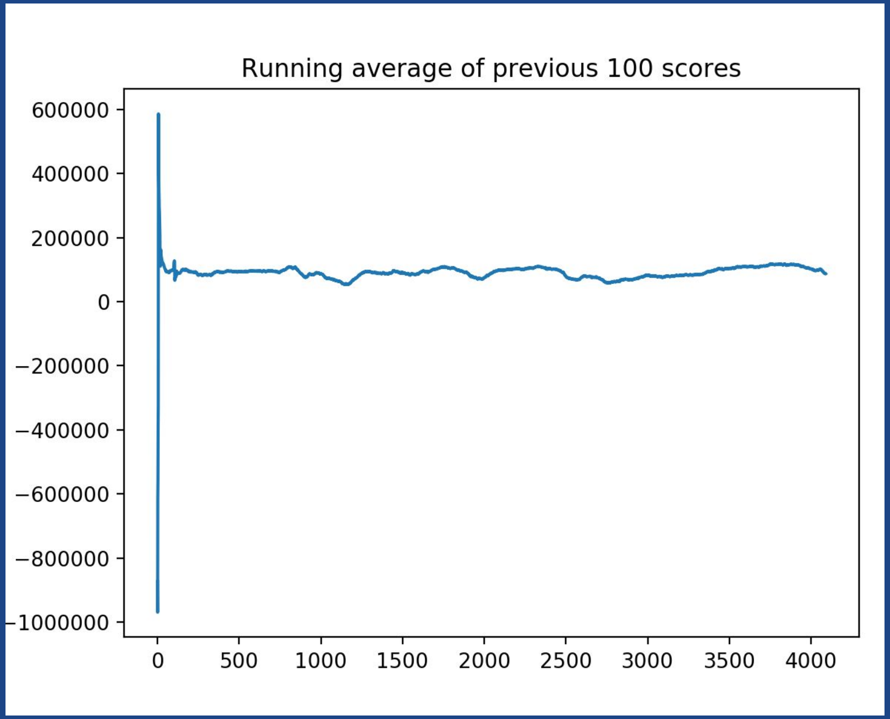
Rationale

- Shorter unsuccessful episodes
- Agent should learn to stay in the target zone longer

DDPG

Performance and Evaluation: Number of Successful Steps & Average Score

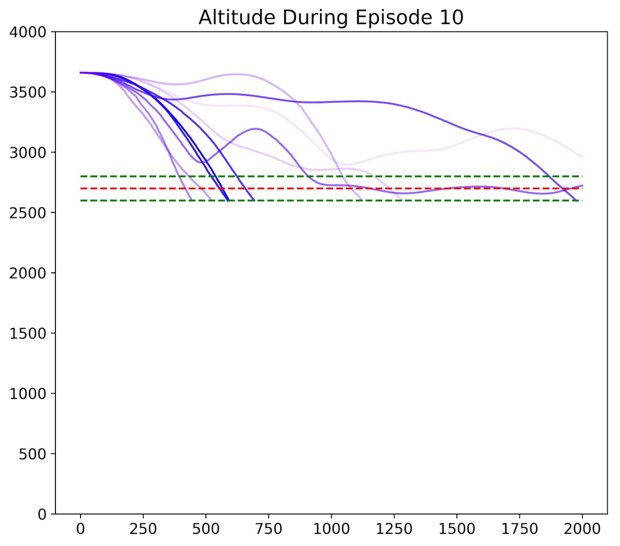




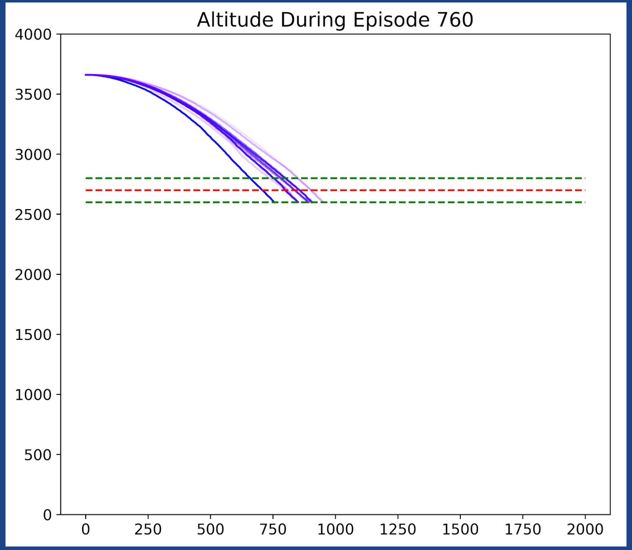
DDPG

Performance and Evaluation: Flight Trajectory

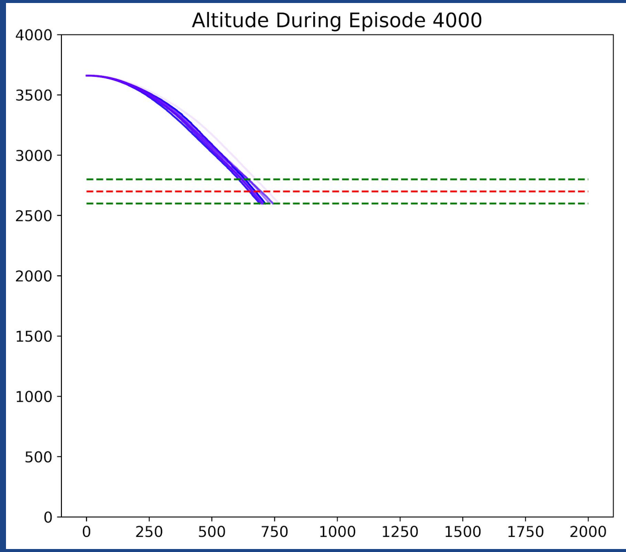
Episode 1-10



Episode 751-760



Episode 3991-4000



RL Agents

1. REINFORCE
2. Proximal Policy Optimization (PPO)
   - Continuous
   - Categorical
3. Deep Deterministic Policy Gradient
   - Original
   - Quick Ending
     - Target Start
4. Soft Actor-Critic

