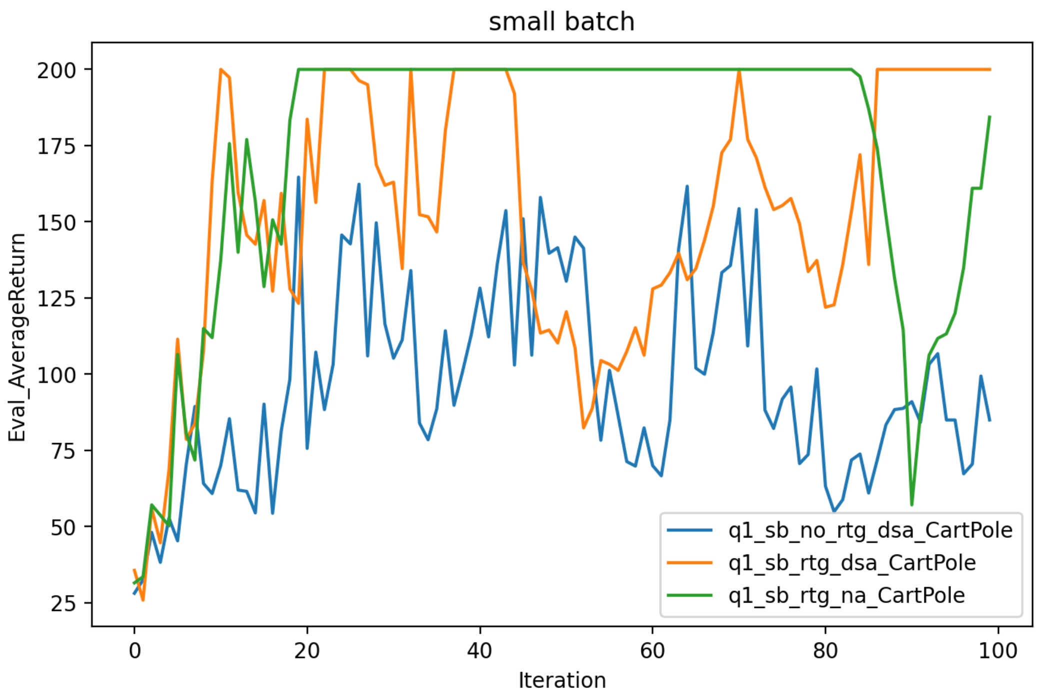
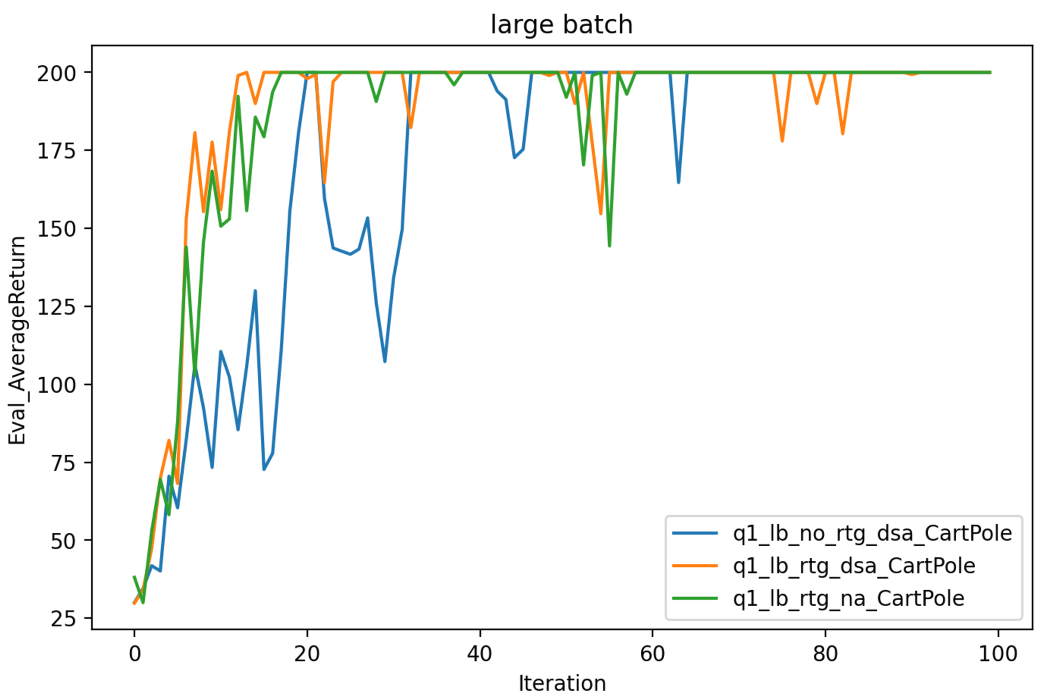
HW 2

