Problem 1. State-Dependent Baseline

(a) One has

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))] \\
&= \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[\mathbb{E}_{(\tau/s_t,a_t)\sim p_\theta(\tau/s_t,a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))|s_t,a_t]] \\
&= \mathbb{E}_{(s_t,a_t)\sim p_\theta(s_t,a_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))] \\
&= \int_{s_t}\int_{a_t} p_\theta(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))da_t ds_t \\
&= \int_{s_t}\int_{a_t} p_\theta(s_t)\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)(b(s_t))da_t ds_t \\
&= \int_{s_t} p_\theta(s_t)b(s_t)\int_{a_t} \nabla_\theta \pi_\theta(a_t|s_t)da_t ds_t \\
&= \int_{s_t} p_\theta(s_t)b(s_t)\nabla_\theta\left(\int_{a_t}\pi_\theta(a_t|s_t)da_t\right)ds_t \\
&= \int_{s_t} p_\theta(s_t)b(s_t)\nabla_\theta(1)ds_t \\
&= 0
\end{aligned}
$$

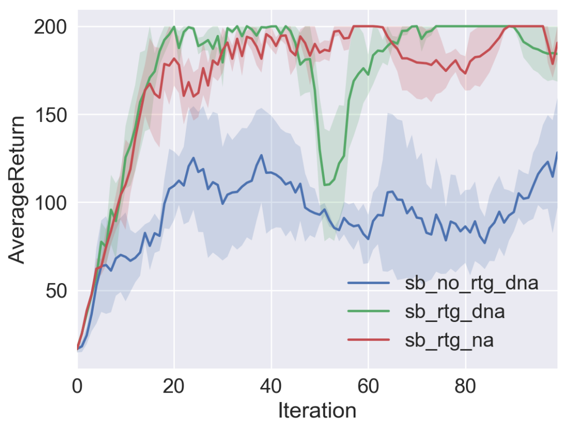
(b) (a) For the inner expectation, conditioning on $(s_{1:t}, a_{1:t-1})$ is equivalent to conditioning only on s_t because our model is a Markov decision process. This means that conditioned on s_t, the rest of the trajectory (specifically a_t) is independent of the trajectory leading up to s_t. Thus, $\pi_\theta(a_t|s_t)$ and $b(s_t)$ are independent of the past when conditioned on s_t.

(b) One has

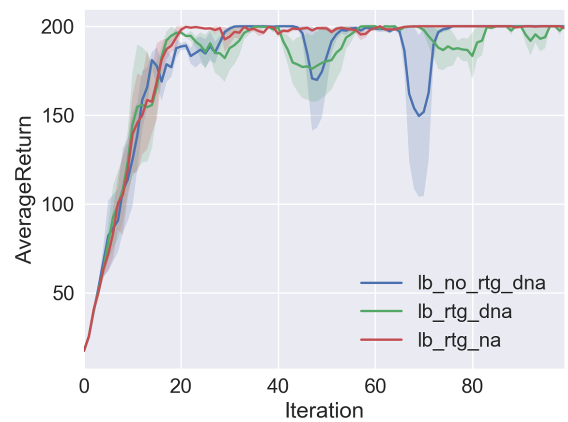
$$
\begin{aligned}
&\mathbb{E}_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}[\mathbb{E}_{(s_{t+1:T},a_{t:T})\sim p_\theta(s_{t+1:T},a_{t:T})}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)|(s_{1:t},a_{1:t-1})]] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}[\mathbb{E}_{(s_{t+1:T},a_{t:T})\sim p_\theta(s_{t+1:T},a_{t:T})}[\nabla_\theta \log \pi_\theta(a_t|s_t)b(s_t)|s_t]] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}[b(s_t)\,\mathbb{E}_{a_t\sim\pi_\theta(a_t|s_t)}[\nabla_\theta \log \pi_\theta(a_t|s_t)]] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}\left[b(s_t)\int_{a_t}\pi_\theta(a_t|s_t)\nabla_\theta \log \pi_\theta(a_t|s_t)da_t\right] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}\left[b(s_t)\int_{a_t}\nabla_\theta \pi_\theta(a_t|s_t)da_t\right] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}\left[b(s_t)\nabla_\theta\left(\int_{a_t}\pi_\theta(a_t|s_t)da_t\right)\right] \\
&= \mathbb{E}_{(s_{1:t},a_{1:t-1})\sim p_\theta(s_{1:t},a_{1:t-1})}[b(s_t)\nabla_\theta(1)] \\
&= 0
\end{aligned}
$$