DDPG

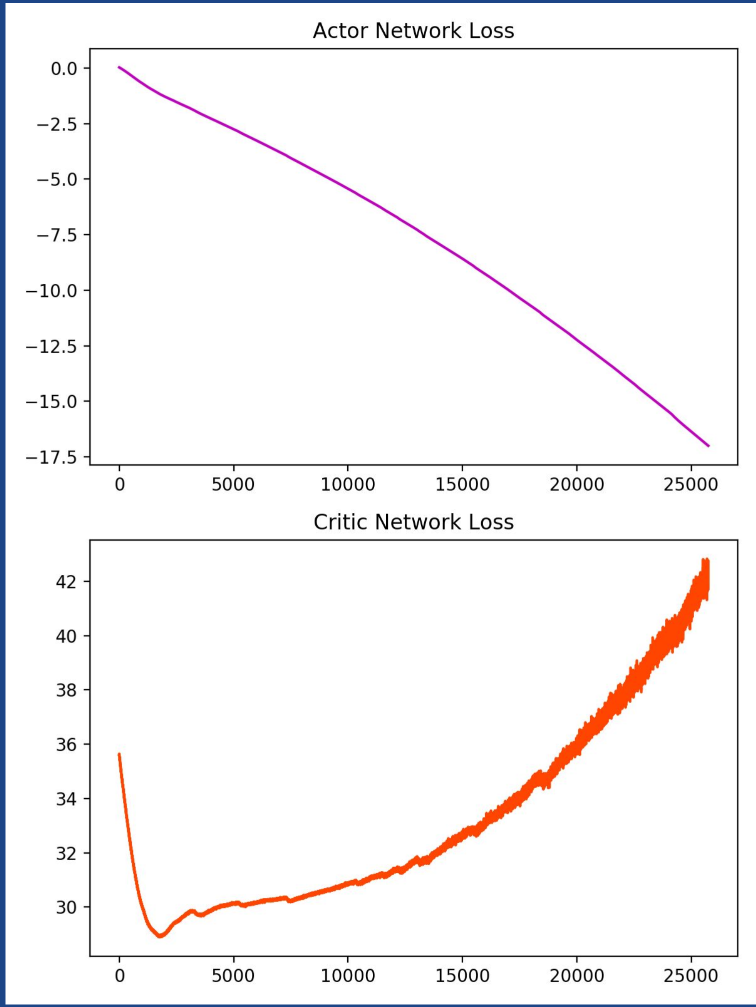
Δ Episode starts with plane in target zone

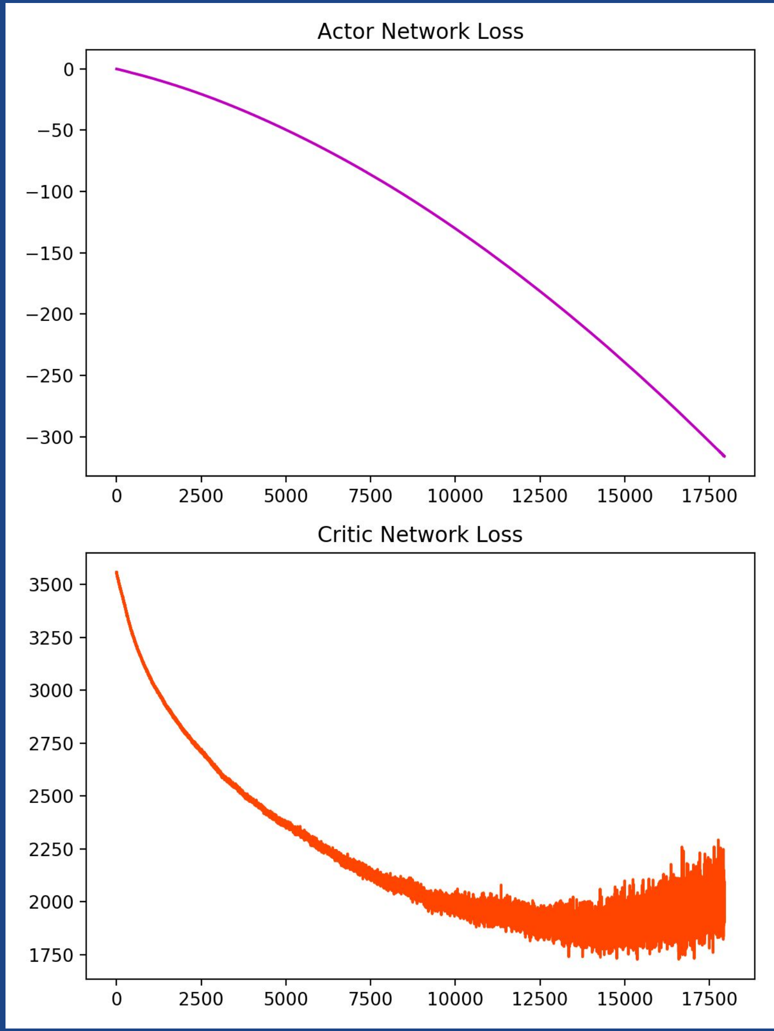
Δ Hyper parameter tuning

- Policy Gradient methods (especially DDPG) are very sensitive
- Learning Rates: **2.5×10^{-7}** (Actor, Target Actor), **2.5×10^{-6}** (Critic, Target Critic)
- Scaled reward