Experiment 1

Commands:

```
python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-dsa --exp_name q1_sb_no_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg -dsa --exp_name q1_sb_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 1000 \
-rtg --exp_name q1_sb_rtg_na

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
-dsa --exp_name q1_lb_no_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
-rtg -dsa --exp_name q1_lb_rtg_dsa

python cs285/scripts/run_hw2.py --env_name CartPole-v0 -n 100 -b 5000 \
-rtg --exp_name q1_lb_rtg_na
```





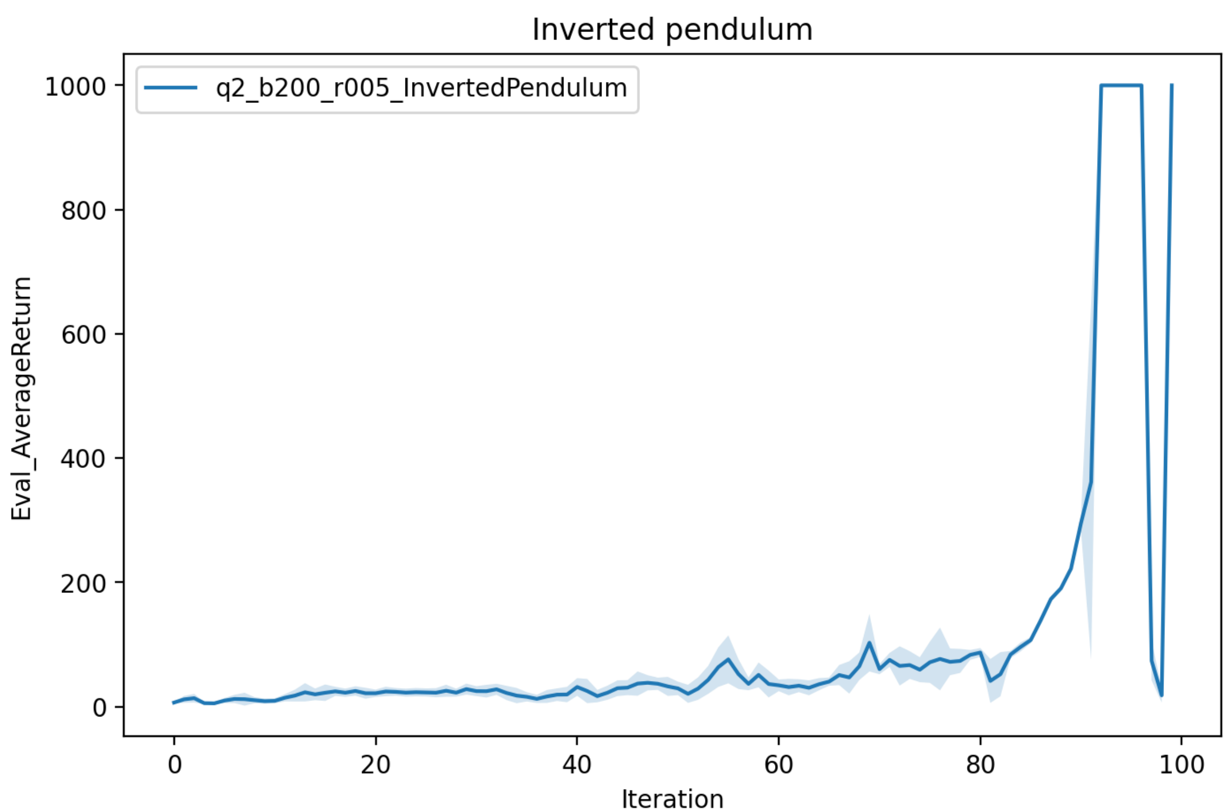
- Without advantage standardization (among `dsa`), experiments **with reward-to-go** (orange curves) have higher average return for both small-batch and large-batch experiments.

