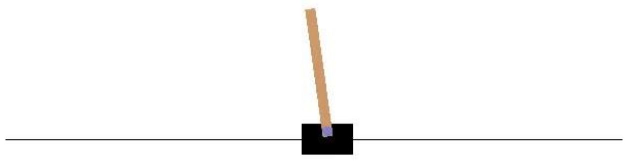
Environments

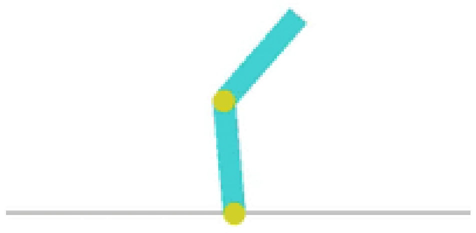
- Cart-Pole

  A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum starts upright, and the goal is to prevent it from falling over by increasing and reducing the cart's velocity. The **possible actions** are pushing the cart to the left or the right. The **reward** is 1 for every step taken. The episode is **terminated** when the **pole angle** is more than **12 degrees positive or negative** or when the **cart position** is more than **2.4**. Or when the **episode length is more than 200.**



- Acrobot

  The acrobot system includes two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height. The **goal** is to swing the end-effector at a height at least the length of one link above the base. Both links can swing freely and can pass by each other, i.e., they don't collide when they have the same angle. The **state** consists of the sin () and cos () of the two rotational joint angles and the joint angular velocities : [cos(theta1) sin(theta1) cos(theta2) sin(theta2) thetaDot1 thetaDot2]. For the first link, an angle of 0 corresponds to the link pointing downwards. The angle of the second link is relative to the angle of the first link. An angle of 0 corresponds to having the same angle between the two links. A state of [1, 0, 1, 0, ..., ...] means that both links point downwards.   The **action** is either applying +1, 0 or -1 torque on the joint between the two-pendulum links.

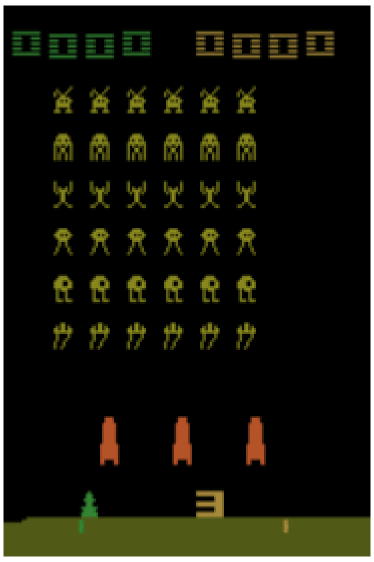


- Pendulum

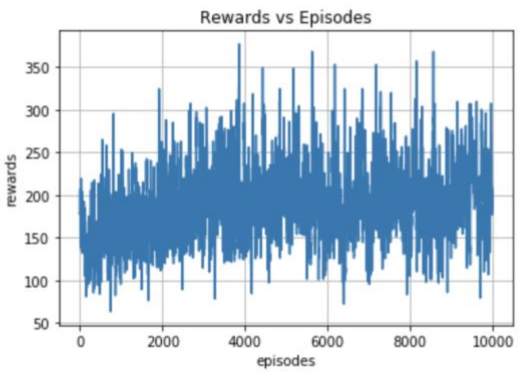
The inverted pendulum swings up problem is a classic problem in the control literature. In this version of the problem, the pendulum starts in a random position, and the goal is to swing it up, so it stays upright. The **action space** is the torque applied on the pendulum which is in the range (-2,2). The state space is the pendulum angle and the pendulum speed. The **default reward** function depends on the angle of the pendulum. If the pendulum is **upright**, it will give **maximum rewards**. We do not need to change the default reward function here. The **starting state** is a random angle from -pi to pi, and random velocity between -1 and 1.

