09/11

Code working for Acrobot with height as reward

Result: Not balancing the acrobot

09/13

1. Experiment with the Pendulum environment

Result: Successfully balance the pendulum

2. Modify the acrobat reward function to include the cost of thetadot

Result: Not balancing, swings over constantly

Concern: dt too large

09/20

Add functionality to change available torque and dt based on a factor

Result: Not balancing, swings over, dt changes doesn’t seem to have much effect

Concern: Not enough torque

09/25

Change dt without change torque and start near straight up

Result: no distinctive difference between the results; does well in some initial conditions but fails more often

Concern: not enough torque options, not large enough neural network, not enough torque, not enough training, overfitting

09/27

FIX dt TO 2 FOR FOLLOWING EXPERIMENTS

1. Change neural network to [128, 128]

Result: significant improvement of score and Q value

2. Change stacking of history to one state

Result: significant improvement of score and Q value

3. Larger torque (by 2, by 4)

Result: Not strictly tested

4. More options with larger torque

Result: Not helping if not hurting

Concern: action space size might not be properly changed, MIGHT NEED TO RERUN

5. Rerun 09/25 experiments with longer training (change stopping condition)

DID NOT RERUN BY OCT 1ST

6. Add dropout on the second layer of [32, 32] network

Result: No significant improvement

10/01

Run no stacking of history with large neural networks: [128,128],[256,256],[512,512]

Result: no significant improvement as neural networks get larger

10/08

Run no stacking of history with deeper neural network

Result: deeper neural network does have more stable performance but not significant improvement

Run pendulum with action taken out of reward function

Result: No significant difference but it diverges faster

10/14

Use frames lasted as the reward

Result: slightly more stable

10/30 (256,256) neural network

Wider range of actions

Result: Very unstable

Starting straight up without randomness

Result: steady increase and then fluctuates around 150

11/03

Slower decay of epsilon

Result: Q diverge to arbitrarily large value, performance bad

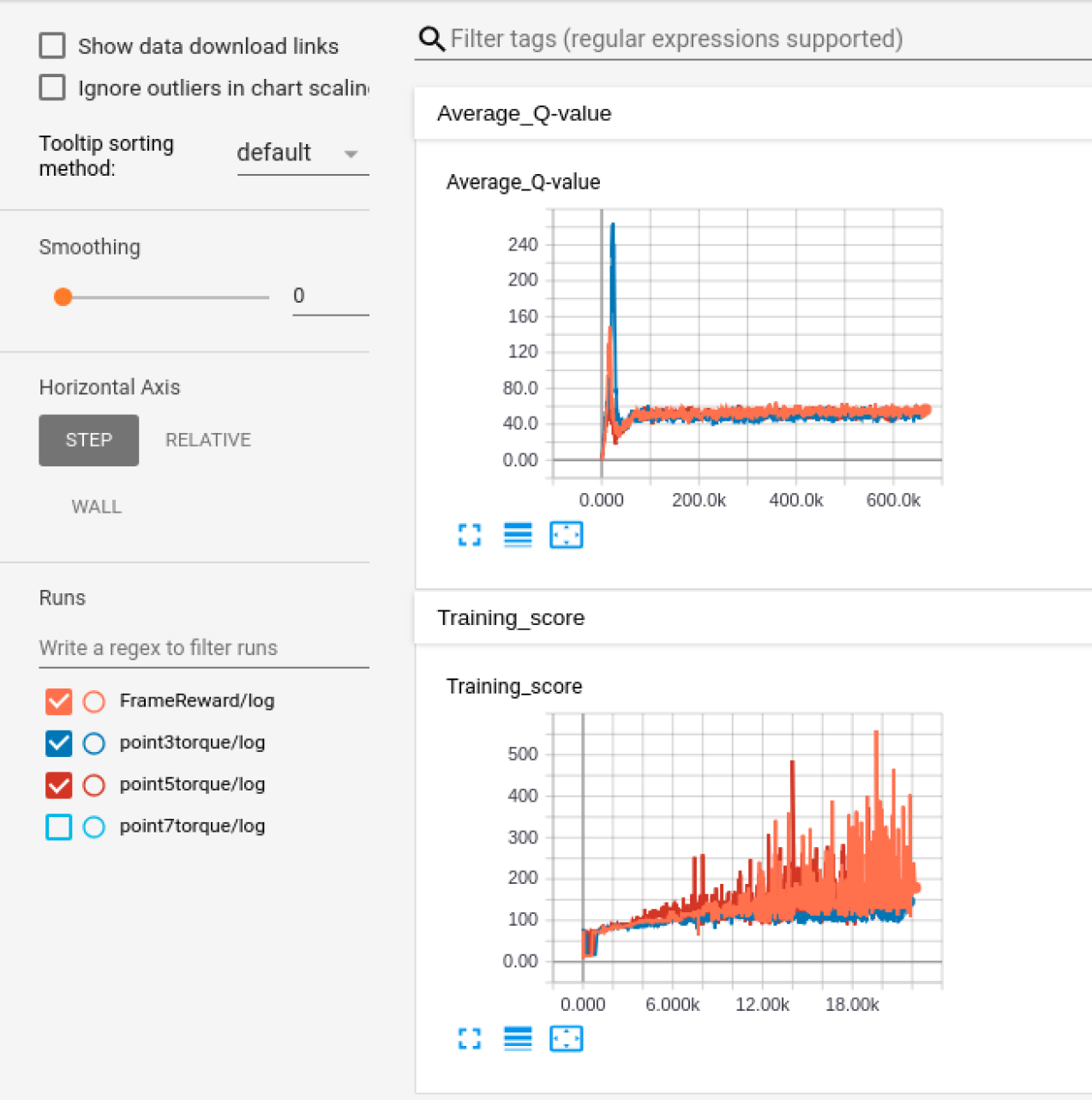
Found: not starting at 0 exactly but machine 0

11/06

Smaller torque:

* 0.7

Result: Diverged

* 0.5
* 0.3
* 

11/11

Smaller torque in magnitude:

Result: Still unstable

11/18

Update tensorboard summary to record loss

Result: the trail results are unstable (1 and 0.7 have different convergence), the loss does blow up when Q value diverges

Tweak: throw away batch if loss too large (>1.5) or increase batch size to balance bad batch

11/19

Implementing Henon map and the method in the paper to observe performance

Result: Method based on paper converges

11/24

Stopping criteria for Henon map:

* Absolute change from the current state to last state:
  + Start from being able to find the fixed point sometimes to finding the wrong fixed point
* Same stopping but store the first fixed point found
  + Converged to a fixed point slightly different from the one in the paper
* Store fixed point found every time
  + Found wrong fixed point

11/24

Find and control at fixed point, reward 1 if within neighborhood, reward 0 if not, reward -1 if went out of neighborhood

Radius update

Result: sparse reward signal makes the q function updated slower, not as frequently fixed points found as imagined.

11/26

Changed to henon system from neuronal network

Result: No obvious changes

11/29

Experiment with different angles of perturbation

Result: Pushing in x direction gives best result (as expected)

Penalize when existing previously enters neighborhood

Result:

12/03

Tracking founded fixed points and take average as estimated fixed point

Result:

Decrease neural net size

Result:

Fast Explore:

Result:

Larger Batch

Result: No obvious changes

Large learning rate

Result: Unstable behavior, did not really learn

12/05

12/10

12/18

12/26

01/03

Large batch:

Result: Unstable, network does not learn faster or use the batch efficiently

Wider/deeper network

Result: Deeper network has more effect on the result, learn faster and reach paper performance