# ACAS Xu Policies via Deep Reinforcement Learning

Kyle Julian

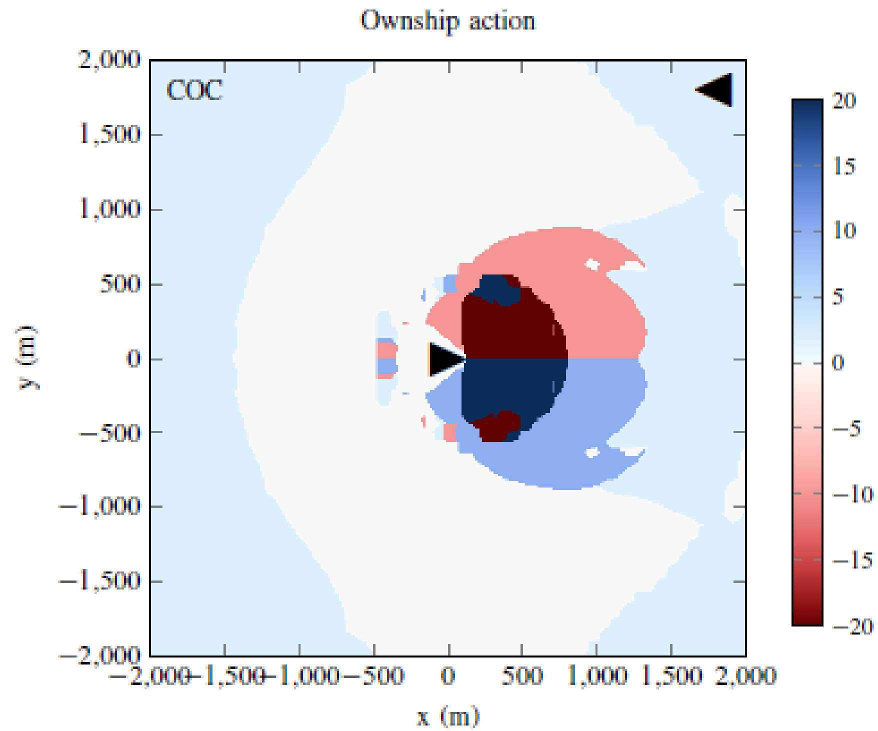
March 9, 2015

# Overview

- Prior Work
- Problem Task and Goals
- Approach
- Results
- Next Steps

# Prior Work

- Airborne Collision Avoidance System X for UAS (ACAS Xu)
- Formulated as POMDP
- Pairwise encounters
- Solved with QMDP




Stanford University

# Project Motivation

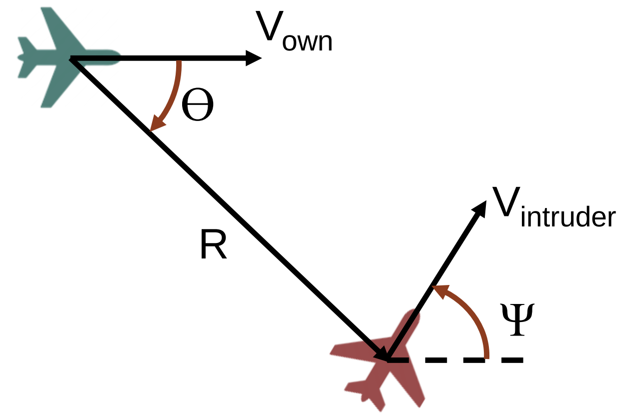
- Q table has 14 million state-action pairs
  - Future tables have up to 600 million Q values
  - Requires large space in memory
- Large policies take a full day to compute
  - Too long to study effect of reward structure on policies

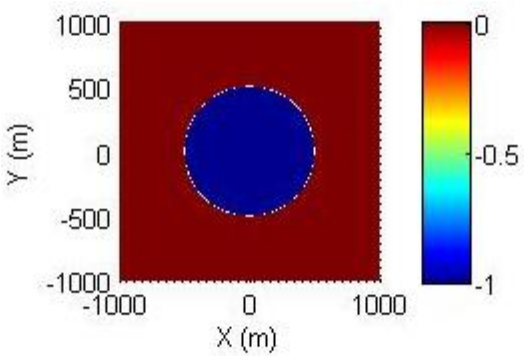
# Project Goals

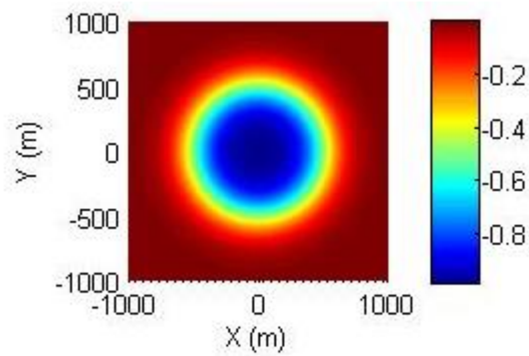
- Solve problem through deep reinforcement algorithm
- Compress representation of Q values
  - Estimated through network parameters
- Compute network quickly

