1 Q1

Can the same model, trained with 200 timesteps, balance the pole for 500 timesteps? Why/why not?

I first trained the model with 200 timesteps. (model: "CartPole-v0_train_episodes_200.mdl")

Using the above model,

- Tests results with max_episode_steps = 200 : Average test reward: 200.0 episode length: 200.0
- Test results with max_episode_steps = 500 : Average test reward: 500.0 episode length: 500.0

It seems that the model trained with 200 timesteps, can also balance the pole for longer timesteps such as 500 and the average test reward is 500.

Figure 1 shows the mean and standard deviation throughout 100 independent test procedures.

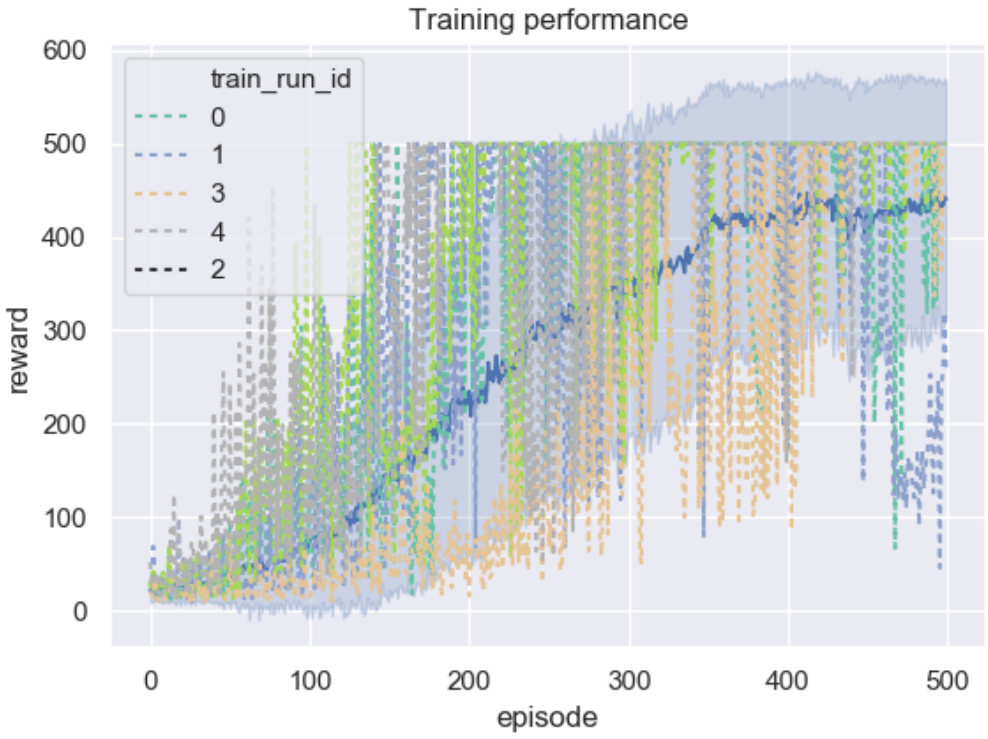

Figure 1: mean and standard deviation throughout 100 independent test procedures

If the pole is not stable and fails, this would happen at the very first steps. And if the agent learns to keep it balance for 200 steps, it can do it for more that 200 steps too.

2 Q2

The large variance could be due to the random initialization. Return is a random variable, and every time that we run our policy we may get a different return. We do not know the expect value of return, so we are sampling a bunch of trajectories from the current policy, to estimate the current expected value and update the parameters of the current policy in the direction that optimizes our estimate.

3 Reward function

The shaped reward functions are written in the Algorithm 1. It is reasonable to shape rewards such that give gradual feedback and let the agent know it's getting better and getting closer. Therefore, I believe one good choice is normal (Gaussian) shape rewards.