CS 294-112 Deep RL Homework 2

Problem 4. CartPole



CartPole with small batch size (b=1000)



CartPole with large batch size (b=5000)

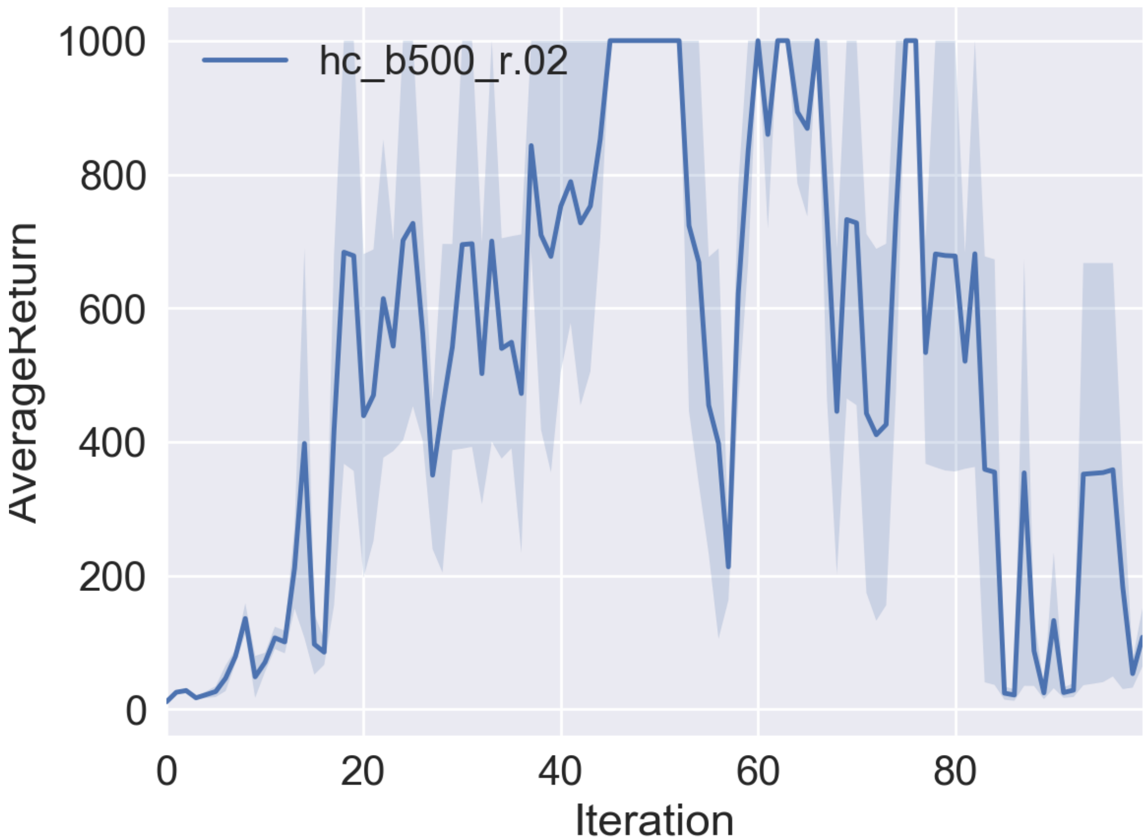
Command Line Configurations

```
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 3 -dna --exp_name sb_no_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 3 -rtg -dna --exp_name sb_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 1000 -e 3 -rtg --exp_name sb_rtg_na
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -dna --exp_name lb_no_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -rtg -dna --exp_name lb_rtg_dna
python train_pg_f18.py CartPole-v0 -n 100 -b 5000 -e 3 -rtg --exp_name lb_rtg_na
```

Questions

- Without advantage-centering, the gradient estimator using reward-to-go has significantly better performance.
- Advantage-centering helps by making the average return much more stable across iterations.
- Large batch size gives better performance than small batch size across all configurations.