# Problem Formulation

- State:
  - R: [0,3000m]
  - Θ, Ψ : [-π, π]
  - $V_{own}$, $V_{intruder}$ : $[10\,\frac{m}{s}, 20\,\frac{m}{s}]$
- Actions (bank angles $\phi$):
  - {-20°, -10°, 0°, 10°, 20°, COC}
- Rewards, four possible penalties:
  - $-\frac{1}{1.0+e^{-(r_{min}-r)*C}}$
  - $-0.0002 * \phi^2$
  - $-0.03$ if action $\neq$ COC



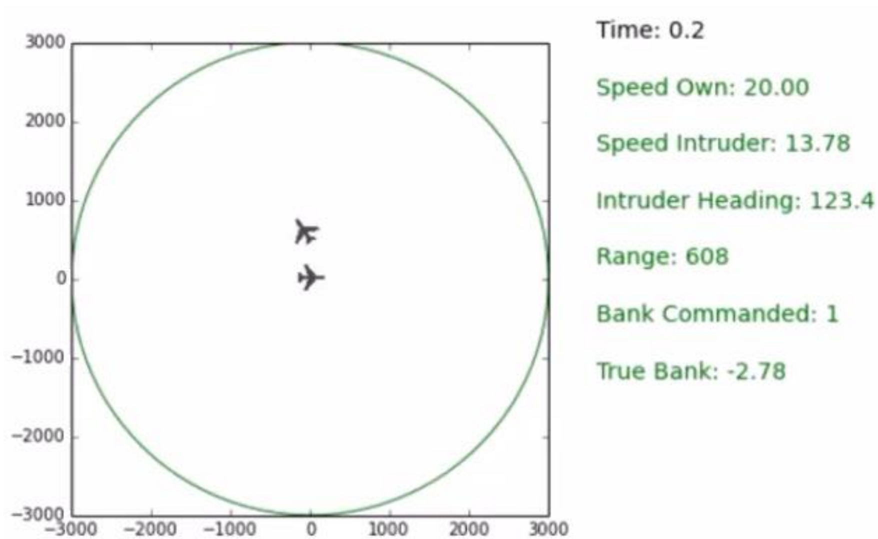






Stanford University

# Dynamics

- Bank angle and speeds are noisy measurements
- True values are drawn from normal distributions:
  - $\phi = \begin{cases} N(action, \sigma = 4°), & action \neq COC \\ N(0°, \sigma = 10°), & action = COC \end{cases}$
  - $V = N\left(V_{meas}, \sigma = 2\frac{m}{s}\right)$
- $x_{i+1} = r_i \cos(\theta) + (v_{intruder} \cos(\psi) - v_{own}) * dt$
- $y_{i+1} = r_i \sin(\theta) + v_{intruder} \sin(\psi) * dt$
- $\Delta\psi = -g * \frac{\tan(\phi)}{v_{own}} * dt$
- $r_{i+1} = \|[x_{i+1}, y_{i+1}]\|_2$
- $\theta_{i+1} = atan2(y_{i+1}, x_{i+1}) + \Delta\psi$
- $\psi_{i+1} = \psi_i + \Delta\psi$



# Techniques Applied to DQNs

- Stochastic transitions
  - Sampled K rewards and next states
  - Used average rewards and Q values
- Smoother Rewards
- Stored multiple (S,A,R,S’) tuples between network updates
  - Larger batch sizes