Algorithm 1: Reward functions

```
def new_reward(state, reward_mode=None):
if reward_mode is None:
    return 1
x, x_d, theta, theta_d = state
if(reward_mode == "fast"):
    return 1 + (np.exp(-np.power(x_d - 0.4, 2.) / (2 * np.power(1, 2.))) *
    np.exp(-np.power(theta, 2.) / (2 * np.power(1, 2.))))
    # return 0.5 + min(abs(x_d), 100)
else:
    mu = float(reward_mode)
    sig = 0.5
    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))
```

I have also added another input argument in order to decide which reward function we want to apply.

Algorithm 2: Reward modes

```
parser.add_argument("--reward_mode", type=str, default=None,
                    help="The type of reward: a float number for arbitrary point, "
                         "or \" fast \" for moving as fast as possible")
```

This input means that

- 0 : if the reward_mode == None, the default reward function will be used,
- 1 : if the reward_mode == 0 , it gives the agent extra incentive to balance the pole close to the center of the screen (close to x = 0),

- 2 : if the reward_mode == x_0 a float number between $(-2.4, 2.4)$, it incentivizes the agent to balance the pole in an arbitrary point of the screen $(x = x_0)$,
- 3 : if the reward_mode == 'fast', it makes the agent try to keep moving the cart as fast as possible, while still balancing the pole.

In the reward functions 1 and 2 the reward is defined as a gaussian with mean at the desired point (x_0) and standard deviation of 0.5. It means that when the cart position x is close to the desired position, the reward would be larger and in the other positions it would be smaller.

Foe example, I set the training_episodes = 500, reward_mode = 0 and the saved model is "CartPole-v0_reward_1.mdl". The reward history in training is shown in figure 2. The average test reward is 189.115 and the frequency of cart position in test is shown in figure 3. As it can be seen, the cart is most of the time around center $(x_0 = 0)$.

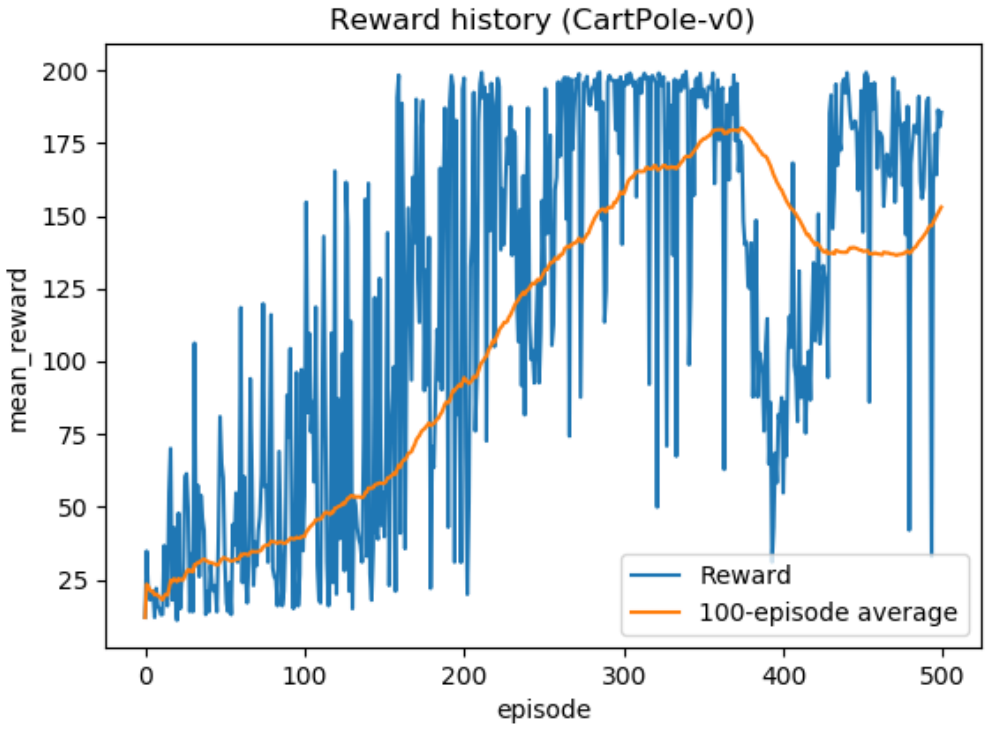

Figure 2: Reward history of training with 500 episodes and reward function 1 (balance the pole close to the center of the screen)

The mean and standard deviation throughout 10 independent train procedures is shown in figure 4.