DDPG

Performance and Evaluation: Network Loss

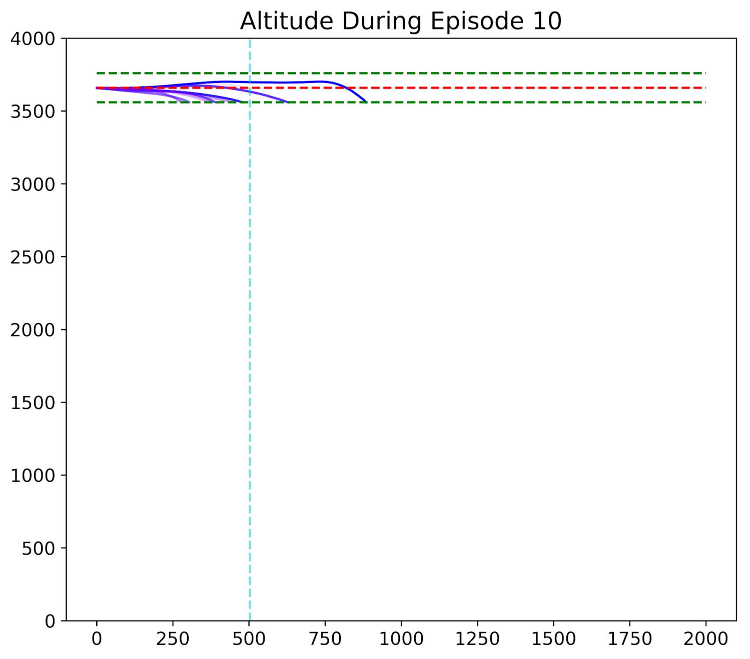




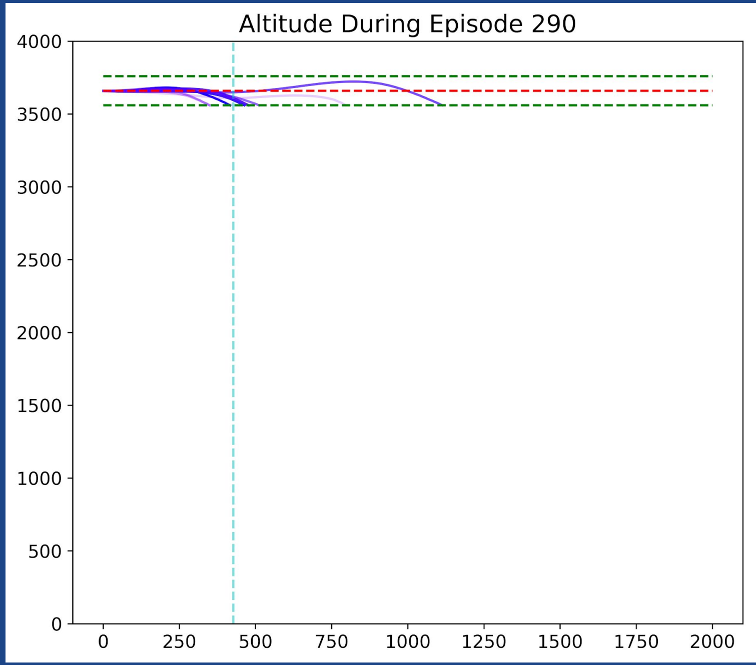
DDPG

Performance and Evaluation: Flight Trajectory

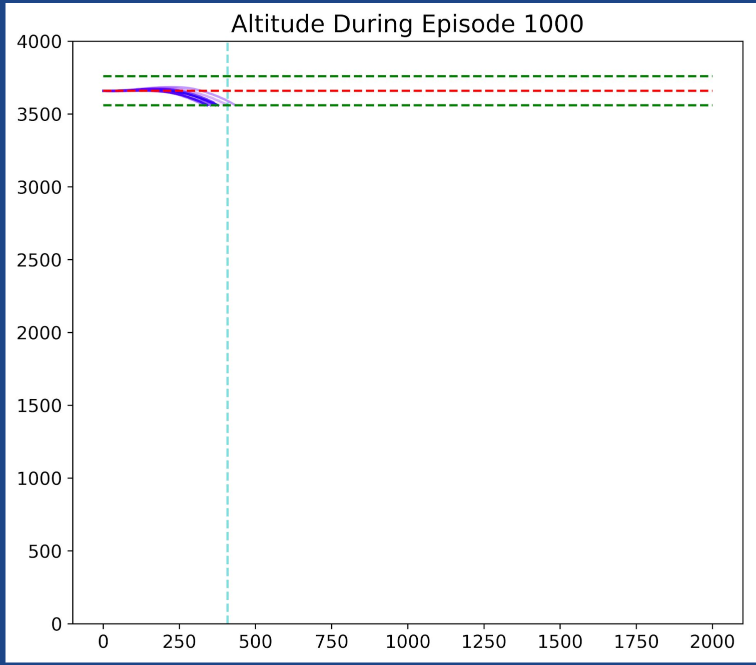
Episode 1-10



Episode 281-290



Episode 991-1000



DDPG

Verdict

DDPG has proven to be extremely sensitive.

Different episode designs and reward schemes, require a very particular combination of hyperparameters.

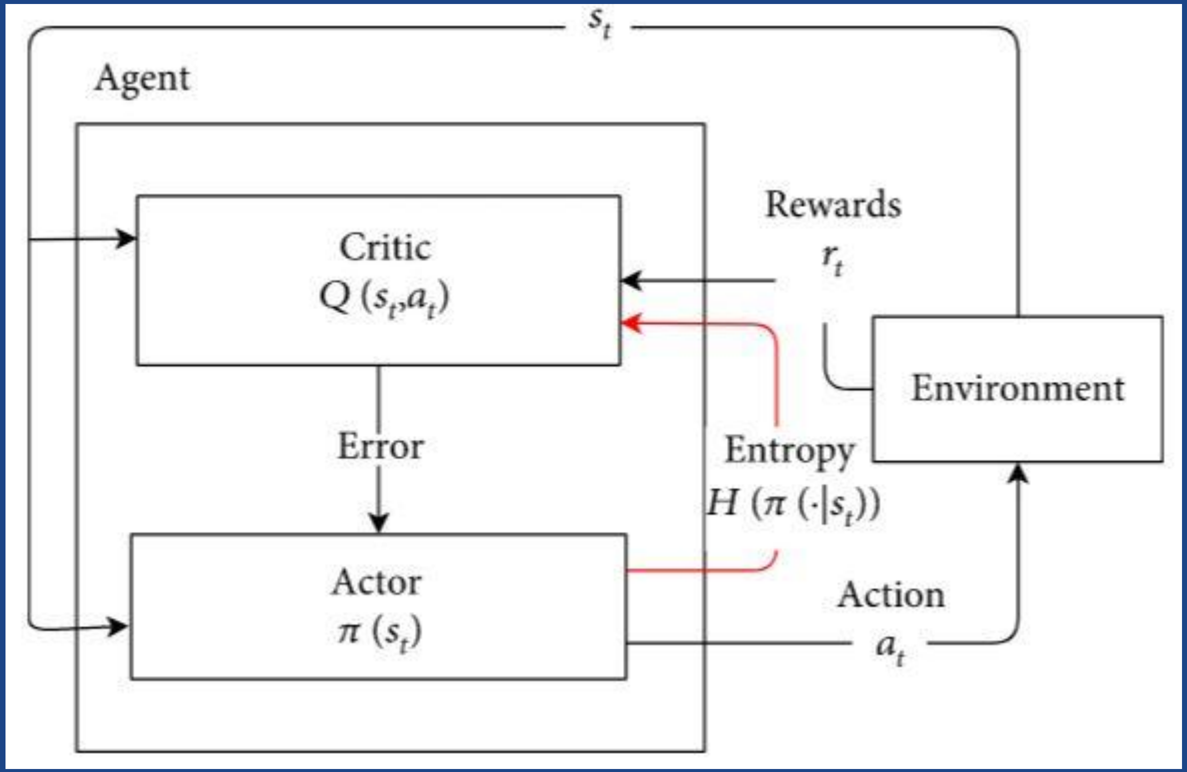
“It is often reported that DDPG suffers from instability in the form of sensitivity to hyper-parameters and propensity to converge to very poor solutions or even diverge.” [1]