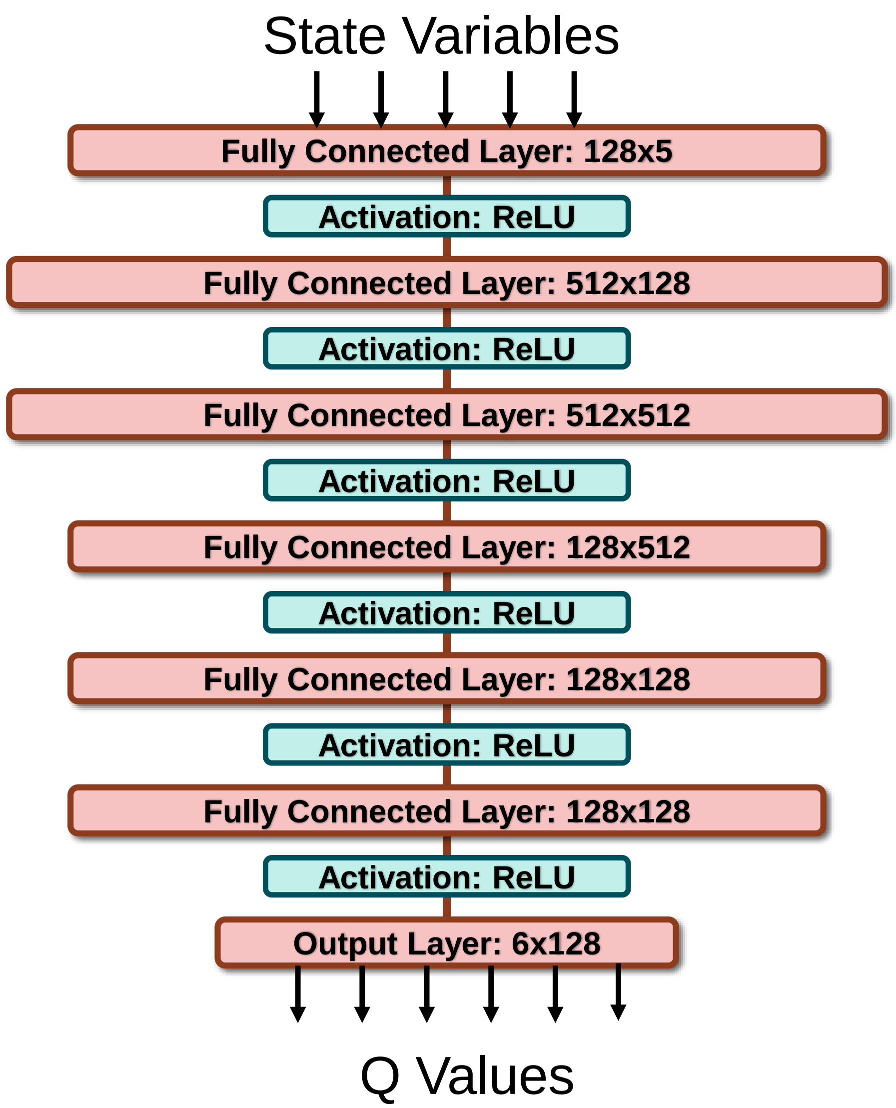
# Pseudocode

**Algorithm 1** ACAS Xu DQN Algorithm

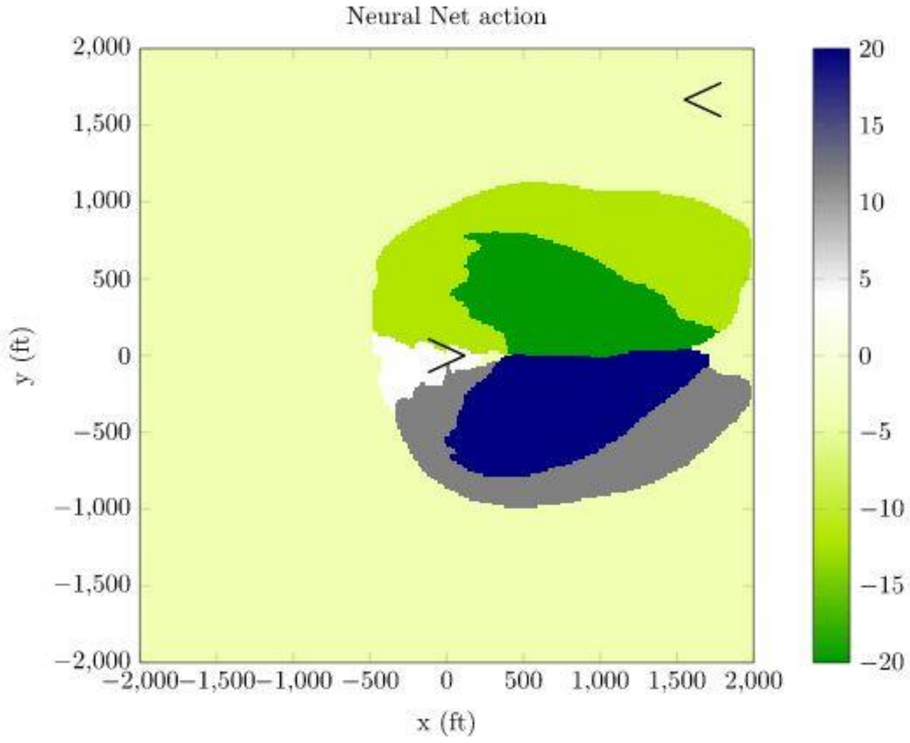
```
initialize DQN
initialize replayMem
initialize stateGen
repeat
    for i ← 1, K do
        action ← DQN.getAction(state)
        nextStates, rewards ← stateGen.nextState(state, action)
        replayMem.store(state, action, rewards, nextStates)
        state ← nextStates[0]
        if state.Range > 3000 then
            state ← stateGen.randomState()
    if repMem.canTrain() then
        batch ← repMem.sampleBatch()
        DQN.train(batch)
until converged
```