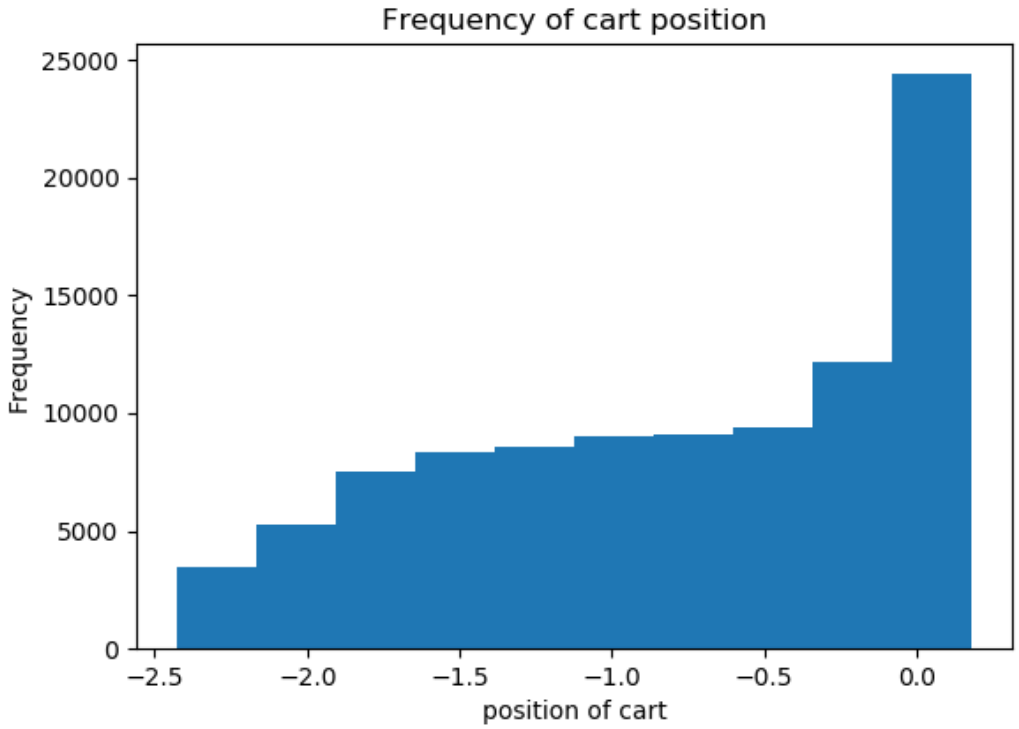

Figure 3: Frequency of cart position at test time with reward function 1 (balance the pole close to the center of the screen)

In order to check the second reward function, I set training_episodes = 500, reward_mode = 1.5 and the saved model is "CartPole-v0_reward_2.mdl". I first trained using maximum episode steps = 200, but the results were not satisfying, so I increased it to 500 and set the training_episodes = 1000. Teh The frequency of the cart positions in test run, is shown in the figure 5. Most of the time the cart is hanging around 1.5. The mean and standard deviation throughout 10 independent train procedures is shown in figure 6.

In the reward function 3, I defined the following function:

$1 + gaussian(\dot{x}, 0.4, 1) * gaussian(\theta, 0, 1)$

It means that whenever the velocity is around maximum (0.4) and pole angel is around 0 (balanced) the reward would be maximum.

The model is saved as "CartPole-v0_reward_3.mdl" and the figure 7 shows the mean and standard deviation throughout 10 independent training procedures. Frequency of cart velocity and pole angel are shown in the figures 8 and 9 respectively. It can be seen that the velocity is not at the maximum level, however the pole angel is mostly around 0 and the pole is balanced.

