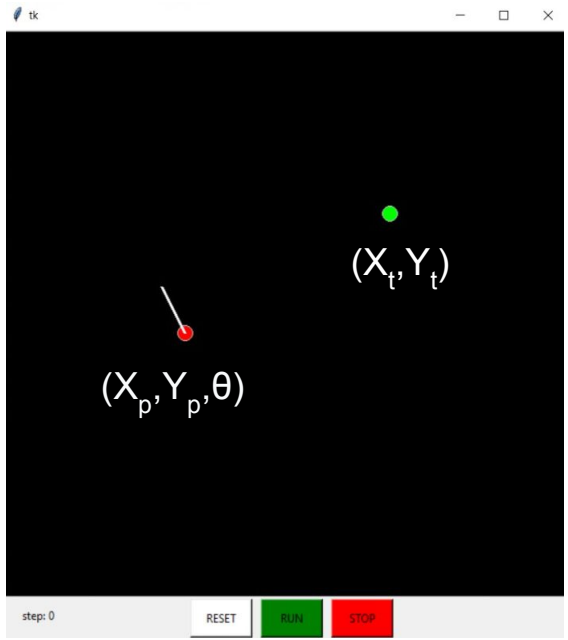
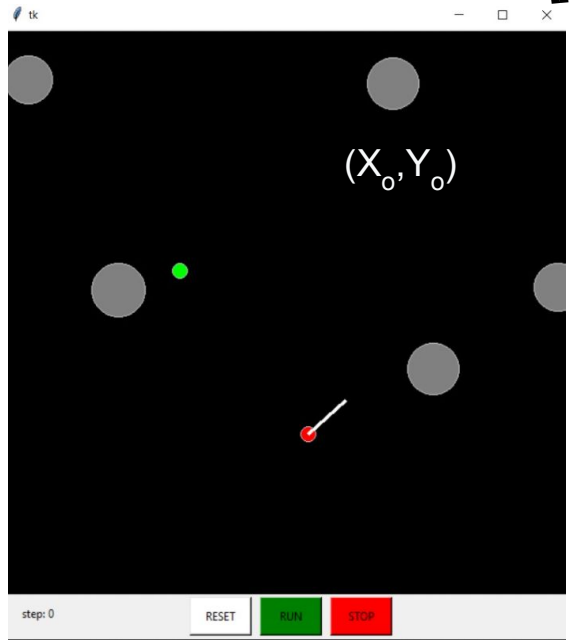
# Design Credit Project:

## UAV Control for LiDAR with RL

# Problem Statement



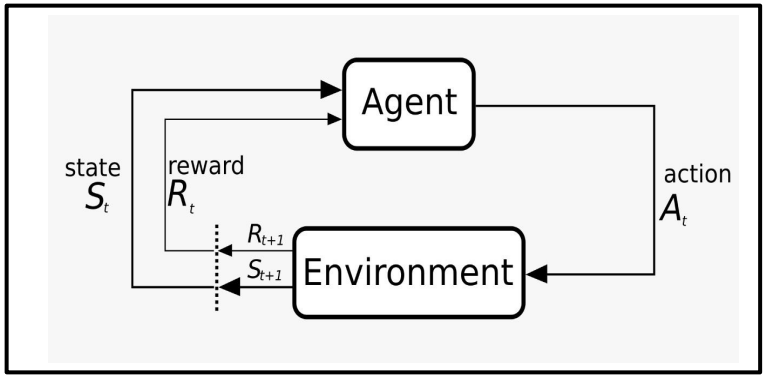




We implemented LiDAR and RL to the classic pursuit problem. The goal here is to reach the stationary target while avoiding all the obstacles and throughout this process the pursuer moves with constant velocity.