## DQN Parameters

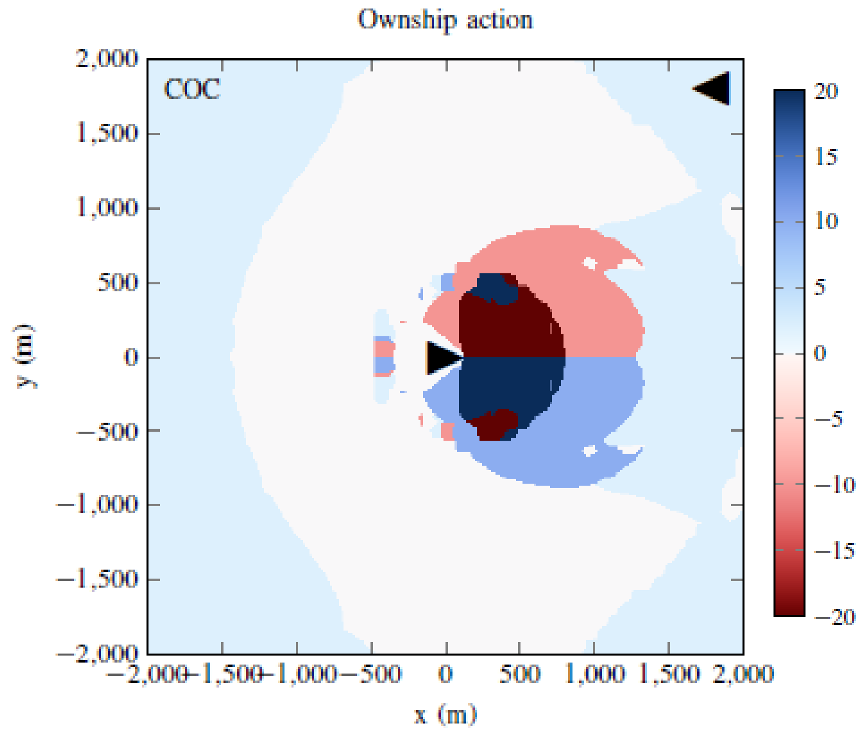
- D_Initial_Size = 200,000
- D_Max_Size = 1.5M
- Final_Epsilon = 0.1
- Final_Exploration = 7.5M
- Train_Freq = 20
- Batch_Size = 512
- Target_Freq = 15000
- Num_Traj = 16
- Gamma = 0.97
- dt = 5
- Network Library = Keras
- Solver = AdaMax
- Loss = MSE