- Under the same batch size, among experiments with reward-to-go (`rtg`), experiments **with standardized advantages** (green curves) perform **better** than those without (orange curves).
- For experiments with the same reward-to-go and normalization settings, those with **larger** batch size perform **better**.

Experiment 2

Command:

```
python cs285/scripts/run_hw2.py --env_name InvertedPendulum-v4 \
--ep_len 1000 --discount 0.9 -n 100 -l 2 -s 64 -b 200 -lr 0.05 -rtg \
--exp_name q2_b200_r005
```

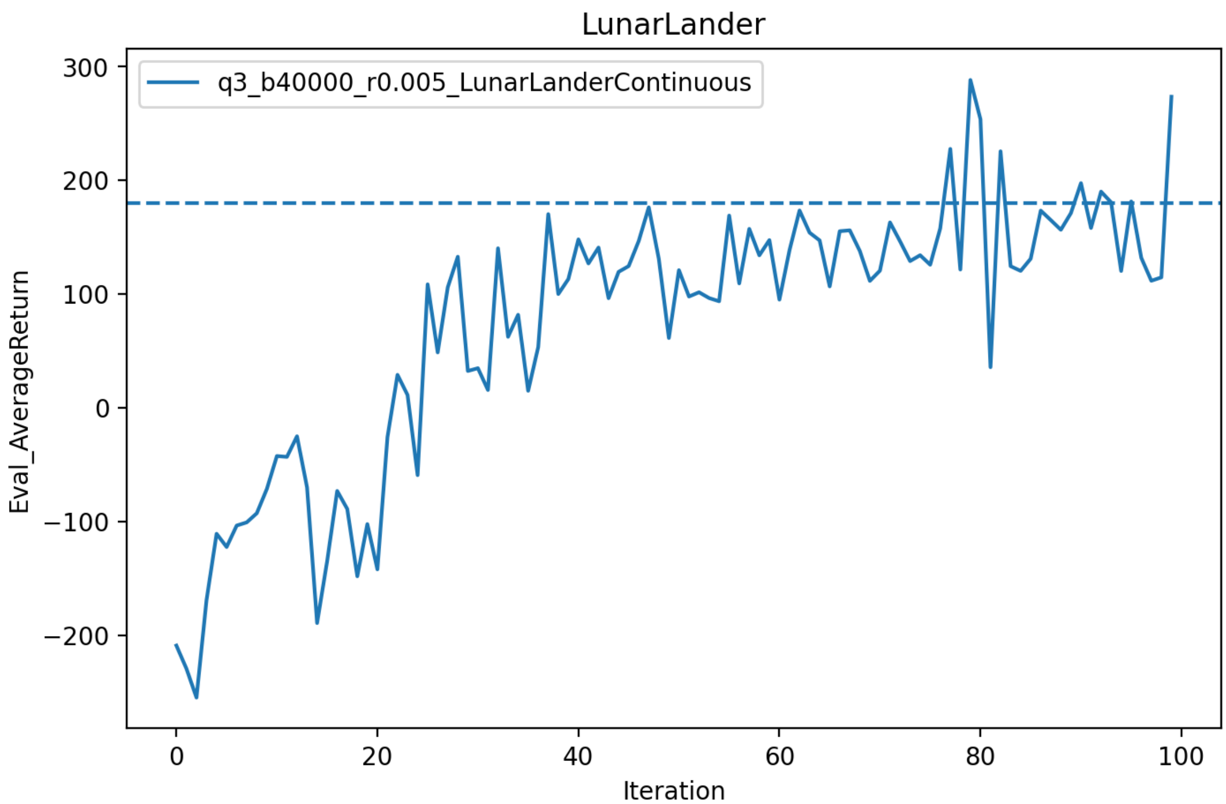


The smallest batch size and largest learning rate is found to be **200** and **0.05,** respectively. The average return with 1σ errorband is plotted above.

