# CSE474/574: Introduction to Machine Learning (Fall 2018)

## Project 4: Tom and Jerry in Reinforcement learning

**Kishan Dhamotharan**
**Person# 50287619**

**Bonus Task Results :**

### Non Atari Environment

Environment :
**Name:** LunarLander-v2 (Continuous control tasks in the Box2D simulator.)
**Environment Description:**
**Rewards:**

- 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.

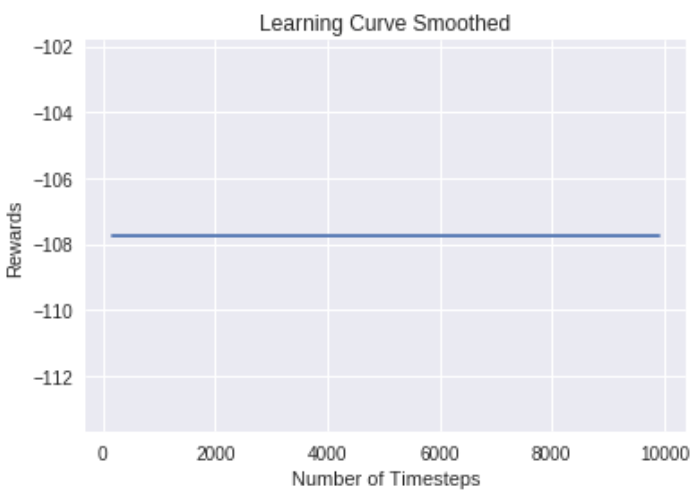
**Actions available:**

1. Do nothing
2. Fire left orientation engine
3. Fire main engine
4. Fire right orientation engine.

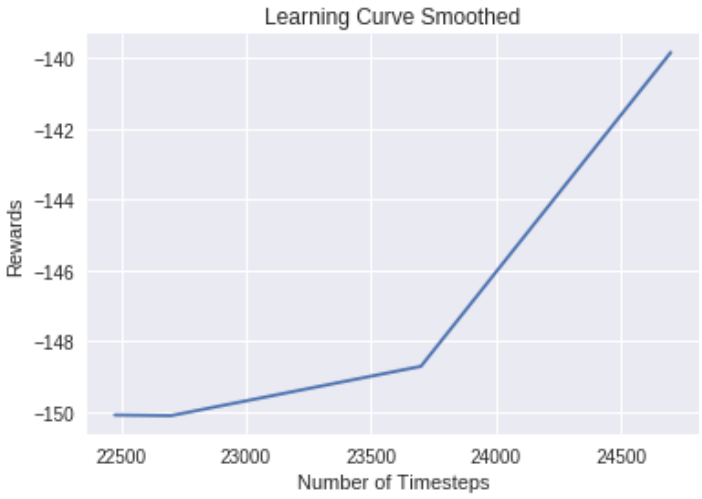
All the understandings have been **commented in the code.**

Following are the graphs, when I tries to vary the model learning from 10000, 25000 and 50000 Timesteps. There was significant improvement with higher values of timesteps, as the system had more to learn. It improved lot better in much faster time. The animation helped me to visually see the changes how the model is growing better each time.

