Deep Reinforcement Learning Homework-2

Author: Yu Huang and Yunwen Zhou

- Part One
  1. Question 1 solution

We can rewrite the update equation (1) and (2) respectively as follows:

$w_{new}^T \phi(s,a) = w_{old}^T \phi(s,a) + \alpha(r + \gamma \max_{a' \in A} w_{old} \phi(s',a') - w_{old}\phi(s,a))$

$w_{new} = w_{old} + \alpha(r + \gamma \max_{a' \in A} w_{old} \phi(s',a') - w_{old}\phi(s,a))\phi(s,a)$

Replace those above 2 equations with the indicator function and keep the gradient step term unchanged, we obtain the following (3) and (4) respectively:

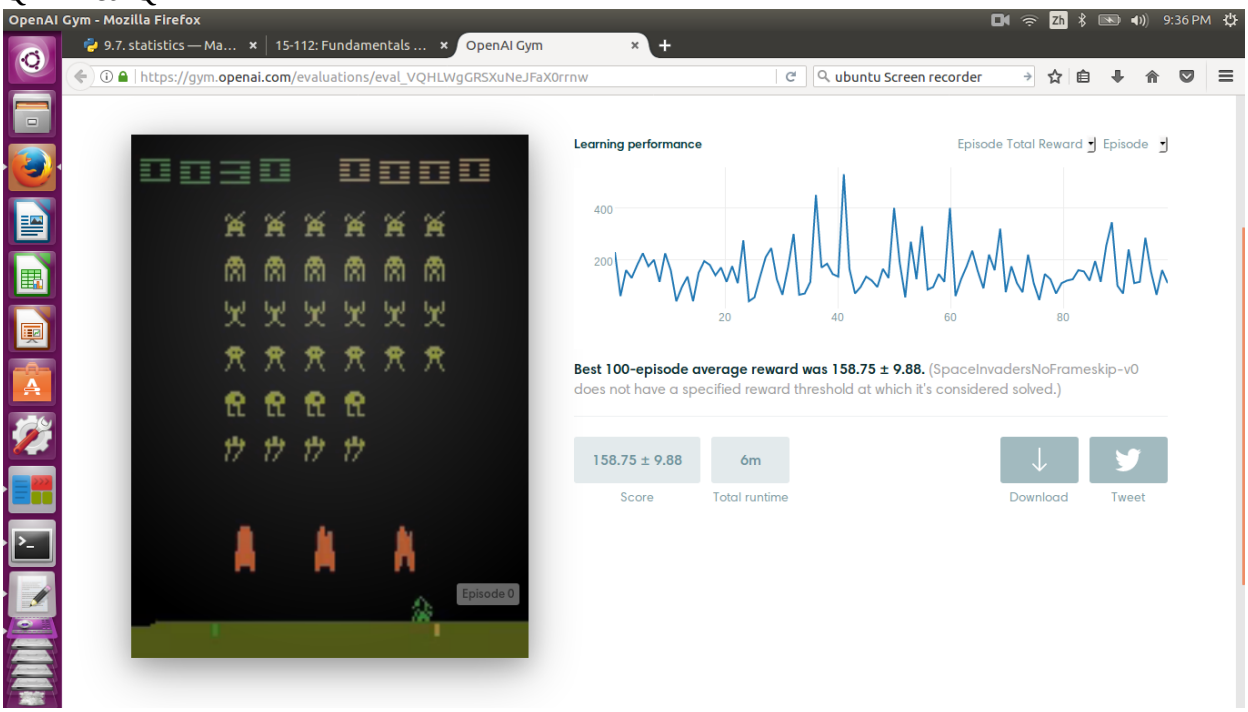
$w_{new}^T \mathbb{I}(s'=s, a'=a) = w_{old}^T \mathbb{I}(s'=s, a'=a) + \alpha(r + \gamma \max_{a' \in A} w_{old} \phi(s',a') - w_{old}\phi(s,a))$ (3)

$w_{new} = w_{old} + \alpha(r + \gamma \max_{a' \in A} w_{old} \phi(s',a') - w_{old}\phi(s,a))\mathbb{I}(s'=s, a'=a)$ (4)