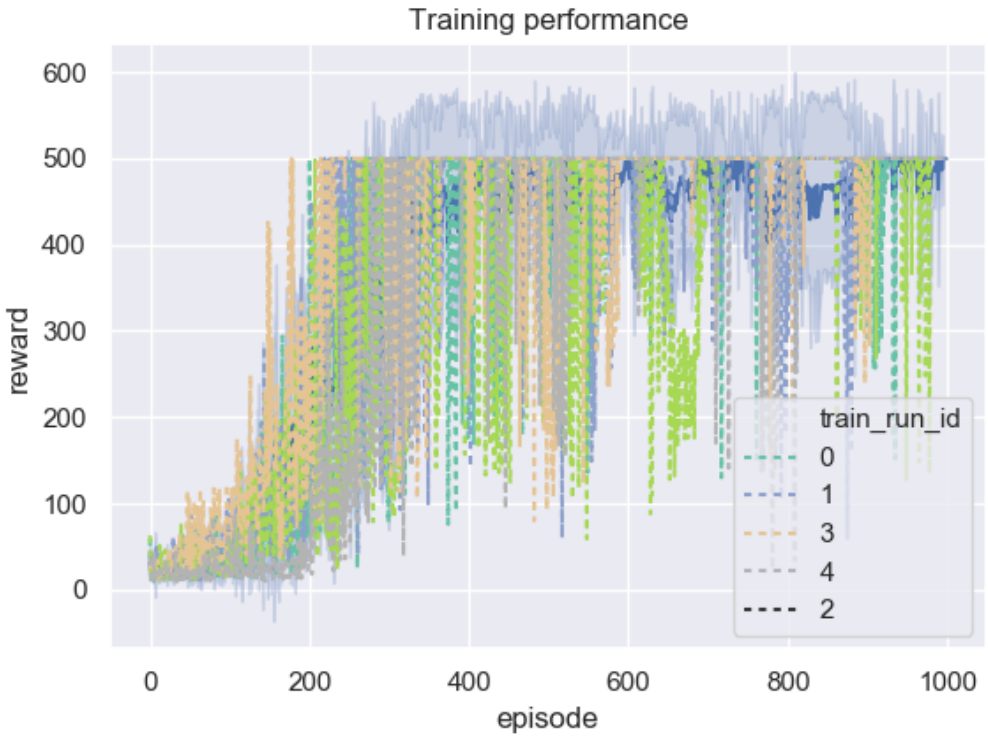

Figure 4: Variance in performance between multiple repetitions of the experiment for the reward function 1 (balance the pole close to the center of the screen)

4 Q3

changing the reward function has a great impact on the running time. Reward shaping is a big deal. The reason for longer time could be not very well defined reward function and the agent is off exploring without having a reasonable feedback from the environment.