RL Agents

1. REINFORCE
2. Proximal Policy Optimization (PPO)
    - Continuous
    - Categorical
3. Deep Deterministic Policy Gradient
    - Original
    - Quick Ending
        - Target Start
4. Soft Actor-Critic

Soft Actor-Critic

- On-policy algorithms are expensive in terms of sample complexity
- Some off-policy methods like DDPG can be extremely sensitive to hyperparameters despite being sample efficient.
- Soft Actor-Critic (SAC) is an off-policy actor-critic deep RL algorithm based on the maximum entropy RL framework
- The actor aims to maximize expected reward while also maximizing entropy, i.e., to succeed at the task while acting as randomly as possible.



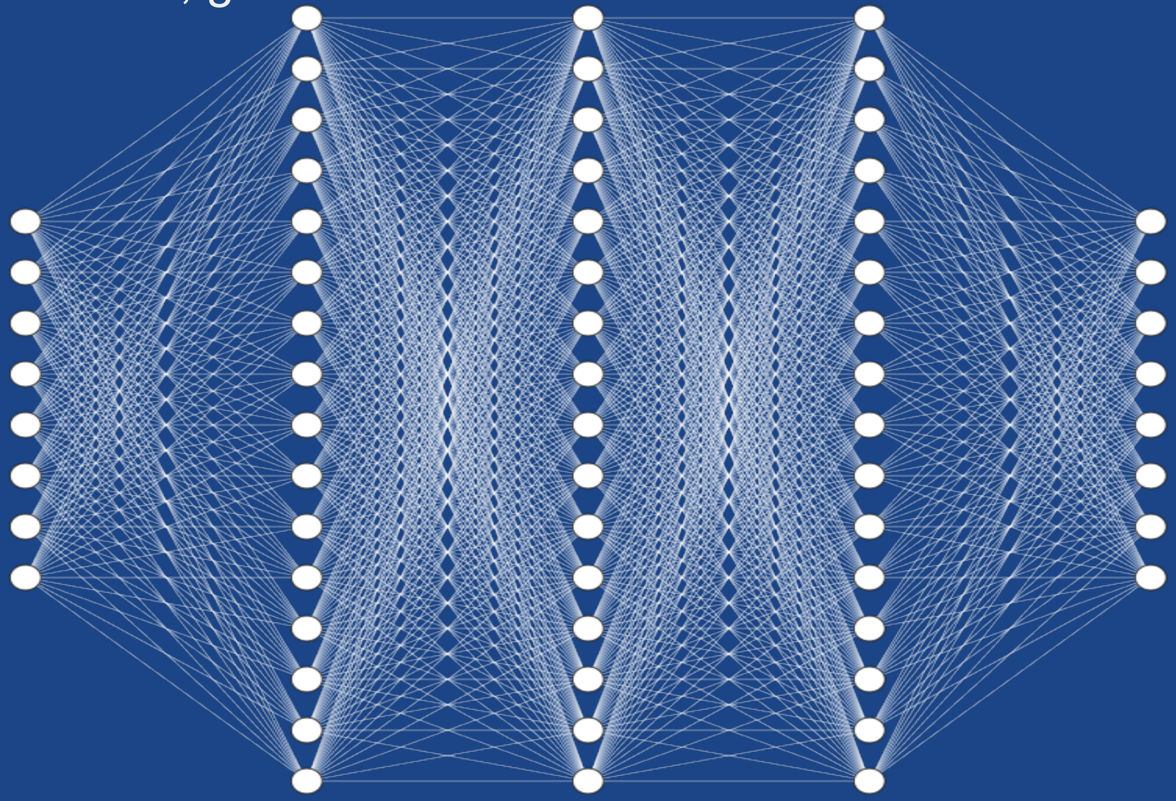
Soft Actor-Critic

- To cater for less time available for training, the observation space was simplified, to see if the algorithm would respond (learn) quickly.
- Since the primary goal was to train the model to change altitude without any constraints on attitude, therefore, the observation space was simplified to only 3 parameters:
  - Indicated Airspeed
  - Vertical Velocity
  - Relative altitude
- Two iterations of the algorithm were run with different hyperparameters and they showed good results.