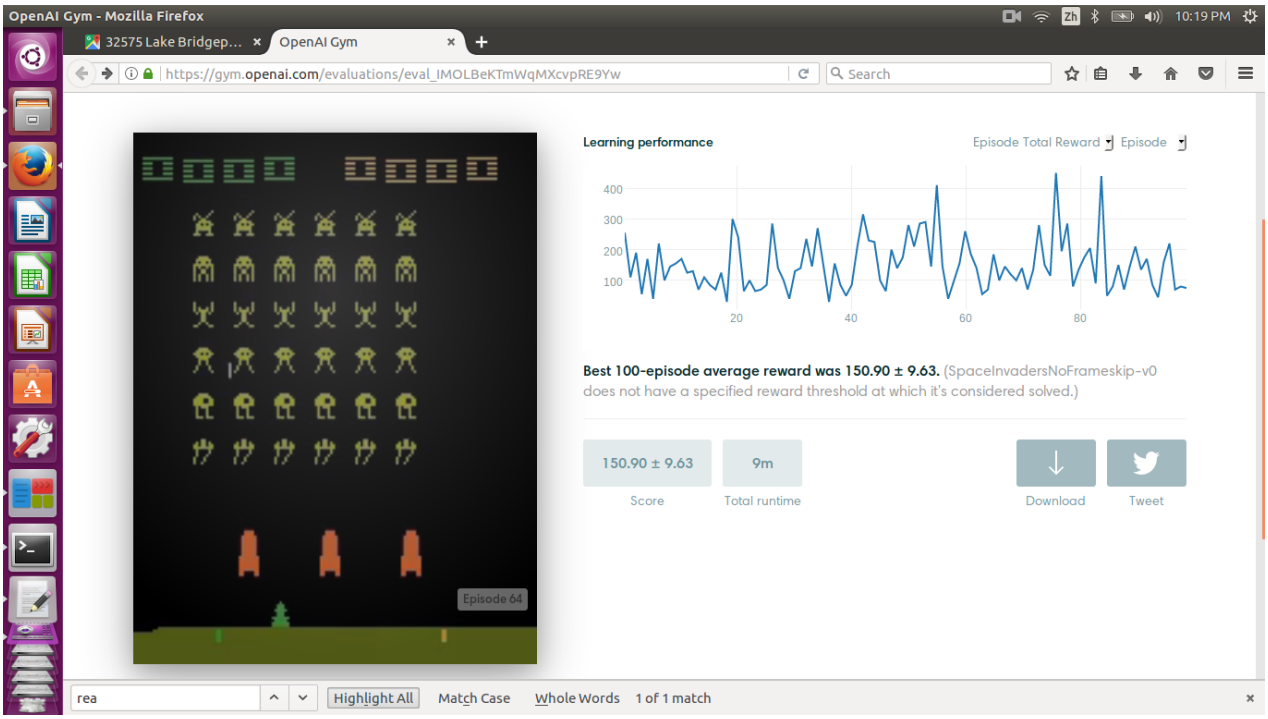
Notice that (3) and (4) are the same because in (3) the $\mathbb{I}(s'=s, a'=a)$ term act as a gating factor that when the condition is satisfied then update the weight while in equation (4) the term $\mathbb{I}(s'=s, a'=a)$ act as a mask that when the condition is satisfied then update the weight otherwise make the new weight the same as the old one. The gradient value in 2 equations are exactly the same.