Experiment 3

Command:

```
python cs285/scripts/run_hw2.py \
--env_name LunarLanderContinuous-v2 --ep_len 1000 \
--discount 0.99 -n 100 -l 2 -s 64 -b 40000 -lr 0.005 \
--reward_to_go --nn_baseline --exp_name q3_b40000_r0.005
```



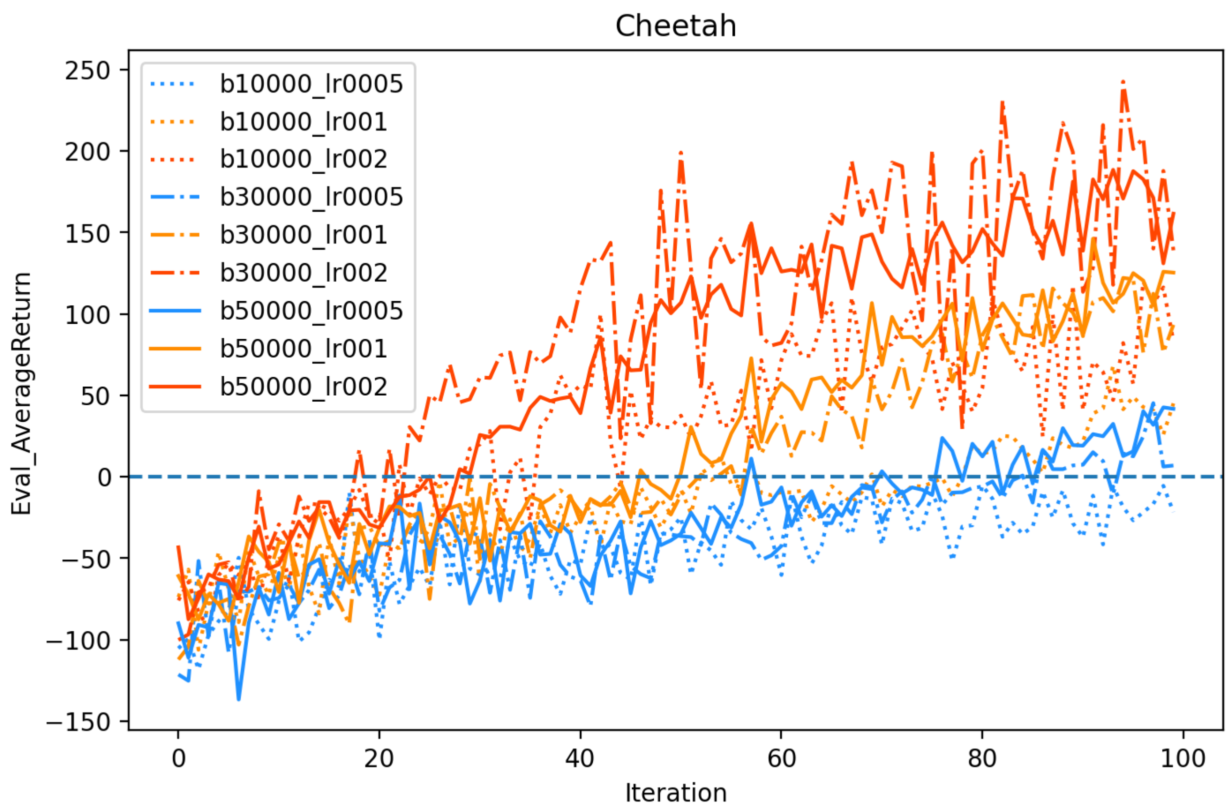
The average return is around 180 (dashed line) after ~80 iterations.