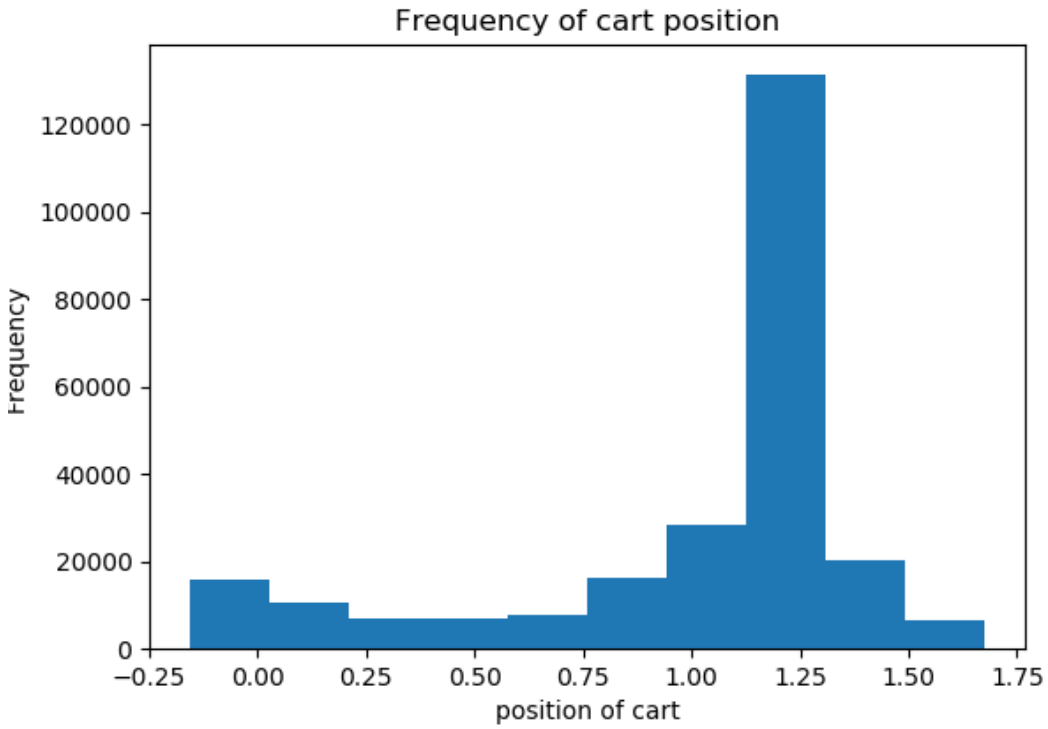

Figure 5: Frequency of cart position at test time with reward function 2 (balance the pole close to the position x_0 = 1.5)

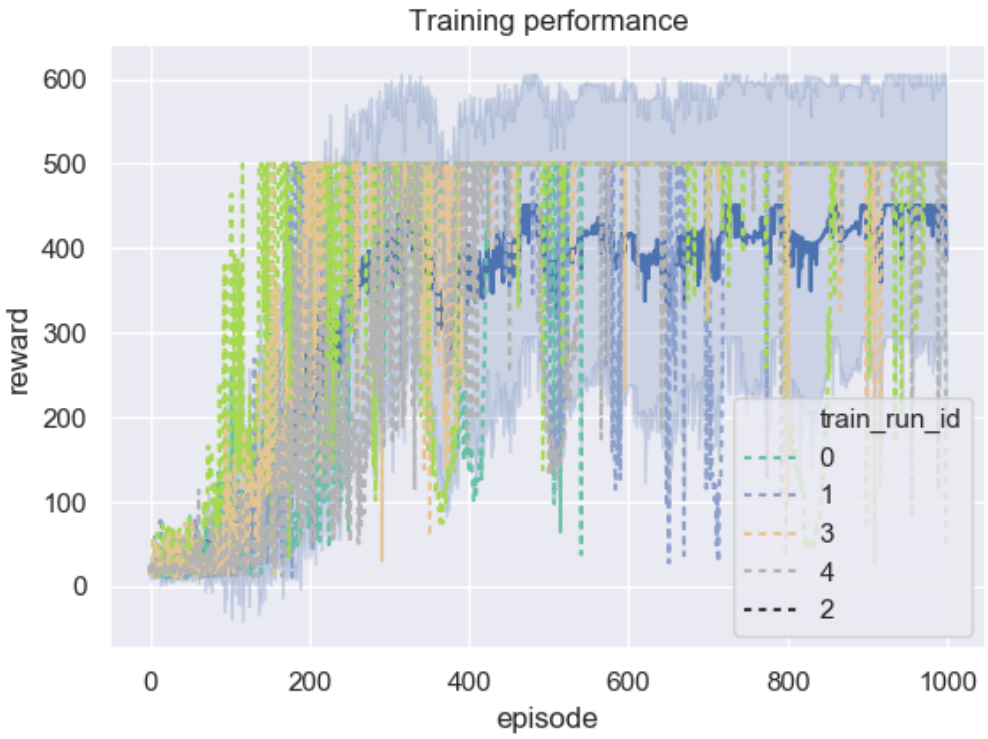

Figure 6: Variance in performance between multiple repetitions of the experiment for the reward function 2 (balance the pole close to the position x_0 = 1.5)