Problem 5. InvertedPendulum



Inverted Pendulum with b=500, r=0.02

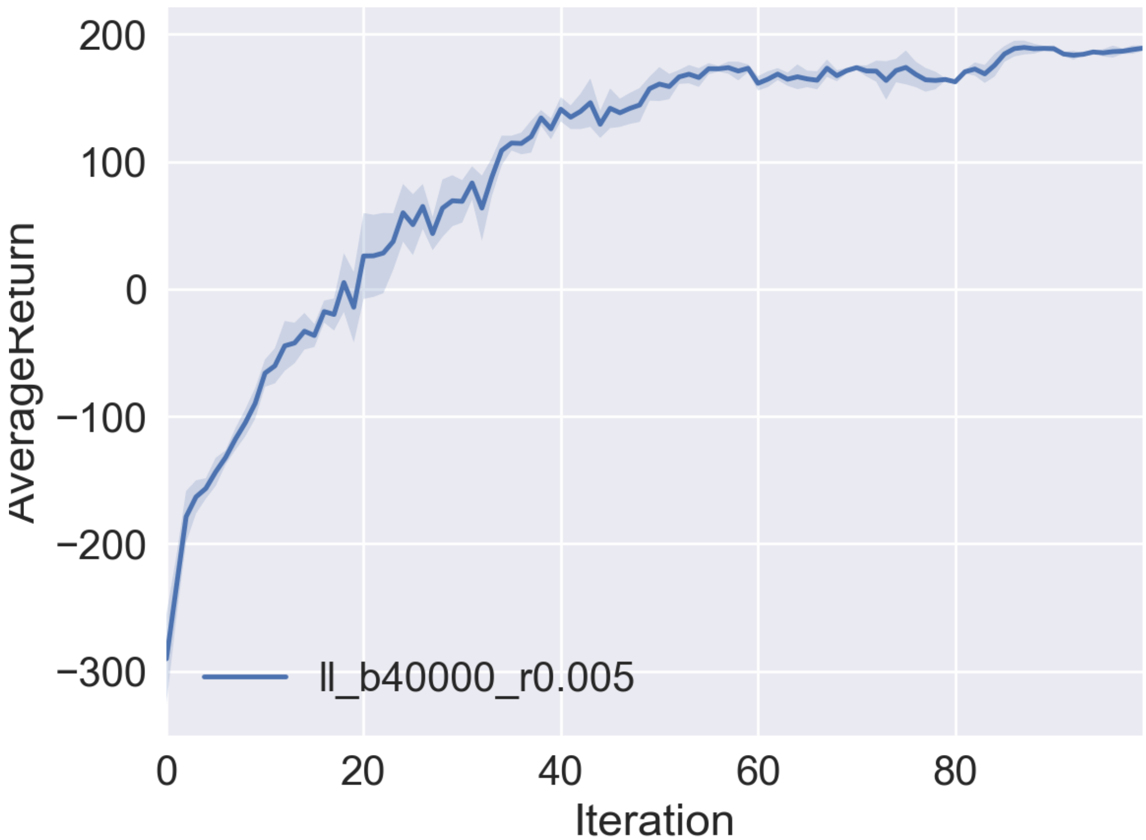
Command Line Configurations

```
python train_pg_f18.py InvertedPendulum-v2 -ep 1000 --discount 0.9 -n 100
-e 3 -l 2 -s 64 -b 500 -lr .02 -rtg --exp_name hc_b500_r.02
```

