1.2

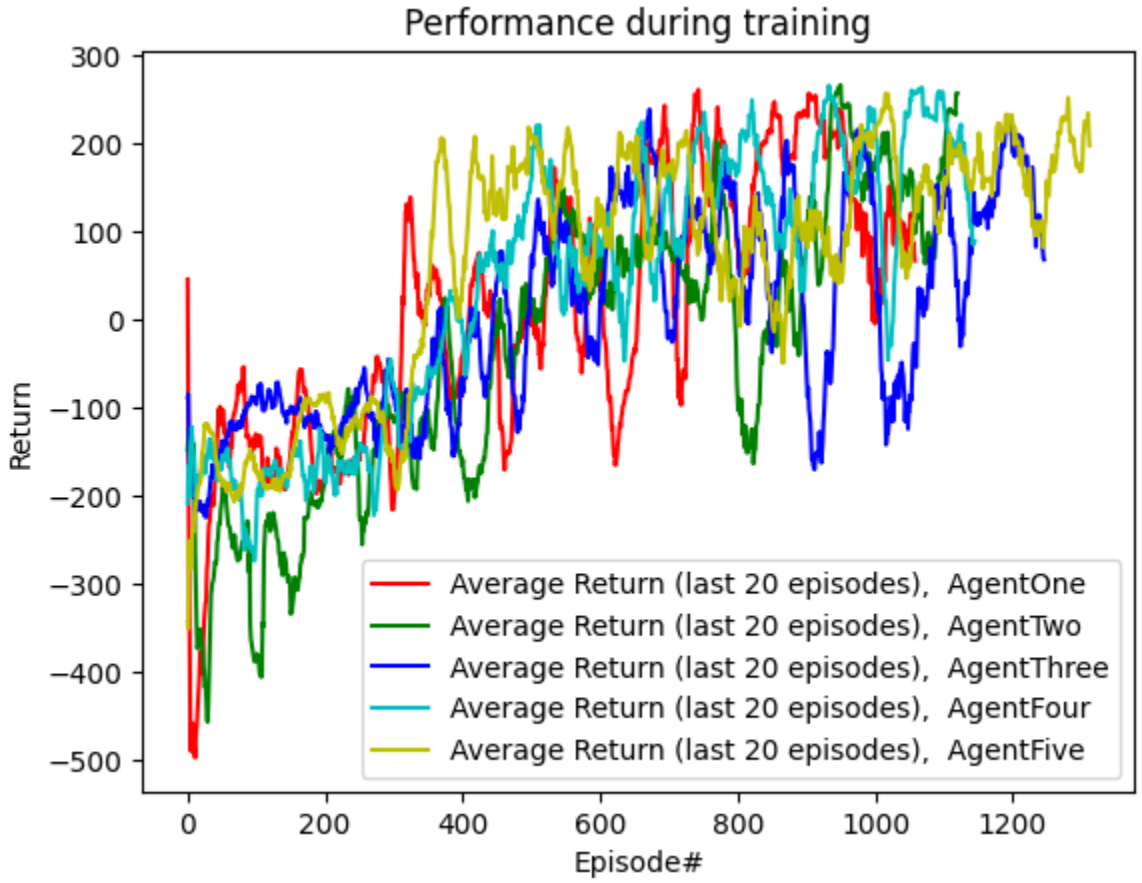


Fig 1: Moving average return over episodes for 5 iterations of the DDPG agent.

The variance comes from the random walk nature of the noise that is added to the action, as well the coming from the 2 (4) neural networks that are trained in some part with the data they previously generated. Similar to random walk, the variance is caused by starting a stochastic step from the position of another stochastic step, which compounds the variance. In a simple binary random walk, the variance is the square of the step size; so applying the same principle of variance depending on step size here, to reduce the variance, we must either reduce the noise, or reduce the weight with which each action is used to update the model (i.e. reduce the learning rate, tau, or gamma). Increasing the buffer size can make sure the data is decorrelated better, decreasing the variance that way as well, although this technique seems to have diminishing returns since at some point the data is already sufficiently decorrelated.