Soft Actor-Critic

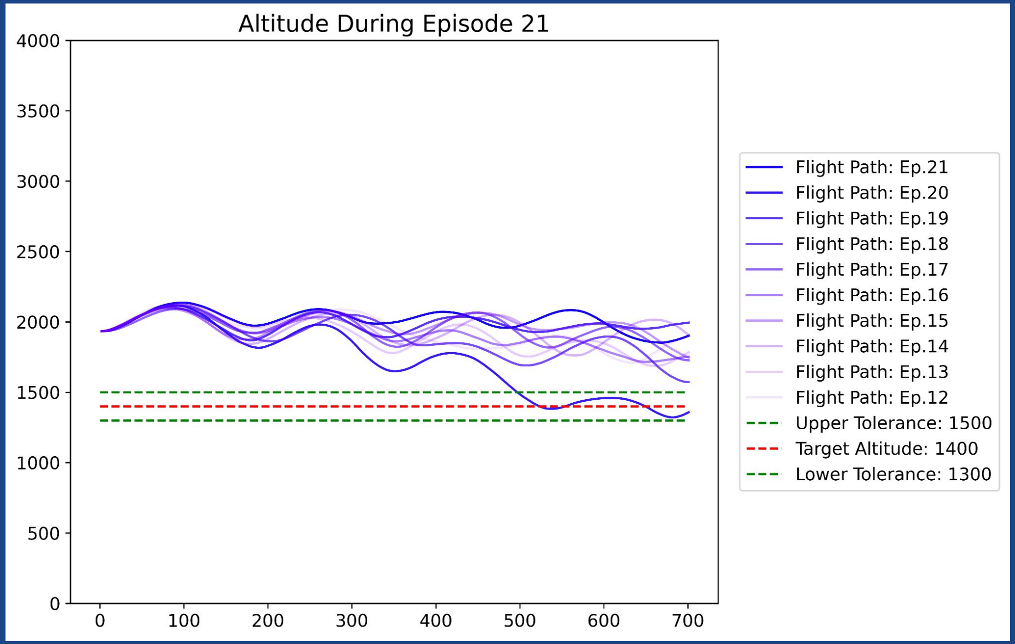
- Same networks were used for Actor, Critic and Value networks except for the output layer.
- Actor outputs mean and std for 4 actions.
- Critic outputs a single value function for the state with state action pair as input.
- Value network estimates value function, given state.

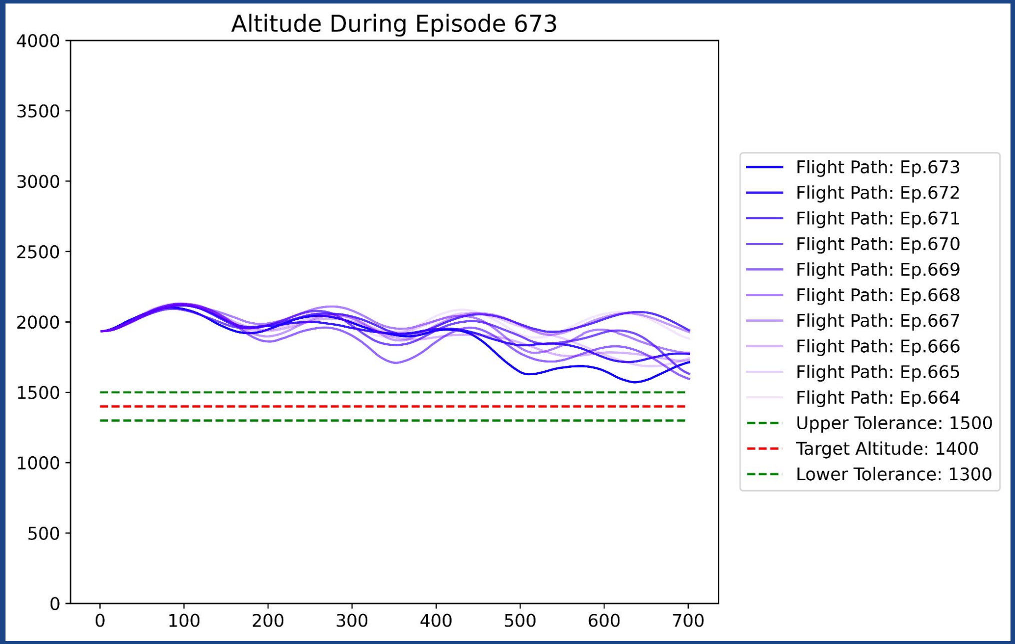
Architecture: 3 x 256 x 256 x 256 x 8
Activation: ReLU
Output Activation: Linear
LR: 3 x 10^-3
Reward Scale: 1.0



Soft Actor-Critic

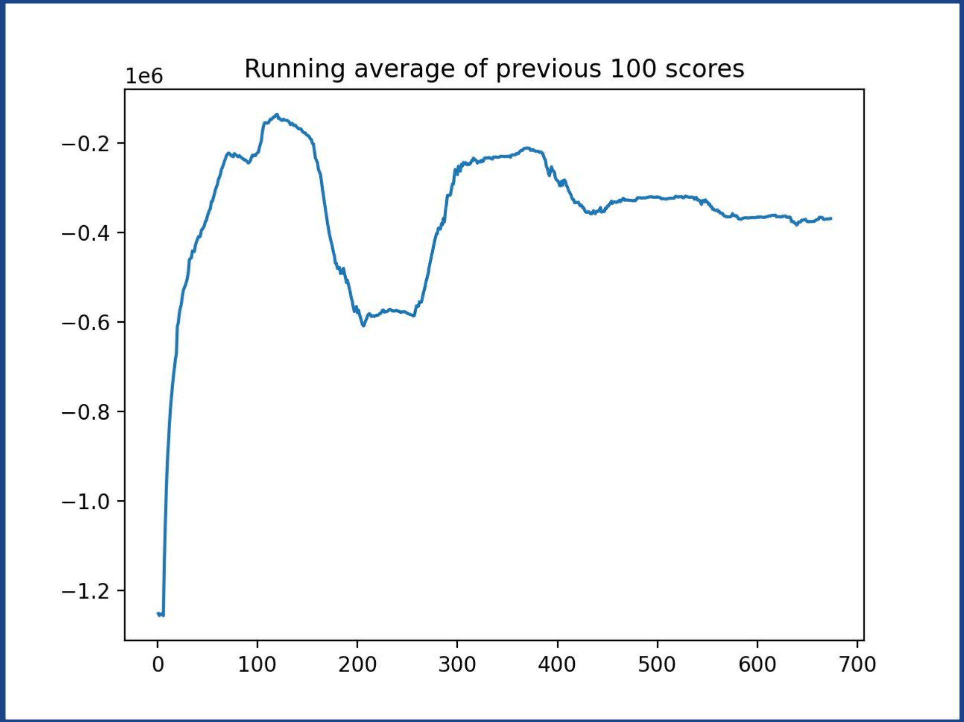
- Initial SAC agent was very conservative and did not explore a lot despite being formulated on maximum entropy framework.
- Over the course of training for close to 700 episodes, the trend in flight trajectory changed only slightly.