Experiment 4

Commands:

```
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.005 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr0005_rtg_nnbaseline
```

```
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.005 -rtg --nn_baseline \
--exp_name q4_search_b30000_lr0005_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.005 -rtg --nn_baseline \
--exp_name q4_search_b50000_lr0005_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.01 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr001_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.01 -rtg --nn_baseline \
--exp_name q4_search_b30000_lr001_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.01 -rtg --nn_baseline \
--exp_name q4_search_b50000_lr001_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 10000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b10000_lr002_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 30000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b30000_lr002_rtg_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_search_b50000_lr002_rtg_nnbaseline
```



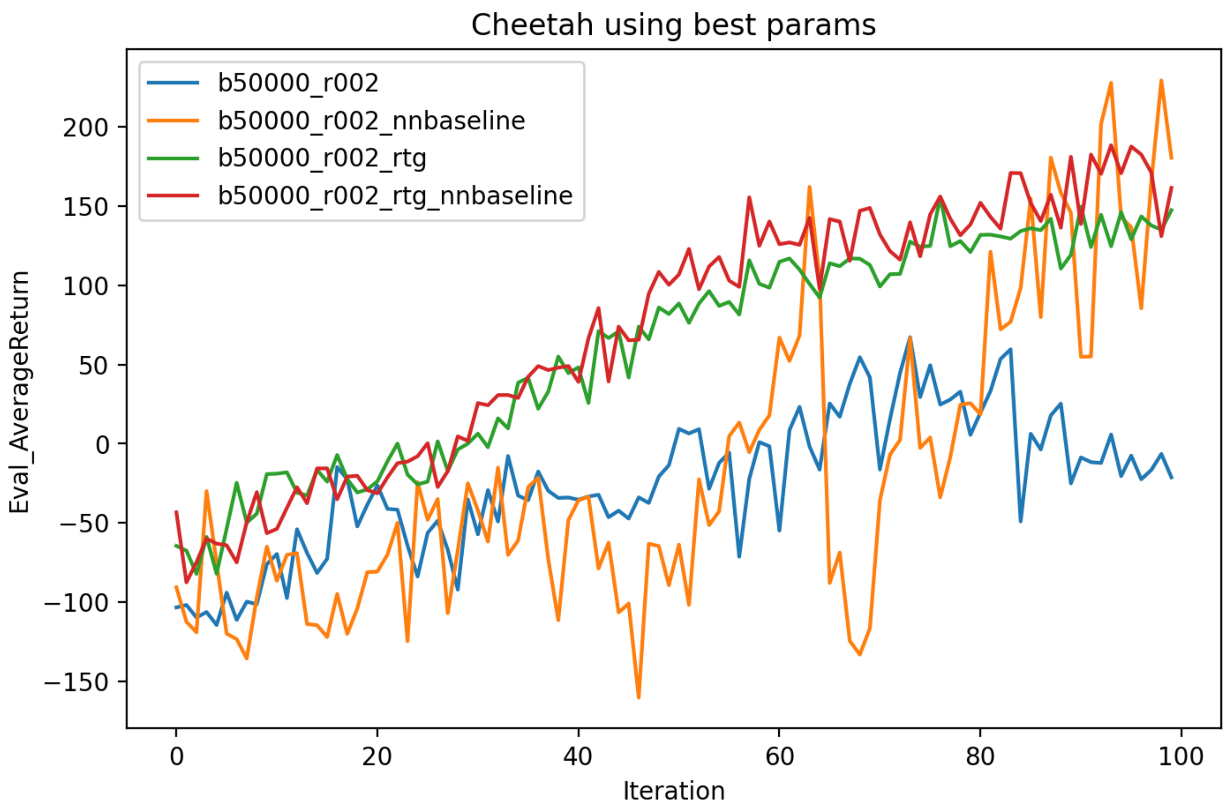
For a given learning rate (same color, different line style), increasing batch size does not improve the average return by a lot, although some improvement is visible. On the other hand, given a batch size (same line style, different colors), i**ncreasing the learning rate significantly improves** the model performance.