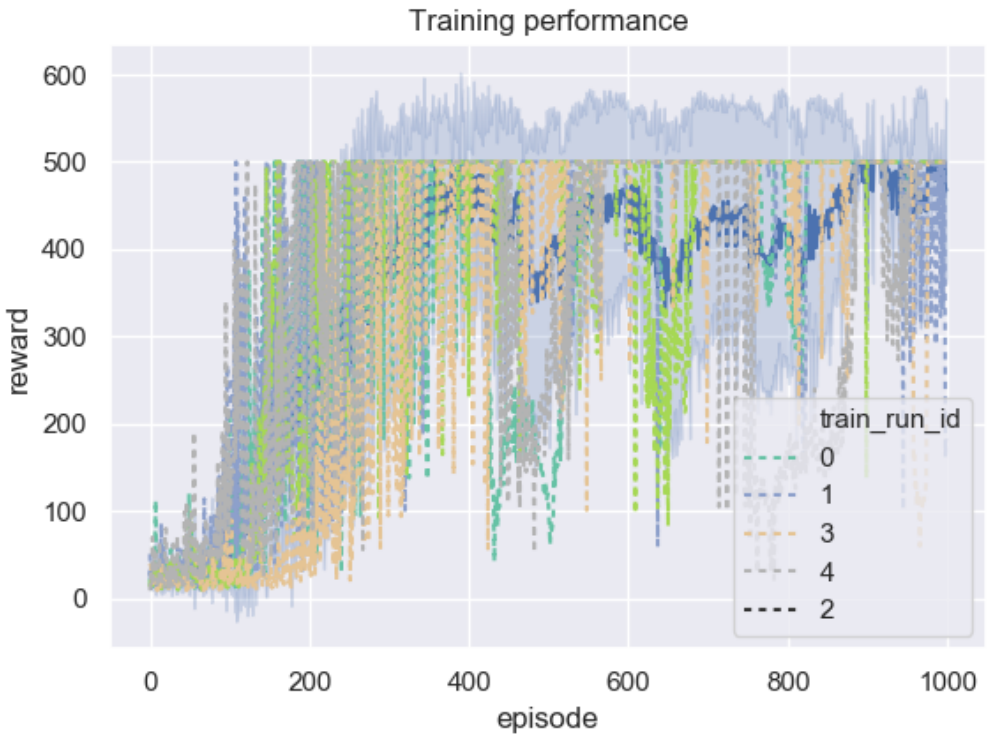

Figure 7: Variance in performance between multiple repetitions of the experiment for the reward function 3 (balance the pole at the maximum speed)