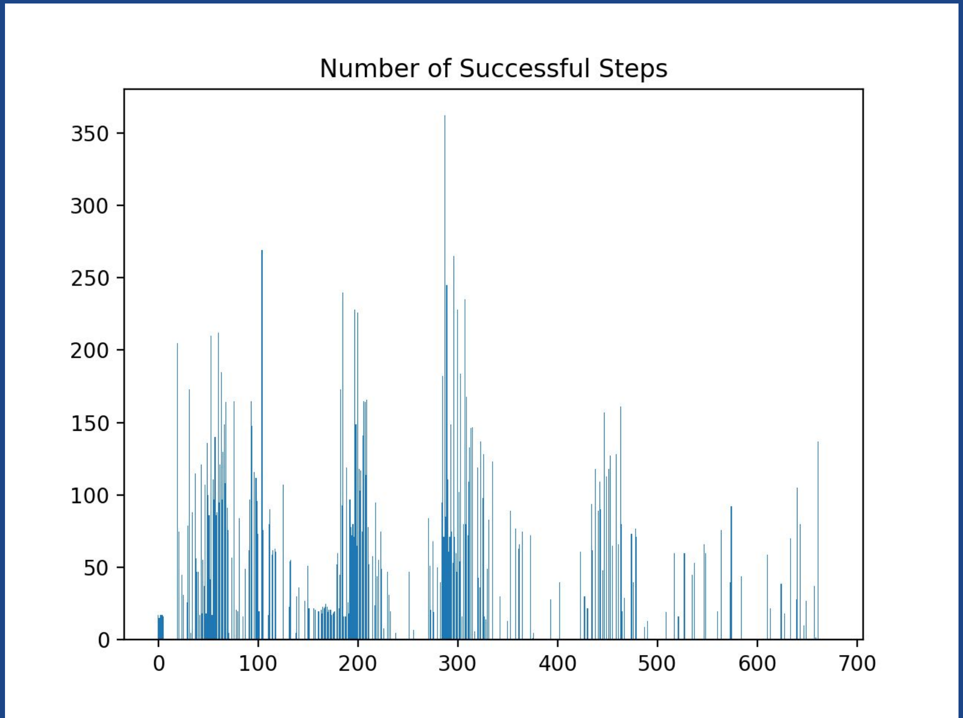




Soft Actor-Critic

- Despite experiencing some episodes with very large positive rewards, performance plateaued.





Soft Actor-Critic

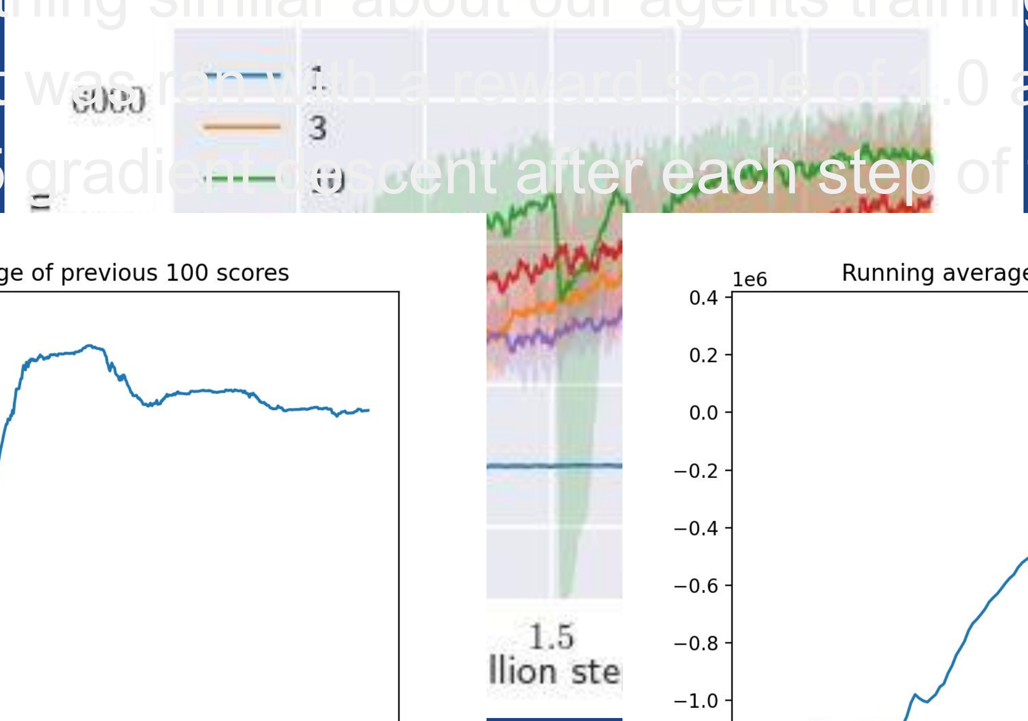
- To look into the issues affecting the performance of our agent we went back to the paper
- Soft Actor-Critic paper discusses the effects of some of the most important hyperparameters [2]:
    - Reward Scale
    - Target value update smoothing constant