1.3

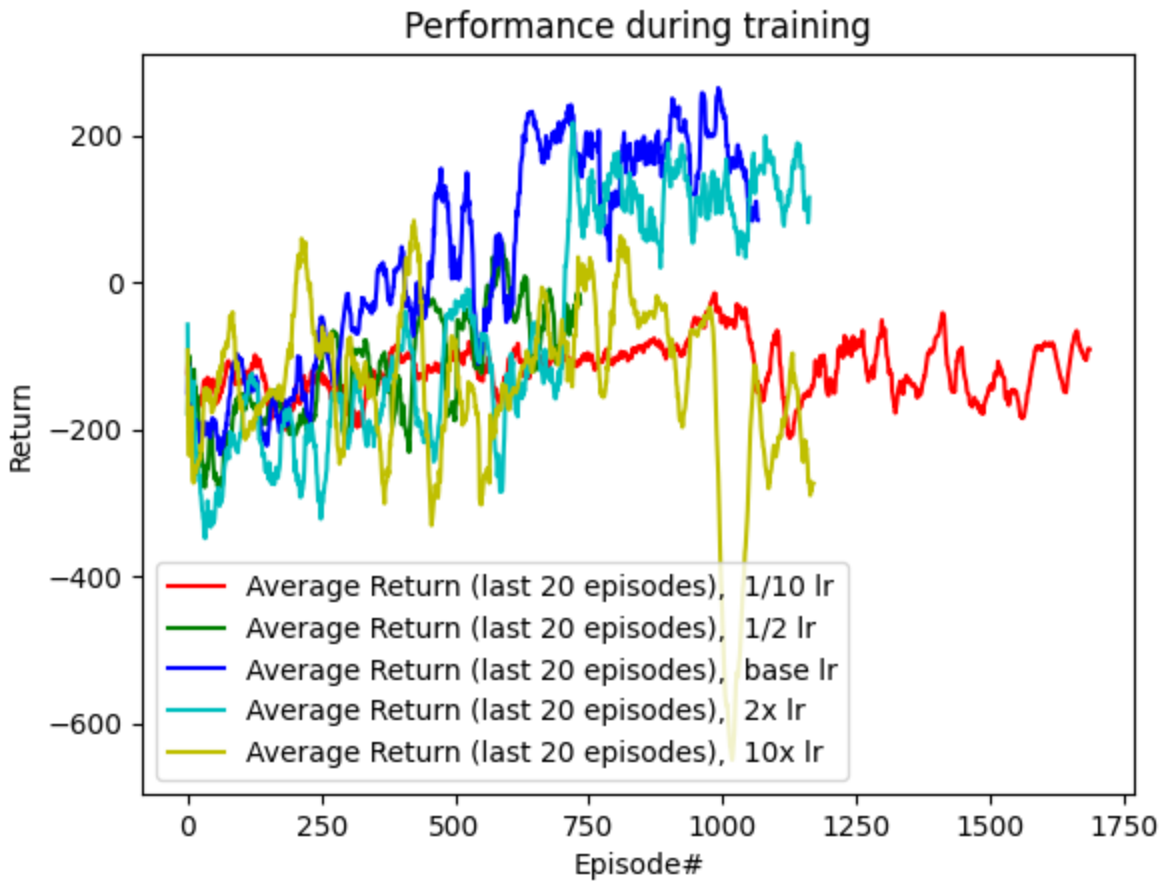


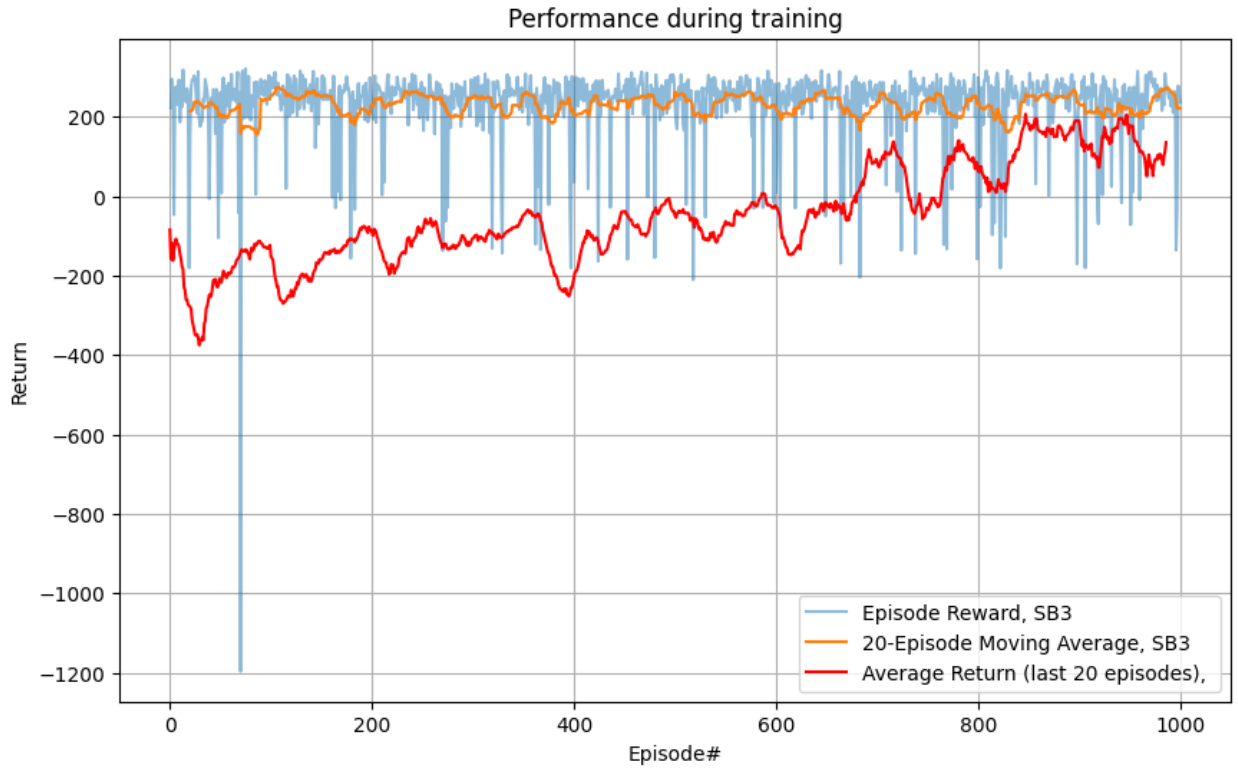
Fig 2: Moving average return over episodes for 5 different learning rate scalings (of both actor and critic).