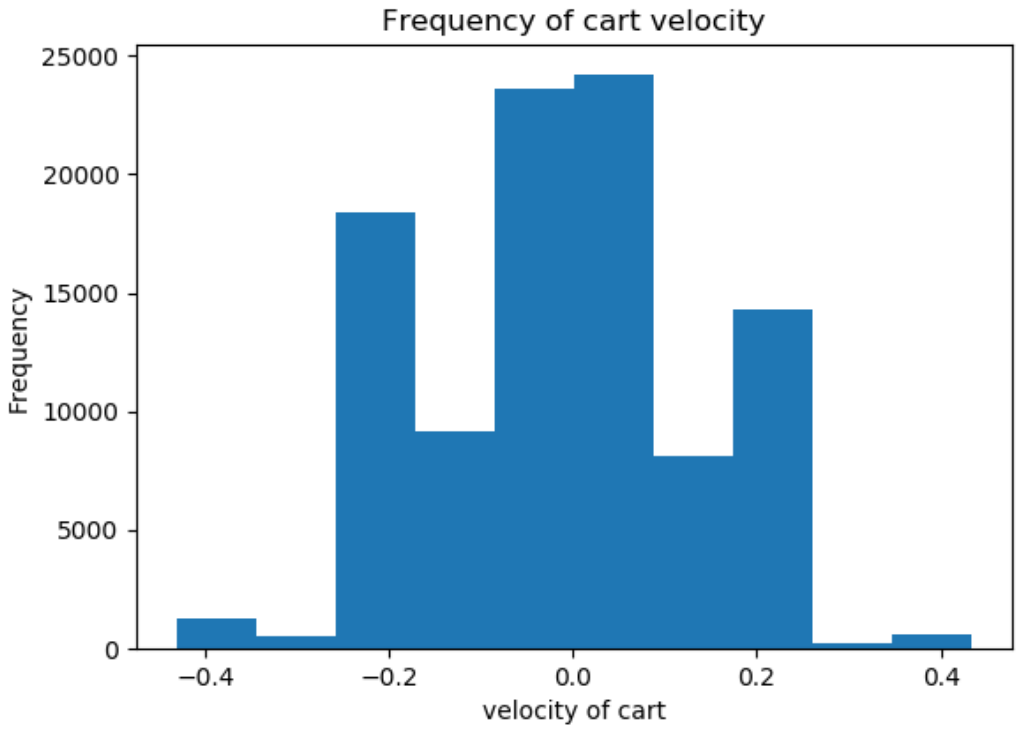

Figure 8: Frequency of cart velocities at test time with reward function 3 (balance the pole at the maximum speed)

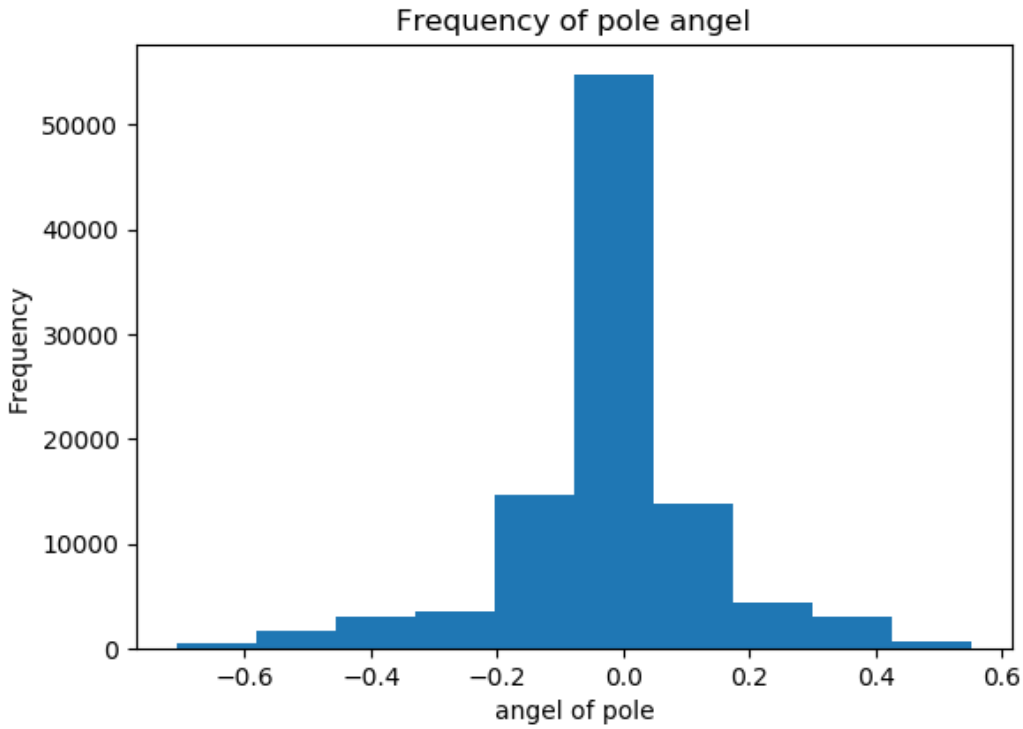

Figure 9: Frequency of pole angels at test time with reward function 3 (balance the pole at the maximum speed)