  2. Performance snapshots

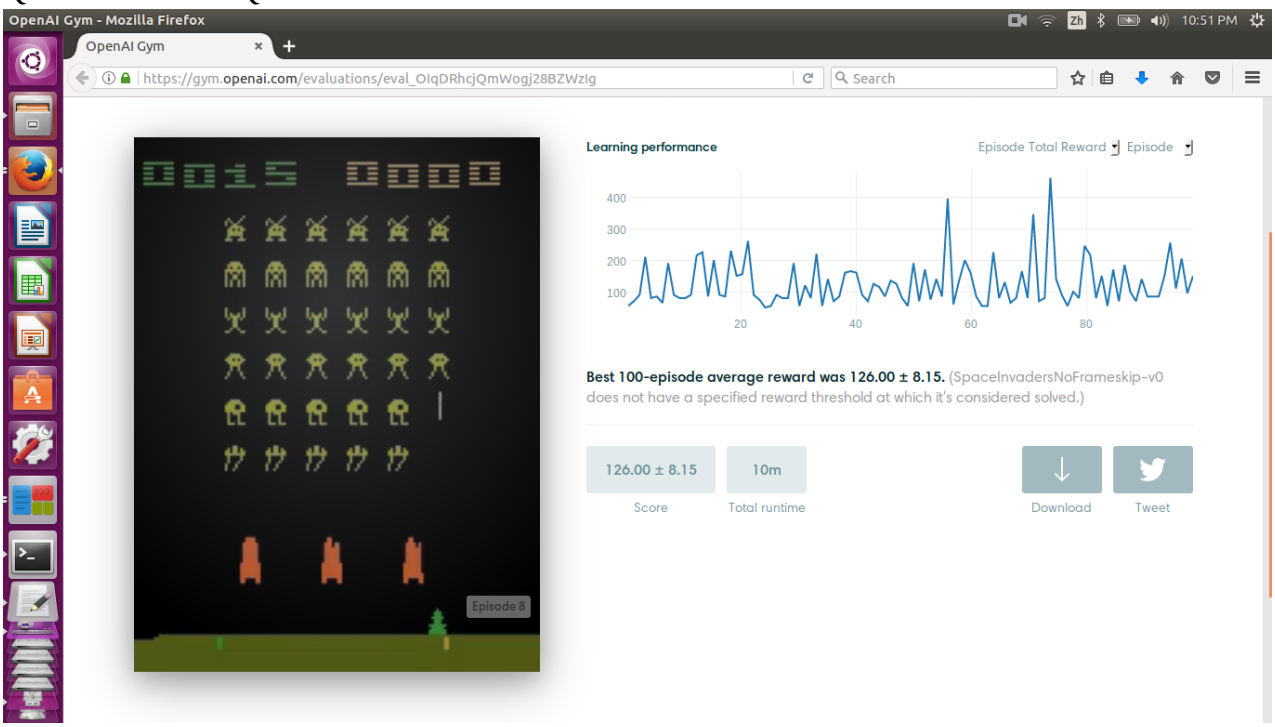
Q2 LinearQ