# 01

# RL & DQN

# Reinforcement learning



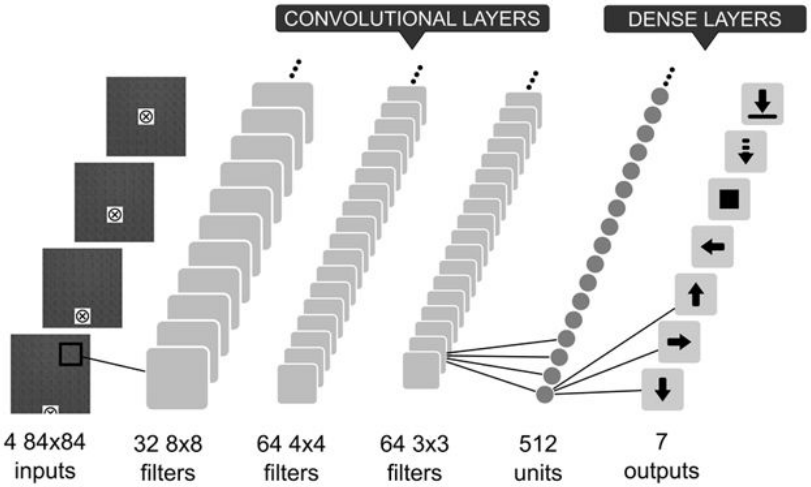

## Theory

This is a model free, off-policy Reinforcement learning algorithm

$$Q^{\pi}(s,a) = \sum_{s' \in \mathcal{S}} P(s'|s,a)\left[R(s,a,s') + \gamma V^{\pi}(s')\right] \quad [1]$$