The best batch size and learning rate are 50000 and 0.02, respectively.

Commands:

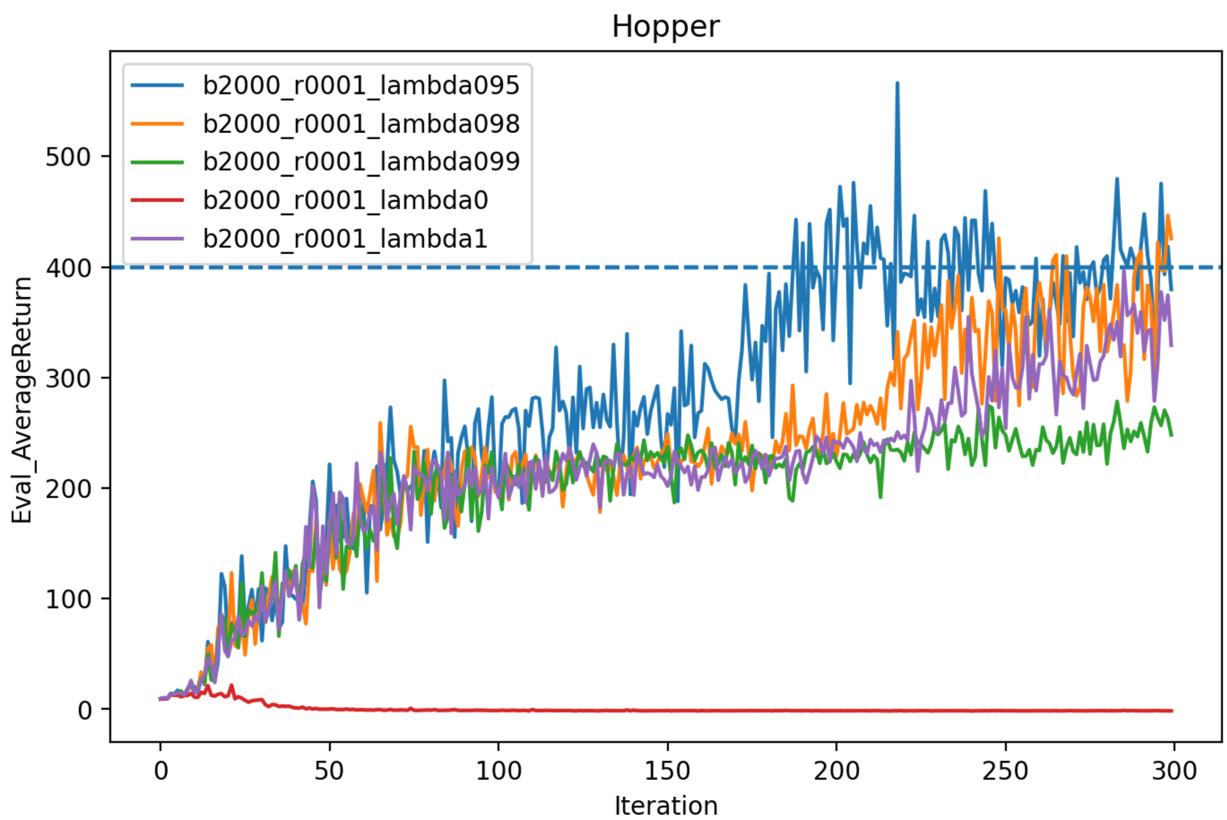
```
python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 \
--exp_name q4_b50000_r002

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg \
--exp_name q4_b50000_r002_rtg

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 --nn_baseline \
```

```
--exp_name q4_b50000_r002_nnbaseline

python cs285/scripts/run_hw2.py --env_name HalfCheetah-v4 --ep_len 150 \
--discount 0.95 -n 100 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline \
--exp_name q4_b50000_r002_rtg_nnbaseline
```



The average return of reward-to-go + baseline (red curve) is around 200 during the end of training.

Experiment 5



Because increase λ increases variance, it helps the model achieve better rewards but the training speed becomes slower. Experiments with non-zero λ all achieve ~400 in rewards.