

How RL Learns To Manage The Skies

Applying PPO (Proximal Policy Optimization) in ATC environment

CS:138: Intro to Reinforcement Learning

Peter Nadel, Daana Masumi, Zain Abbas

12/19/25

Reinforcement Learning in ATC

- ATC involves dynamic decision making under uncertainty
- In this project, we model the problem as a single decision-making agent (e.g. aircraft) who learns to navigate in a continuous environment
- Using BlueSky Gym environments, the task is modelled as Partial Observable Markov Decision Process (POMDP), allowing the agent to receive partial observations of the airspace environment

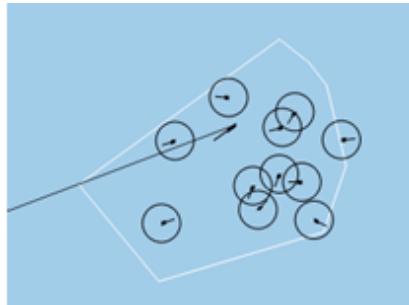


Review of Deep Reinforcement Learning Approaches for Conflict Resolution in Air Traffic Control

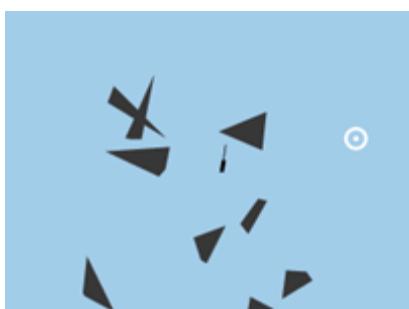
BlueSky Gym

- BlueSky Gym is an open-source simulator that provides ATC environments
- It is based on Markov Decision Process (MDP) logic
- To test our agent, we used following BlueSky-Gym environments:

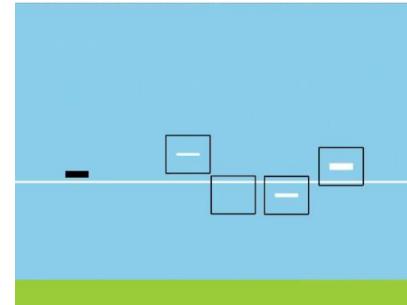
SectorCREnv-v0



StaticObstacleEnv-v0



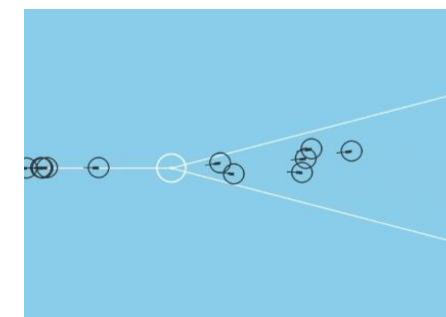
VerticalCREnv-v0



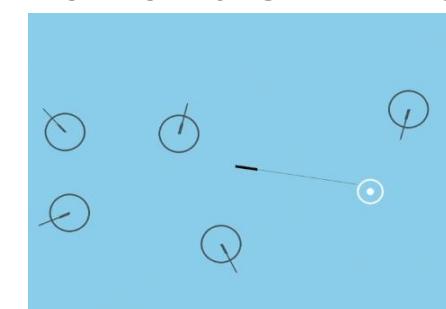
DescentEnv-v0



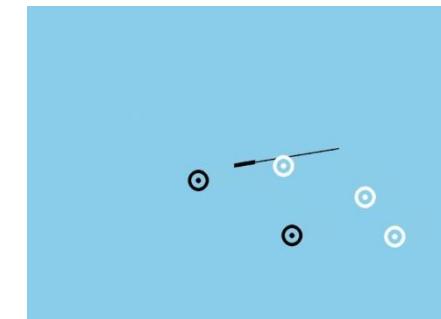
MergeEnv-v0



HorizontalCREnv-v0



PlanWaypointEnv-v0



PPO (Proximal Policy Optimization)

- To train our agent in POMDP continuous environment, we chose PPO algorithm since it has shown promising results in aircraft collision avoidance
- The PPO algorithm is based on actor-critic method, a hybrid architecture combining value-based and policy-based methods to stabilize the training.