Smallest batch size: 500

Largest learning rate: 0.02

Problem 7: LunarLander



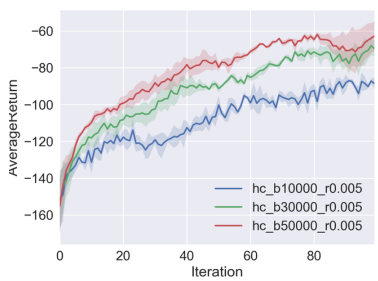
Lunar Lander with b=40000, r=0.005

Command Line Configuration

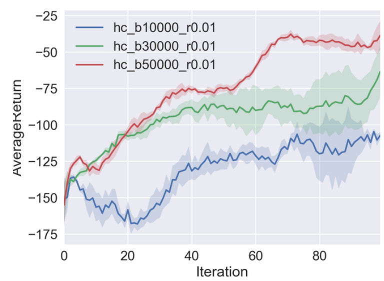
```
python train_pg_f18.py LunarLanderContinuous-v2 -ep 1000 --discount 0.99 -
n 100 -e 3 -l 2 -s 64 -b 40000 -lr 0.005 -rtg --nn_baseline --exp_name ll_
b40000_r0.005
```

Problem 8: HalfCheetah

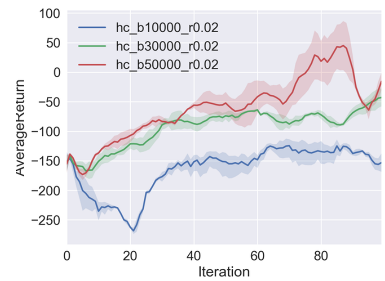
Hyperparameter Search



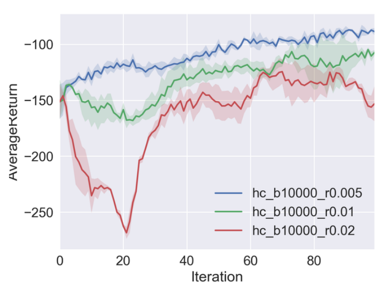
Compare batch size for lr=0.005



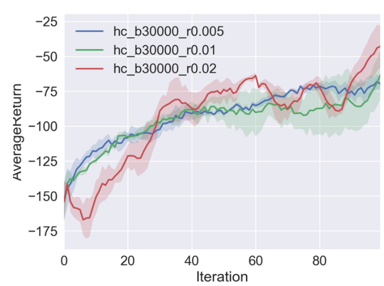
Compare batch size for lr=0.01



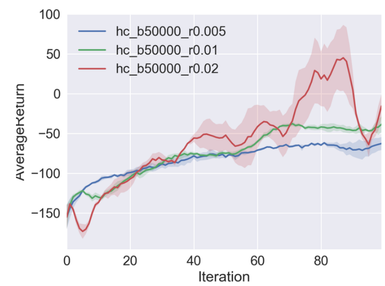
Compare batch size for lr=0.02



Compare lr for b=10000



Compare lr for b=30000



Compare lr for b=50000

Command Line Configurations

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 10000 -lr 0.005 --exp_name hc_b10000_r0.005
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 10000 -lr 0.01 --exp_name hc_b10000_r0.01
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 10000 -lr 0.02 --exp_name hc_b10000_r0.02
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 30000 -lr 0.005 --exp_name hc_b30000_r0.005
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 30000 -lr 0.01 --exp_name hc_b30000_r0.01
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 30000 -lr 0.02 --exp_name hc_b30000_r0.02