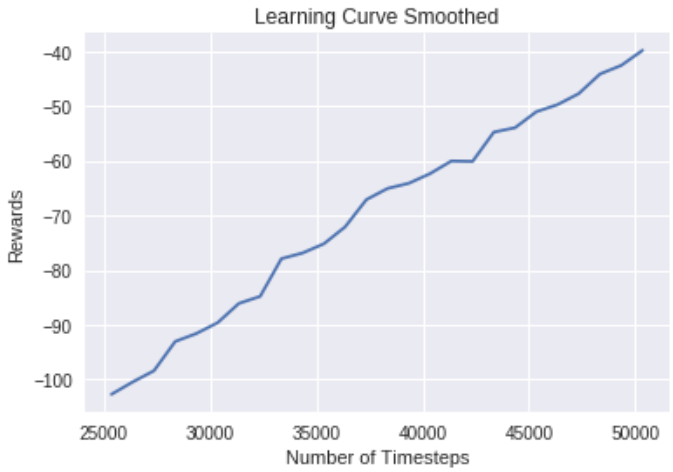
**10,000 timesteps**



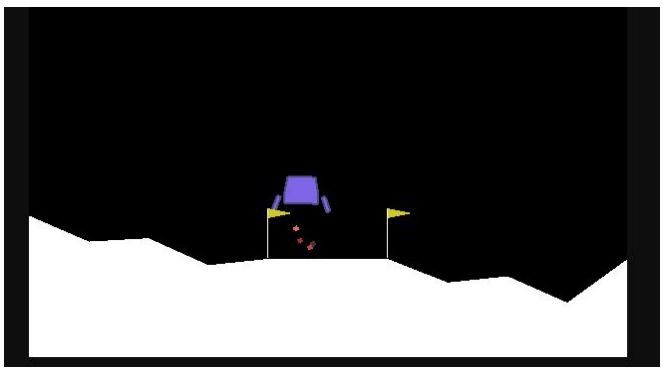

**25,000 timesteps**

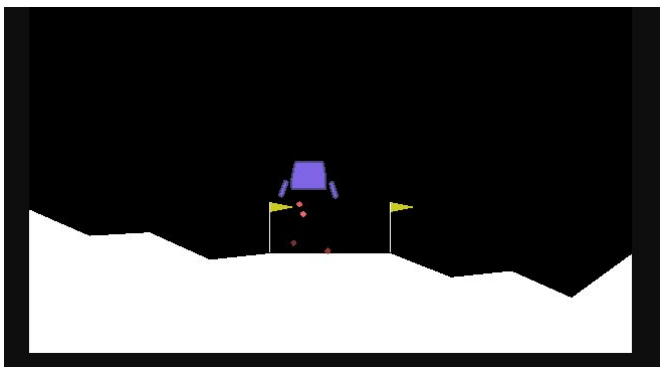



**50,000 timesteps**




Results of  Lunar Lander when the system was learned using 25k timesteps, was able to land very quickly at zero speed between the flags.  With 10K timesteps, it took longer to make the landings.





**Atari Environment**

**Environment :**

**Name:** Atlantis-ram-v0 (**Atari**)

**Environment Description:**

In this environment, the observation is the RAM of the Atari machine, consisting of (only!) 128 bytes. Each action is repeatedly performed for a duration of k frames, where k is uniformly sampled from {2, 3, 4}.

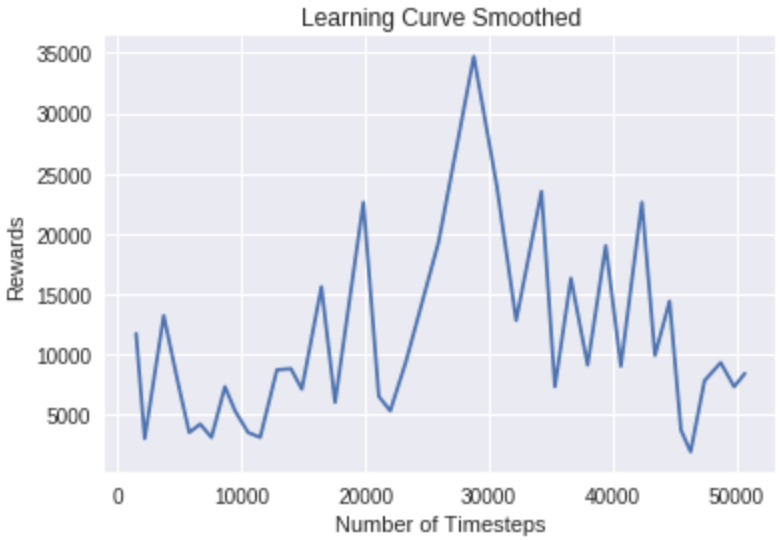
**Actions available:** fire left , fire main, fire right

Runs much faster than the image version. Here we can observe that there is good amount of learning happening in the system, though there is a huge amount of noise. It is a very good sign that the system is trying out different actions and will eventually learn from the experience. Similarly if this continues for a million episode, we can see much much better results.

All the understandings have been **commented in the code.**

**10000 timesteps**


**50000 timesteps**



**Environment :**
**Name:** Atlantis-v0 (**Atari**)
**Environment Description:**
In this environment, the observation is an RGB image of the screen,  which is an array of shape (210, 160, 3)
Each action is repeatedly performed for a duration of kk frames, where kk is uniformly sampled from {2, 3, 4}.
**Actions available:**
fire left , fire main, fire right
As the below graph we can tell that some kind of learning is happening through the system, but with just 25k episodes, they do not perform that well. They will require episodes around millions for them to perform well.
The image version of the Atari is highly computationally intensive, takes around 5 minutes per 1000 timesteps.



**Comments on the inbuilt DQN by stable baseline**
The inbuilt DQN is very easy to use as almost everything is abstract, the possible parameters such as gamma, learning_rate, batch_size, buffer_size, env can be just passed on to the DQN function, few of these parameter are for the neural network. Also we pass another parameter called the policy to the DQN , mlp and cnn

policies, basically these is to tell if it has to implement DNN or CNN. Also these neural networks have variants with combination of different number of layers. This returns a model, which we can use to predict. The implementation must be very good , by looking at the results that we have extracted.