Looking at this graph, we can see that 1x and 2x lr converge decently to the desired reward value of 150. The 10x lr is notable for huge fluctuations in return which look like it's converging at first, but seems to always try to find another local minimum, resulting in a lot of variance in return. The ½ lr seems to follow the upward trend, but needs more training to confirm convergence. It does seem to fluctuate around the starting values, and nevertheless does not reach 0 after 500000 timesteps. This is expected since a lower lr takes longer to reach a certain return value. The 1/10 lr seems to have the shortest episodes, training for the longest number of episodes in 500000 timesteps without much improvement; we can conclude that this agent has failed to learn since its simulations terminate very quickly in a failure – likely due to the loss being unable to escape some local minimum.

Interestingly, we do note that the 1x lr does converge to higher values earlier than 2x lr, which can be attributed to higher lr's tendency to fluctuate in return, trying out many local minima before settling on one, which delays higher reward values. Learning rate tuning is truly more of an art than a science, and we do not know whether each minimum that was picked was truly optimal, we just know it's "good enough." Another interesting thing to note is that ½ lr and 1x lr do have fewer episodes compared to the other lr tunings, which does mean that the agent is

spending more timesteps per episode, confirming that it is learning and “hanging on.” We can interpret this as these lr tunings having a steadier learning curve than the others. Would be very interesting to see the ½ lr training extended to see if it converges. Its return does fluctuate quite a bit in the early stages compared to base lr though, so it’s likely that it hasn’t found the optimal minimum yet due to lack of lr to “jump out of a hole.”

1.4



SB3 seems to converge to a more optimal minimum than my implementation, but they are pretty close in performance. It’s not shown here but (in the colab notebook) the rate of convergence and training seem to be pretty similar between the two.

Note for 1.2 and 1.3: My colab submission will not have all the images, as I ran these on multiple instances due to my trouble with computational resources. However, the code used to produce the above images are the same as that in the colab submission.

Q2:

Actions:
R: right
L: left
S: straight

RL Hw 4 Q2

1. $H(p) = -\sum_x p(x) \log(p(x))$

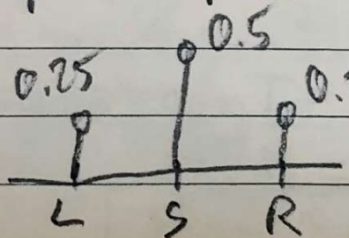
2. $p(L) + p(R) + p(S) = 1 \rightarrow p(S) = \frac{1}{2} = 0.5$
   $p(L) + p(R) = p(S)$

Let's examine now $p(L|turn)$ and $p(R|turn)$
$p(L|turn) + p(R|turn) = 1$
maximize entropy!
write $p(R|turn)$ as $1 - p(L|turn)$

$$H(p(L|turn)) = -p(L|turn)\log(p(L|turn)) - (1-p(L|turn))\log(1-p(L|turn))$$

$$\nabla H(p(L|turn) = -p \cdot \frac{1}{p} - \ln(p) + \ln(1-p) + (1-p)\frac{1}{1-p}$$

Set to 0: $0 = \ln(1-p(L|turn)) - \ln(p(L|turn))$
$1 - p(L|turn) = p(L|turn)$
$\therefore p(L|turn) = 0.5$
given that $p(turn) = 1 - p(S) = 0.5$,
$p(R) = p(L) = 0.5 \cdot p(turn) = 0.25$



→ this matches the intuition that probability should spread evenly (where possible) for max entropy!