$$Q_*(s,a) = \max_{\pi} Q_{\pi}(s,a) \quad [2]$$

# DQN

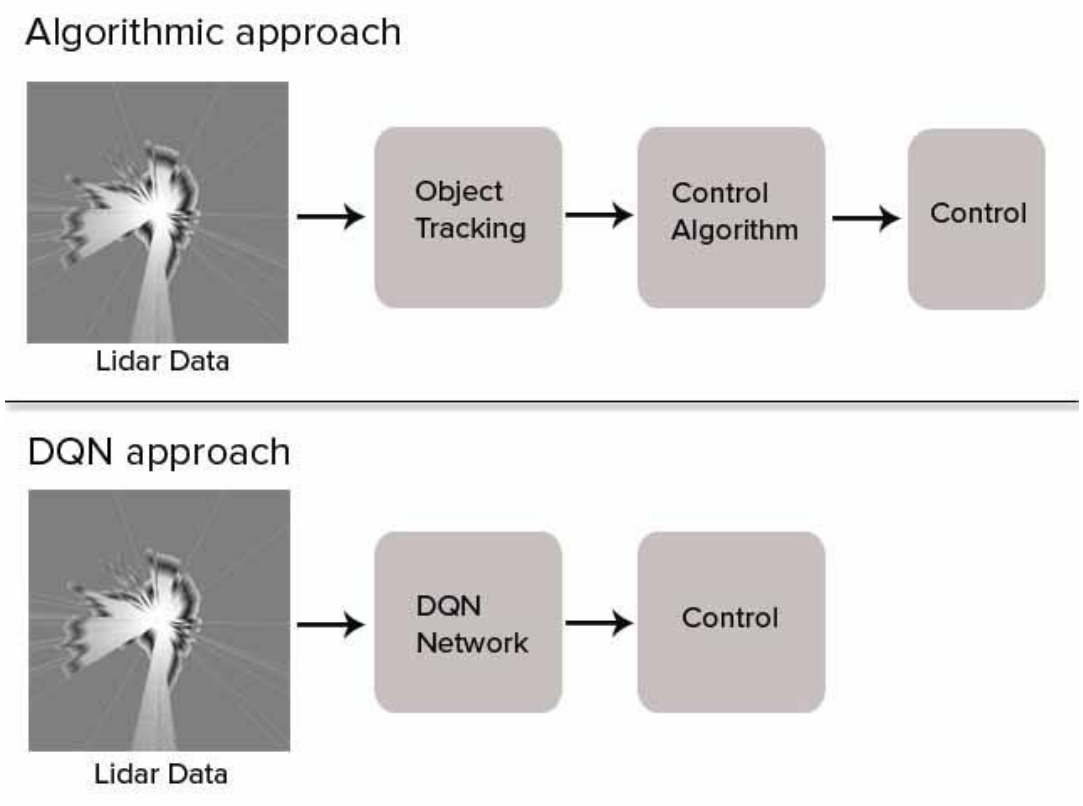



## Working -

The weights (Q values) are updated with every step through Q update equation -

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

# Why Deep Q Networks?



# 02

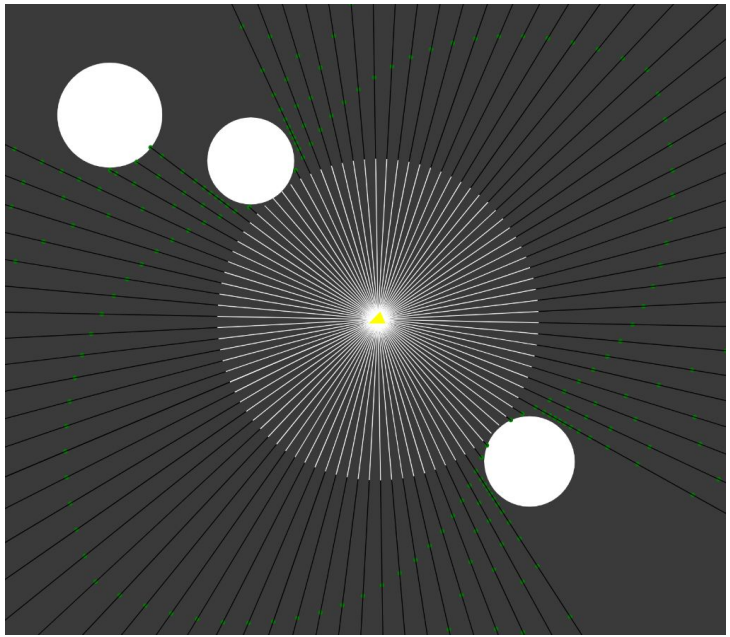
# LiDAR

# LiDAR

## Theory

LiDAR measures the return time of laser after its reflection to determine the difference in distances of various objects.

## Our LiDAR model -



# 03

# Training & Results

# Training Method

## Episode Batching

- Initialize a Batch of random environments.
- Predict and fit every step
- Utilises parallel computation on GPU

```
5%|4         | 920/20000 [02:05<46:08,  6.89step/s]
5%|4         | 941/20000 [02:08<41:04,  7.73step/s]
5%|4         | 945/20000 [02:08<42:04,  7.55step/s]
5%|5         | 1009/20000 [02:17<40:34,  7.80step/s
5%|5         | 1015/20000 [02:17<43:19,  7.30step/s
6%|6         | 1203/20000 [02:43<42:00,  7.46step/s
6%|6         | 1246/20000 [02:48<41:54,  7.46step/s
```

## Naive step training

- Only initialize a single random environment
- Predict every step, fit after sufficient data is learned
- Complex calculations that run on the CPU

```
5%|5         | 1095/20000 [1:40:30<24:32:52,  4.67s/episodes]
5%|5         | 1099/20000 [1:40:48<23:24:50,  4.46s/episodes]
6%|5         | 1109/20000 [1:41:45<23:41:25,  4.51s/episodes]
6%|5         | 1143/20000 [1:45:23<30:31:27,  5.83s/episodes]
6%|5         | 1163/20000 [1:47:16<25:31:40,  4.88s/episodes]
6%|6         | 1242/20000 [1:55:41<28:45:27,  5.52s/episodes]
6%|6         | 1279/20000 [1:59:24<30:04:12,  5.78s/episodes]
```