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.005 --exp_name hc_b50000_r0.005
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.01 --exp_name hc_b50000_r0.01
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.9 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 --exp_name hc_b50000_r0.02
```

Batch size

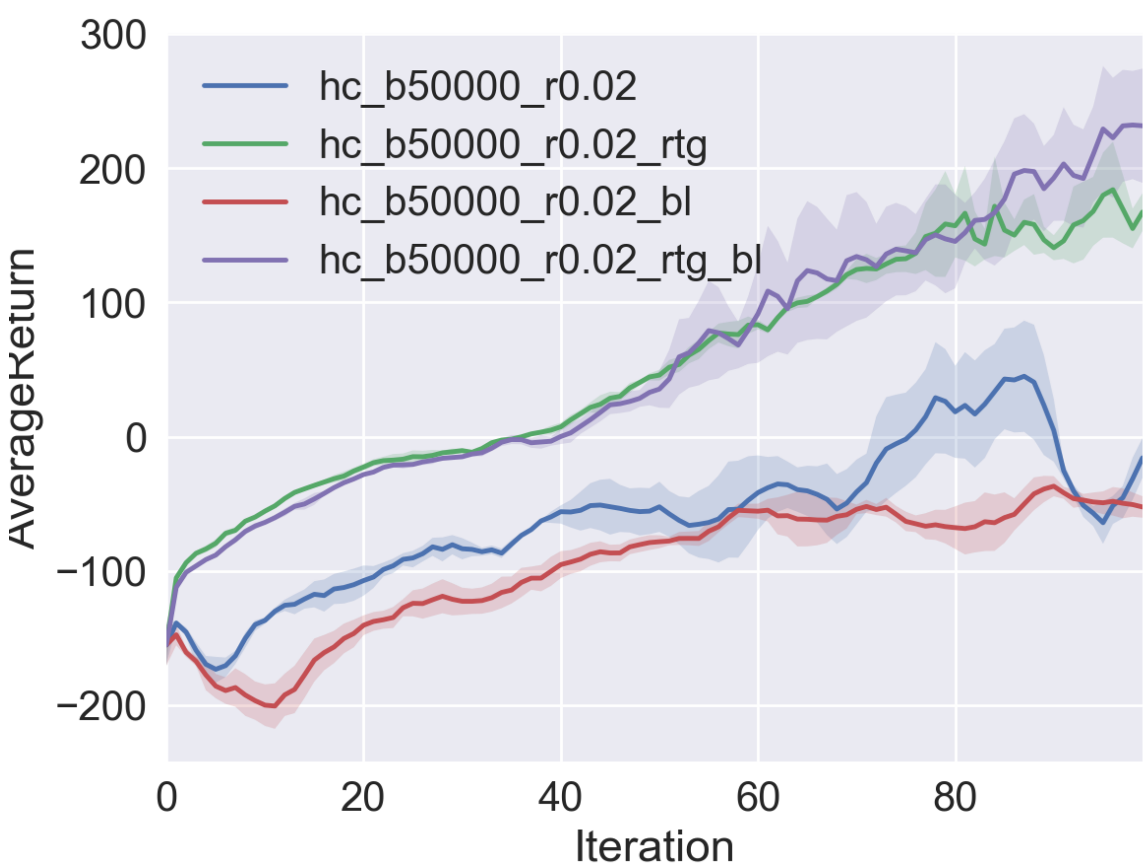
- For all learning rates, performance improved with larger batch size.

Learning rate

- For a batch size of 10000, smaller learning rates had better performance.
- For a batch size of 30000, all three learning rates had similar performance, with 0.02 performing slightly better than 0.005 and 0.01 but exhibiting significantly higher variance.
- For a batch size of 50000, performance improved with larger learning rates. A learning rate of 0.02 had the best performance but still with high variance.

Best hyperparameter values

- $b* = 50000$
- $r* = 0.02$



Final plot with b=50000, r*=0.02*

Command Line Configuration

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 --exp_name hc_b50000_r0.02
```

```
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --exp_name hc_b50000_r0.02_rtg
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 --nn_baseline --exp_name hc_b50000_r0.02_bl
python train_pg_f18.py HalfCheetah-v2 -ep 150 --discount 0.95 -n 100 -e 3 -l 2 -s 32 -b 50000 -lr 0.02 -rtg --nn_baseline --exp_name hc_b50000_r0.02_rtg_bl
```