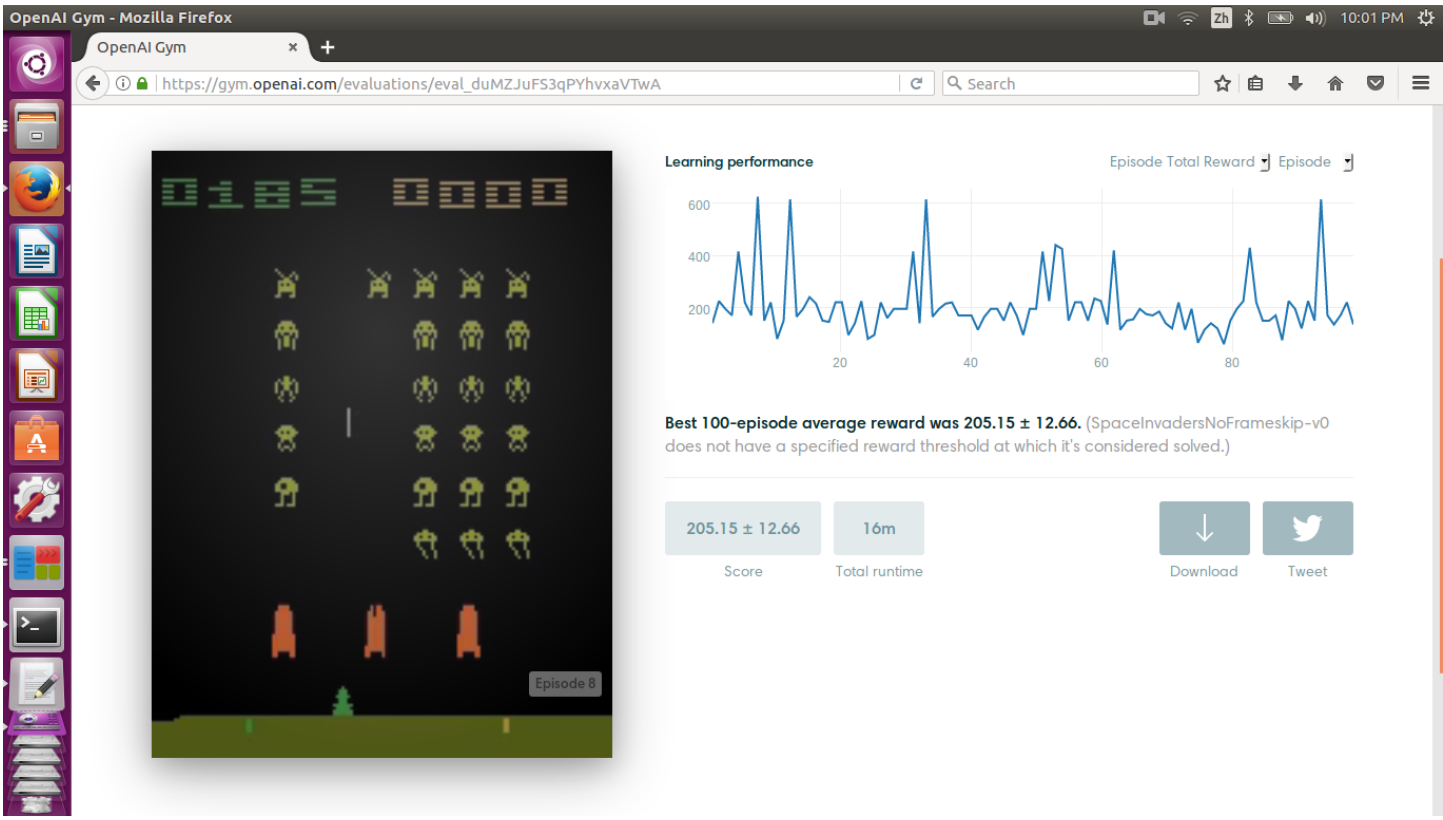
Q3 LinearQ with memory replay and target fixing