# Various Trained Models-

Three model based on different observations given to the pursuer -

## Model .a

Instantaneous positions

## Model .b

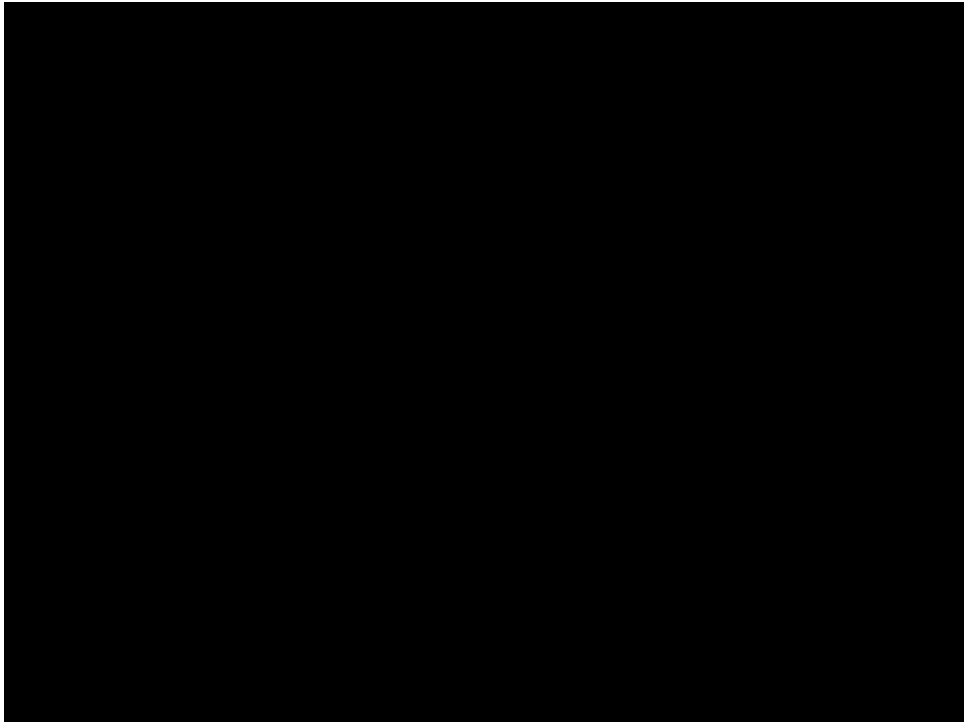
Instantaneous positions with pursuer’s velocity vector

## Model .c

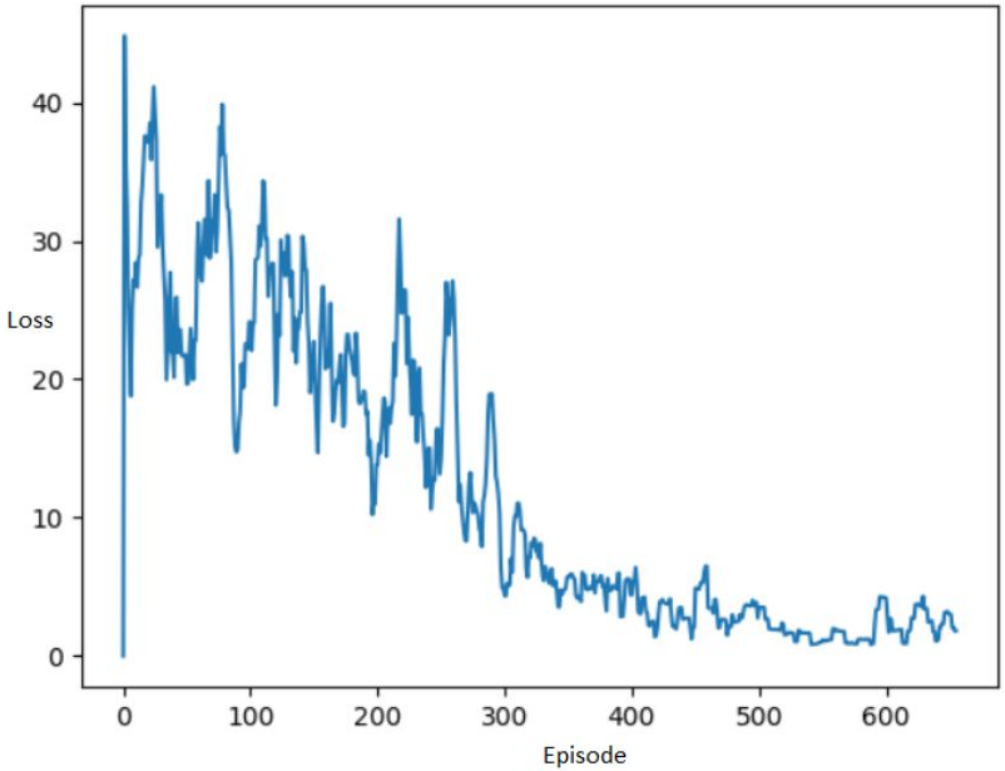
Relative positions with velocity vector

# Model 1

# Model 2



# Loss vs No. of episodes



# Thanks for Listening!

Presented by :- Aditya Gadhavi(B19EE004)
Ajay Kumar (B19EE005)