Policy Objection Function

- The Policy Objective function is defined as:

$$L^{PG}(\theta) = E_t[\log \pi_\theta(a_t | s_t) A_t] \quad (I)$$

where:

$L(\theta)$: Policy loss

E : expectation over timesteps

$\log \pi_\theta(a_t | s_t)$: is the log probability of taking that action at that state

A_t : Advantage if $A > 0$, this action is better than other action possible at that state

$$A_t = Q(s_t, a_t) - V(s_t)$$

$Q(s_t, a_t)$: action-value (expected return after taking action a_t in state s_t)

$V(s_t)$: state-value (expected return from state s_t under the policy)

By taking the gradient ascent step on this function, we allow our agent to take actions that lead to higher reward and avoid harmful actions.

Caveats:

- A smaller step size leads to slower training process
- A larger step size leads to high variability in the training

PPO: Clipped Surrogate Function

- To obtain optimal policy, PPO constrains the policy update with a new objective function called *Clipped surrogate function*. This function constrains the policy change in a smaller range using a clip

$$L^{CLIP}(\theta) = E_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon)A_t)] \quad (II)$$

where:

$$r_t(\theta) = \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta_{old}}(a_t | s_t)} :$$

It is the probability of taking a_t at state s_t in the current policy divided by the old policy

If $r_t(\theta) > 1$, the action a_t at state s_t is more likely in the current policy than the old policy

If $r_t(\theta)$ is between 0 and 1, the action is less likely for the current policy relative to the old one.

Continue PPO

- Without clipping, policy gradient would maximize. This is called L^{CPI} conservative policy iteration

$$L^{CPI}(\theta) = E [r_t(\theta) A_t]$$

- The second term in equation II is the clip function.

$$\text{clip}(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon) A_t$$

- By clipping the probability ratio, it prevents large policy updates, even when the advantage signal is strong. The ratio $r_t(\theta)$ is restricted to $1 - \varepsilon \leq r_t(\theta) \leq 1 + \varepsilon$
- Finally, we take the minimum of the clipped and unclipped function, so the final objective is a lower bound on the unclipped objective.
- This allows us to ignore the drastic change in probability ratio. Samples that would cause too large policy updates stop contributing. Training becomes stable and monotonic.

PPO in ATC environment

- In ATC environment, PPO helps agent to act conservatively while learning air traffic controller:
 - Learns continuous control laws
 - Improves decisions incrementally
 - Avoids drastic policy shifts
 - Maintains safety while optimizing efficiency

Example:

- DescentEnv

Action representation:

$$\pi_\theta(v_s|s) = N(\mu_{vs}(s), \sigma_{vs}(s))$$

where,

v_s = vertical speed

$\mu_{vs}(s)$ = learned “ideal” descent rate

$\sigma_{vs}(s)$ = exploration width (standard deviation)

In DescentEnv, PPO learns:

- When to level off
- When to descend

Clipping objective function in PPO prevent large jumps in descent rate

Smooth convergence – realistic vertical profiles

As a result, the agent learns stable, human-like descent planning.

Recurrent PPO

- In an environment where memory is useful to observe a given state, we utilized PPO variant called Recurrent PPO.
- Recurrent PPO outperforms other algorithms specifically in StaticObstacleEnv-v0
- Recurrent PPO consists of PPO + LSTM:

$$\pi_{\theta}(a_t | s_t, h_t)$$

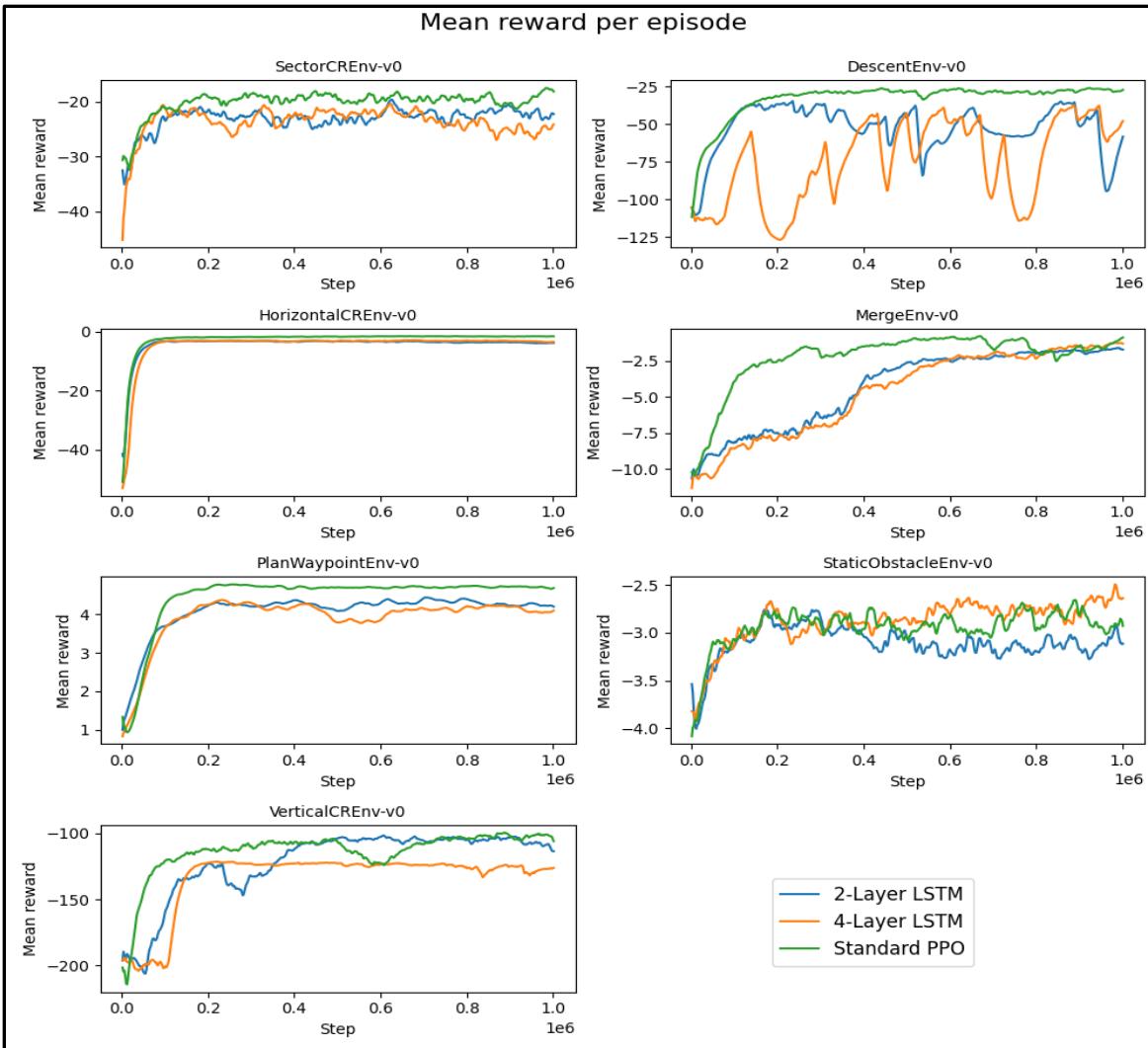
Where

- h_t = hidden states from the LSTM
- $h_t = \text{LSTM}(h_{t-1}, s_t)$

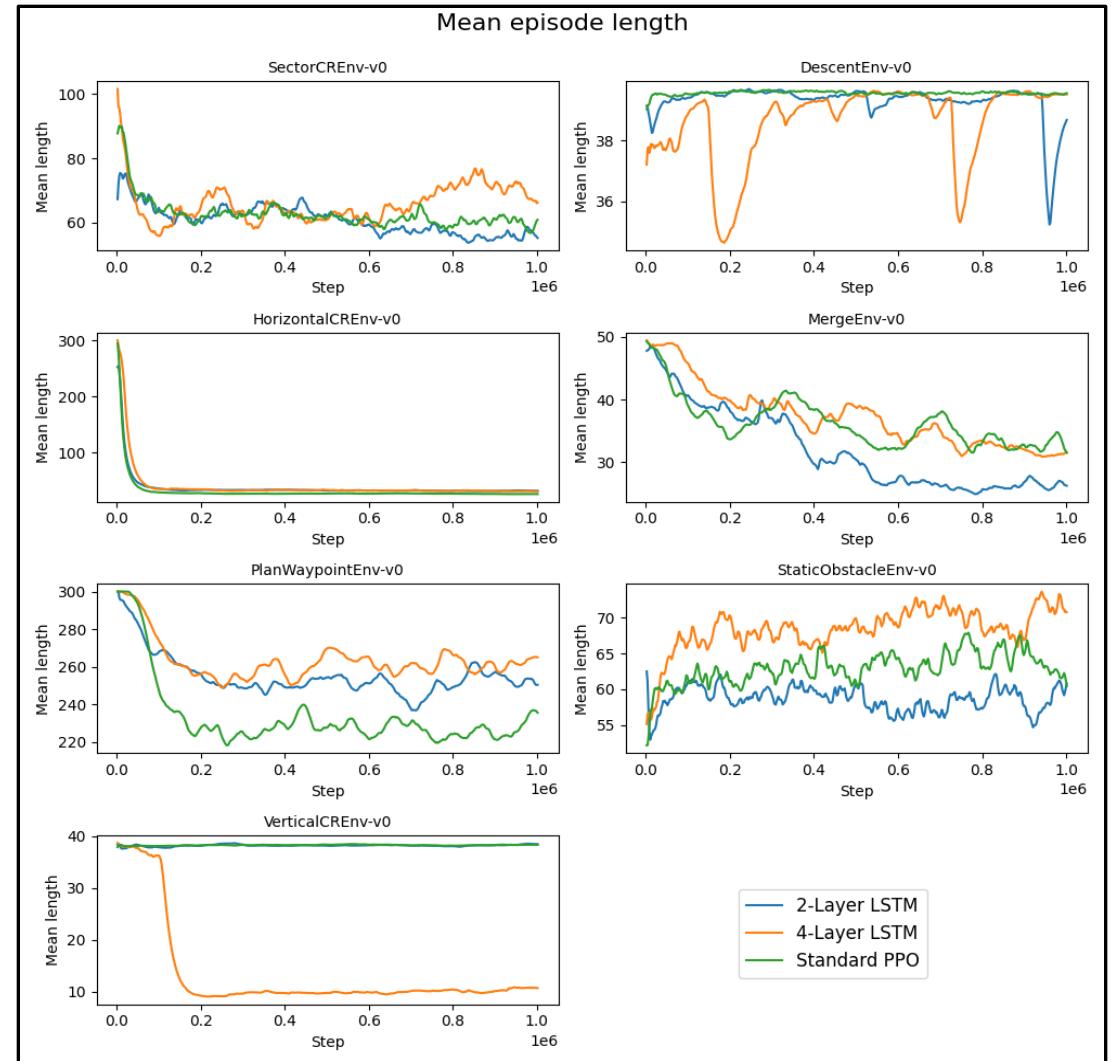
- Using the LSTM component of the RPPO allows the agent to
 - Remember where obstacles were last seen
 - Avoiding re-checking already explored regions
 - Smoother trajectories around unseen hazards

Results

Plots of average reward per episode



Plots of average episode length



Interpretation of Results

- Standard PPO outperformed RecurrentPPO variants on 5 out of 7 environments tested (SectorCR, Descent, HorizontalCR, Merge, and PlanWaypoint)
- RecurrentPPO (2-layer and 4-layer) excelled on StaticObstacle environment, outperforming Standard PPO
- RecurrentPPO (2-layer) outperformed StandardPPO and RecurrentPPO (4-layer) on VerticalCR environment
- LSTM memory helps when spatial/temporal reasoning is required in cases with obstacles, otherwise it introduces overhead with no benefit