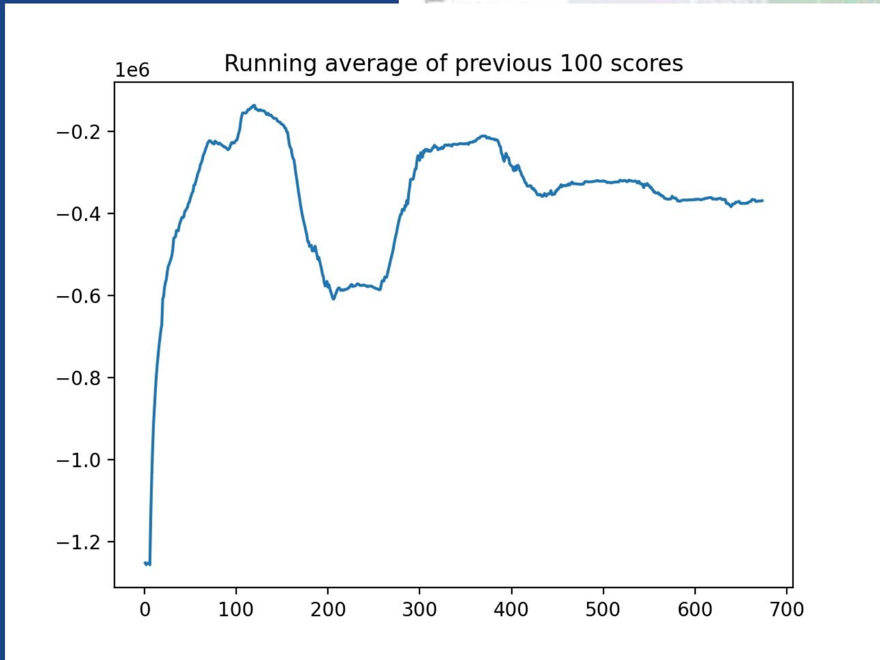
$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t)\left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - r(\mathbf{s}_t, \mathbf{a}_t) - \gamma V_{\bar{\psi}}(\mathbf{s}_{t+1})\right)$$

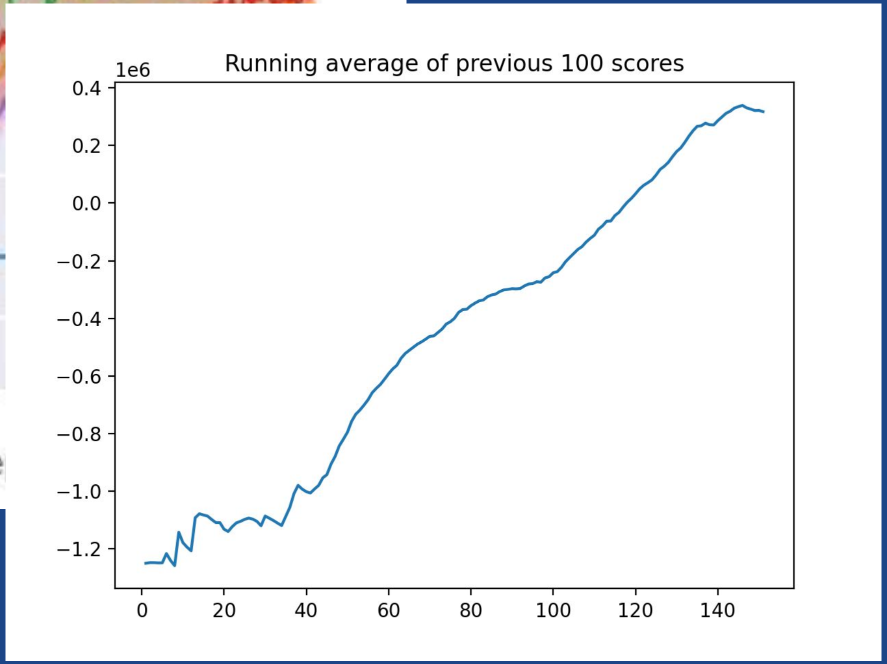
- Reward scale gives more weightage to rewards, that the agent collects, while calculating the loss of the critic network.
- With the right reward scaling, the model balances exploration and exploitation, leading to faster learning.

Soft Actor-Critic

- SAC paper shows sensitivity to reward scaling with this plot for Ant-v1 environment
- We observed something similar about our agents training as well
- The first experiment was ran with a reward scale of 1.0 and the second one with 10.0 and with 5 gradient descent after each step of the agent