# Results

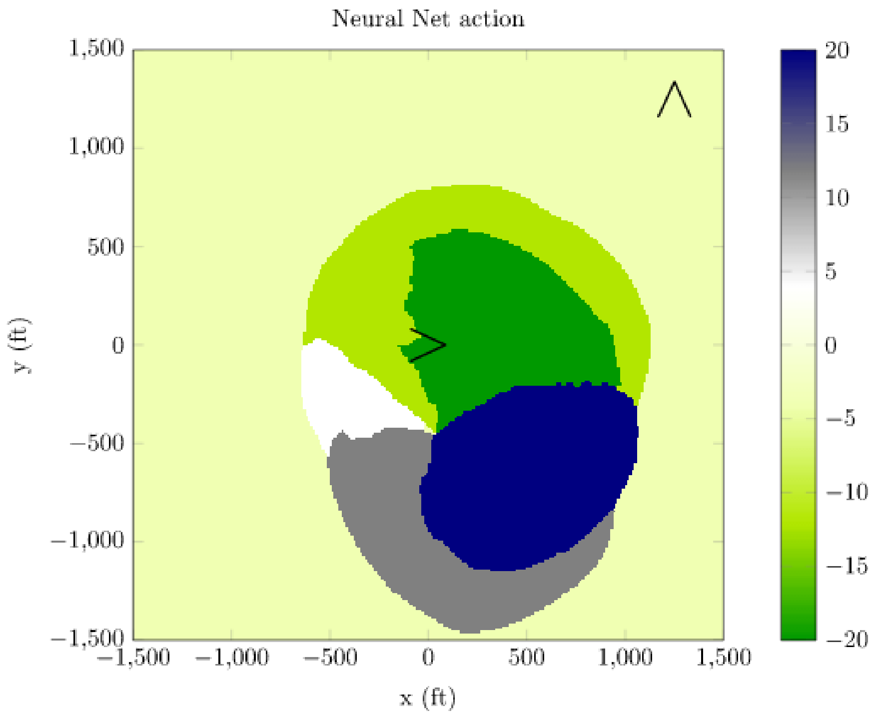


Neural Net action
2,000
1,500
1,000
500
0
−500
−1,000
−1,500
−2,000
y (ft)
−2,000 −1,500 −1,000 −500 0 500 1,000 1,500 2,000
x (ft)
20
15
10
5
0
−5
−10
−15
−20

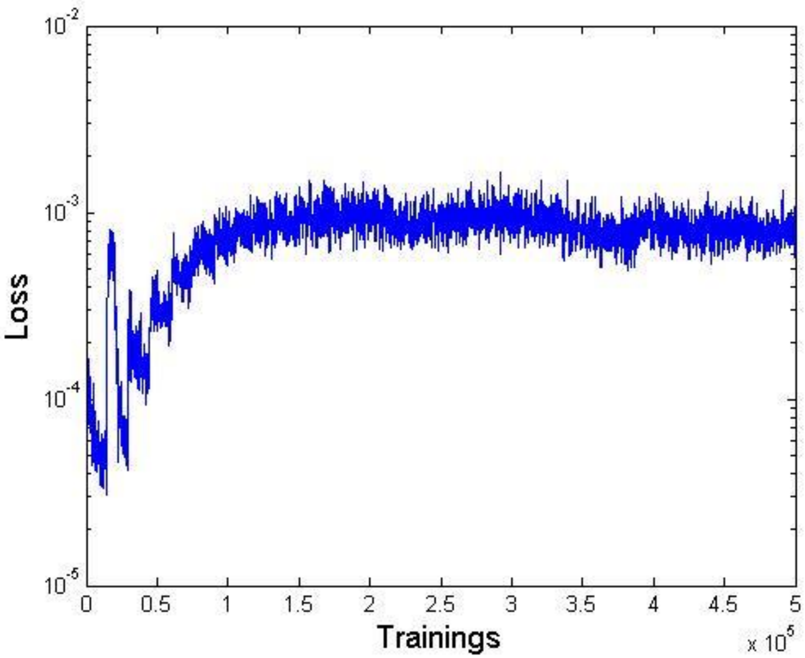



Ownship action
COC
2,000
1,500
1,000
500
0
−500
−1,000
−1,500
−2,000
y (m)
−2,000 −1,500 −1,000 −500 0 500 1,000 1,500 2,000
x (m)
20
15
10
5
0
−5
−10
−15
−20

# Results



Neural Net action
1,500
1,000
500
0
−500
−1,000
−1,500
y (ft)
−1,500
−1,000
−500
0
500
1,000
1,500
x (ft)
20
15
10
5
0
−5
−10
−15
−20

$10^{-2}$
$10^{-3}$
$10^{-4}$
$10^{-5}$
Loss
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Trainings
$x\ 10^{5}$

# Future Work

- Prioritized experience replay
- Optimize speed of algorithm
- Use learned parameters with different rewards
- Expand state space
- Simulate policy

# References


Hao Yi Ong and M. J. Kochenderfer, "Short-term conflict resolution for unmanned aircraft traffic management," *Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th*, Prague, 2015, pp. 5A4-1-5A4-13.

Mnih, V. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

Sascha Lange, Thomas Gabel, and Martin Riedmiller. “Batch Reinforcement Learning”. M. Wiering, M. Van Otterlo (Eds.). Reinforcement Learning: State of the Art. Springer-Verlag (2012). pp. 45-70

Diderik Kingma and Jimmey Lei Ba. “Adam: A Method for Stochastic Optimization”. *International Conference on Learning Representations.* San Diego, 2015.

M.J. Kochenderfer and J.P. Chryssanthacopoulos, “Robust Airborne Collision Avoidance through Dynamic Programming,” MIT Lincoln Laboratory, Project Report ATC-371, 2011.

Blake Wulfe. Hierarchical Reinforcement Learning Algorithms. https://github.com/wulfebw/hierarchical_rl

Thanks!
Questions?