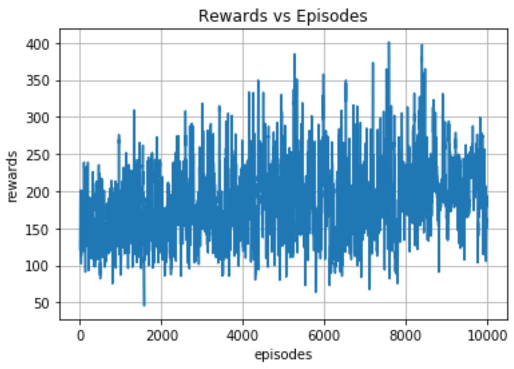
- Space Invaders

This is a one of the classic Atari games. In this game we have a lot of aliens that must be killed by the player. The aliens can attack the player by shooting at the player. There are also these blocks that can be used by the players as cover to hide from the shots of the aliens, which are destructible by the shots fired and can be destroyed completely as the game progresses. As the game progresses the aliens also move closer to the player. Player gets 3 lives and has 6 possible actions to be taken at any given point. These actions are NOOP (no operation), FIRE (shoot without moving), RIGHT (2, move right), LEFT (3, move left), RIGHTFIRE (4, shoot and move right), LEFTFIRE (5, shoot and move left). Player is rewarded when aliens are killed.



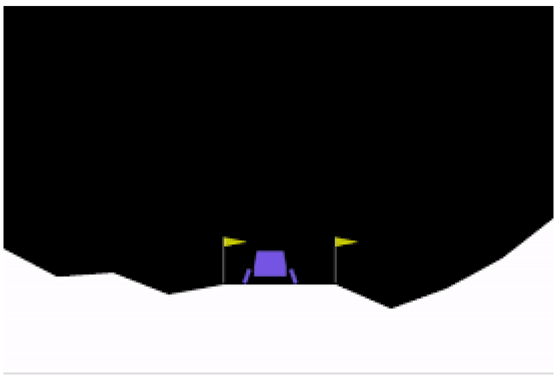
Based on current runs we got below rewards for DQN and Double DQN algorithms.





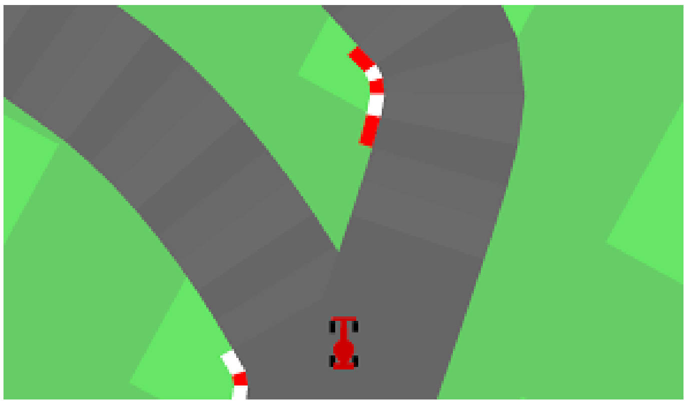
- Lunar Landing

Landing pad is always at coordinates (0,0). Coordinates are the first two numbers in state vector. **Reward** for moving from the top of the screen to landing pad and zero speed is about 100 to 140 points. If lander moves away from landing pad it loses reward back. **Episode finishes if the lander crashes** or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. **Solved is 200 points**. Landing outside landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt. **Four discrete actions** available: do nothing, fire left orientation engine, fire main engine, fire right orientation engine



- Car Racing

Easiest continuous control task to learn from pixels, a top-down racing environment. Discreet control is reasonable in this environment as well, on/off discretisation is fine. **State** consists of 96x96 pixels. **Reward** is -0.1 every frame and +1000/N for every track tile visited, where N is the total number of tiles in track. For example, if you have finished in 732 frames, your reward is 1000 - 0.1*732 = 926.8 points. **Episode finishes when all tiles are visited**. Some indicators shown at the bottom of the window and the state RGB buffer. From left to right: true speed, four ABS sensors, steering wheel position, gyroscope.



Plan

- We are planning to first improve our model design by experimenting with layers involved in the dqn and then trying to modify the algorithm so that it can accommodate for and train on any environment that it was initiated on. We will also be trying out other algorithms to compare the performances between these different algorithms.