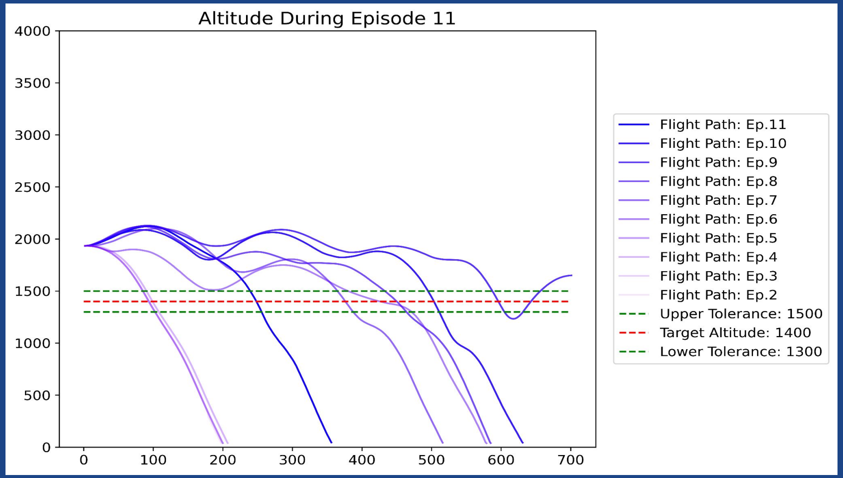


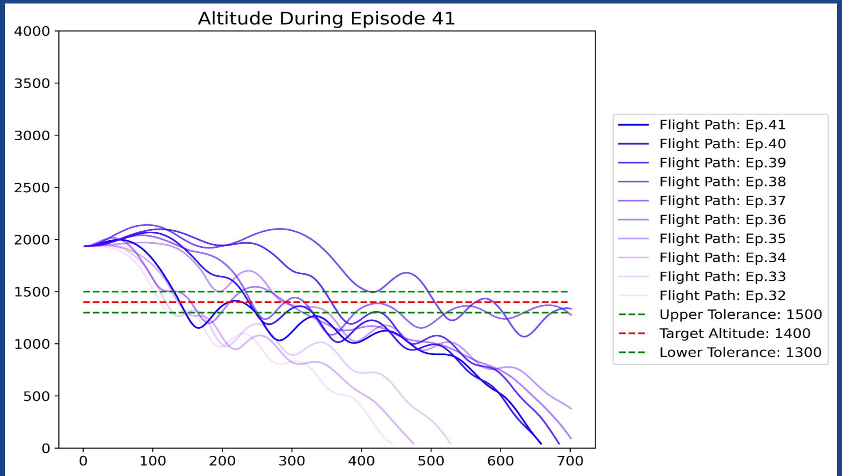


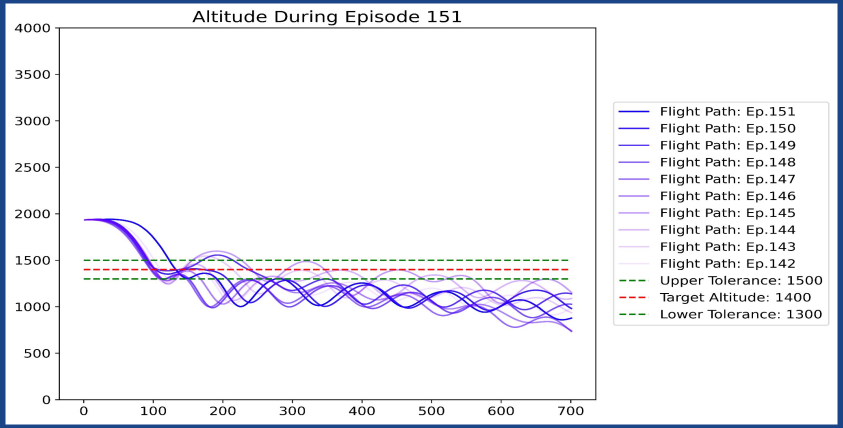


Soft Actor-Critic

- With reward scale of 10, the agent did keep trying to stay closer to high reward zone.

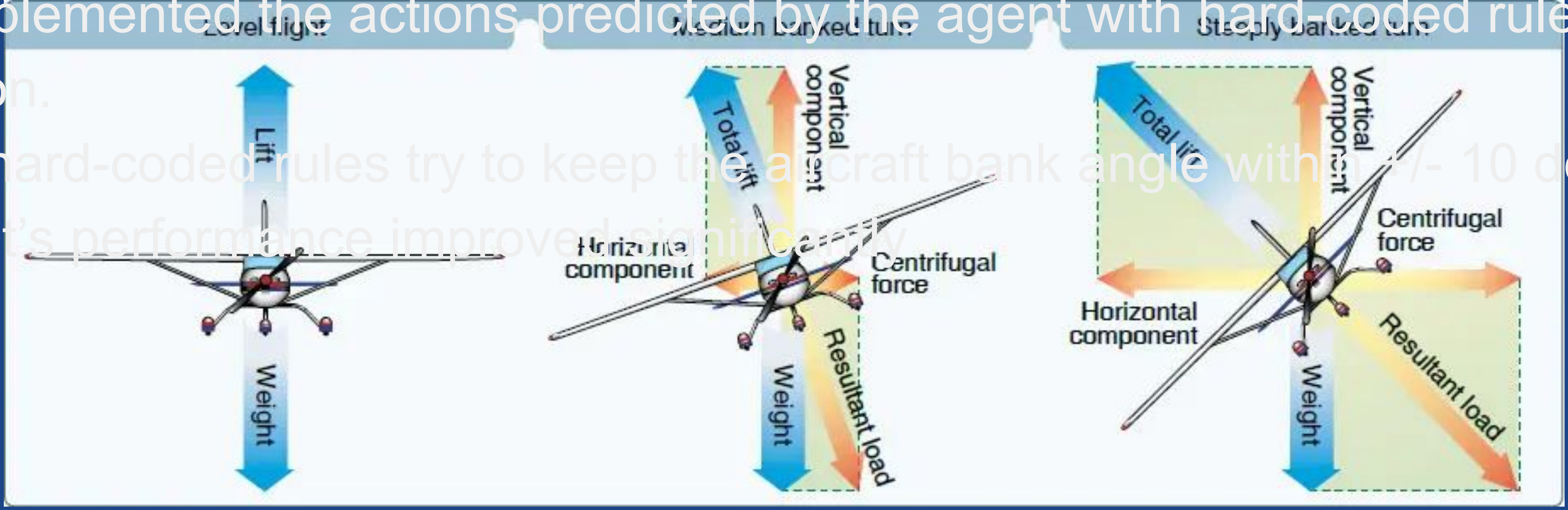






Soft Actor-Critic

- Elevator and Throttle controls make more intuitive sense during straight & level flight.
    - Pitch-up / throttle increase = Aircraft moves up
    - pitch-down / throttle decrease = Aircraft moves down
- During the turn the force vectors act in complicated ways making it difficult for the agent to reorient the aircraft.
- Complemented the actions predicted by the agent with hard-coded rules for aileron.
- The hard-coded rules try to keep the aircraft bank angle within +/- 10 degrees
- Agent's performance improved significantly



Limitations

1. Environment issues slowed down the progress significantly
2. Sampling trajectories from the environment was very expensive, therefore, on-policy algorithms like PPO became infeasible
3. Reward function was not intuitively designed
4. Not adequate focus on hyper-parameter tuning

Future Work

Our roadmap moving forward will be:

1. Build a fully compatible OpenAI Gym environment
2. Try to run X-Plane 11 on cloud
3. Standardize a set of metrics to gauge performance
4. Improve reward shaping w.r.t the environmental context
5. Explore Curriculum Learning - moving from easier tasks to more difficult ones
6. Imitation learning paired with off-policy methods

Website Link:

https://priya007007.github.io/Website527/

Individual Contributions

Muhammad Rizwan Malik

- REINFORCE, PPO, SAC
- Weekly Presentation
- Project Documentation

Muhammad Oneeb Ul Haq Khan

- REINFORCE, DDPG
- Weekly Presentation
- Project Documentation

Martin Huang

- DDPG
- Weekly Presentation

Krishnateja Gunda

- REINFORCE
- Project Documentation

Rengapriya Aravindan

- Project Website
- Project Documentation

References

1. Matheron, Guillaume, Nicolas Perrin, and Olivier Sigaud. "The problem with DDPG: understanding failures in deterministic environments with sparse rewards." *arXiv preprint arXiv:1911.11679* (2019).
2. Haarnoja, Tuomas, et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." International conference on machine learning. PMLR, 2018.
3. Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).
4. Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." Advances in neural information processing systems. 2000.

Thank you!