Q4 double Linear Q



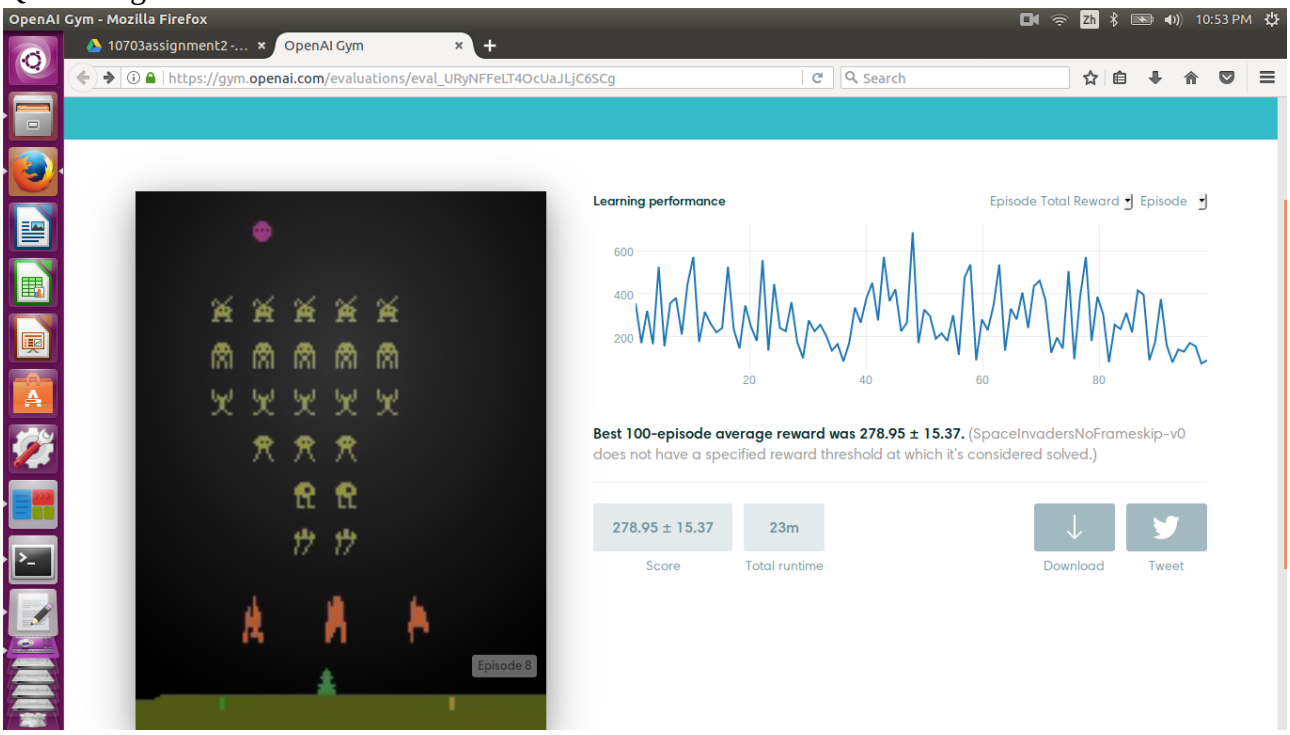
Q5 deep Q nets



Q6 Double Q

Always 0 reward, can`t find the reason.

Q7 dueling nets



3. Question 2 to Question 7 result statistics

Question Number	Training Phase0		Training Phase1		Training Phase2		Training Phase3	
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Q2	17.0	27.98	235.0	0.0	168.75	58.80	158.75	88.96
Q3	345.0	77.37	101.75	140.54	170.75	123.52	150.9	86.69
Q4	168.5	147.80	235.0	0.0	52.0	5.56	126.0	73.33
Q5	180	0	180	0	167.75	94.30	205.15	113.9
Q6	0	0	0	0	0	0	0	0
Q7	105	0	145.0	109.2	112.5	66.32	278.5	138.7

- # Part Two: Coding Part Write-Up

1. Running our code

   For the coding question from 2 to 7, we did not closely follow the code templates provided. Instead, we only use tensorflow to build the weights training files(Q2LinearQ.py Q3LinearQMemRepTarFIx.py Q4DoubleQLinearMemRepTarFIx.py Q5DeepQ.py Q6DoubleQ.py Q7dueling.py). For each of those files listed above, you should be able to run the code(make sure you put the file objective.py and policy.py at the same directory) and obtain those *.out files which are files containing trained weights. Once you obtained those weights, you should be able to run those corresponding recording files(Q2Q3Q4recording.py Q5DeepQrecording.py Q6DoubleQrecording.py and Q7duelingRecoding.py). In those files listed above, we provided more detailed instruction about how to run those files in the comments.

2. Experiment Settings

   Again, in the headers of each weights training files, we have the detailed explanation. Here, we just give a general description about our experiment parameter settings. After going through a long time of extremely painful trials along with the annoying waiting time and other memory restrictions brought by PSC, we decided no to follow strictly those parameters described on those papers, what we choose are as follows:

   Training Rate α : 0.001

   Optimizer: AdamOptimizer

   Exploration Policy: epsilon-greedy policy with linear decay from 1 to 0.05 with decay step 100000

   Total Iteration: 300000

   Training experiment: SpaceInvaders-v0

   Memory Replay Size: 100000

   History Length: 3

   Huber Loss Cut-off delta: 10

   Image Preprocess Method: cv2.cvtcolor and then cv2.resize using nearest neighborhood option

Reward Preprocess: No preprocess (we found it will make agent play better and accelerate training)

Initialization Method: truncated normal with standard deviation 0.1

CNN structure: more complicated exactly as describe in reference 2

Note: for deepQ and double deepQ algorithm we changed the training rate to 0.0005, policy linear decay steps to 200000 because after several rounds of debugging we found that if we use the above parameters, the trained policy always somewhat stick to specific one action and we don't know why. But Question6 still end up with a pretty bad result, we still don't know why.

- Part Three: Summary

We know that our submission seems like a simplified version of what actually Deepmind did. We are confident that if we are allowed more computational resource, we can get a significantly better result. However, the reality is that each time we submit code to PSC, it will wait for at least 12 hours to begin and we are not familiar with PSC, sometimes out jobs just get killed because of the job files and we need to wait at least 12 hours…

The reason we choose the parameters like this is we found that it could take less time to see the convergence and less time to train so that we can submit the homework before last day.

We provide a file which has links to OpenAI gym website containing our agents performance, make sure you open this file using Sublime Text otherwise you will see something funny. If you don't want see them one by one through those links we also provide links to YouTube where we made one video play list containing all the episode note that because of the screen recording app`s problem, somethings the bullets are invisible.

YouTube playlist:

https://www.youtube.com/watch?v=FnJIPnmhfyA&list=PL91rNqpk6aPG952kRH-1JdIui8MrdR9T_