

# CS185 Homework 2 Report

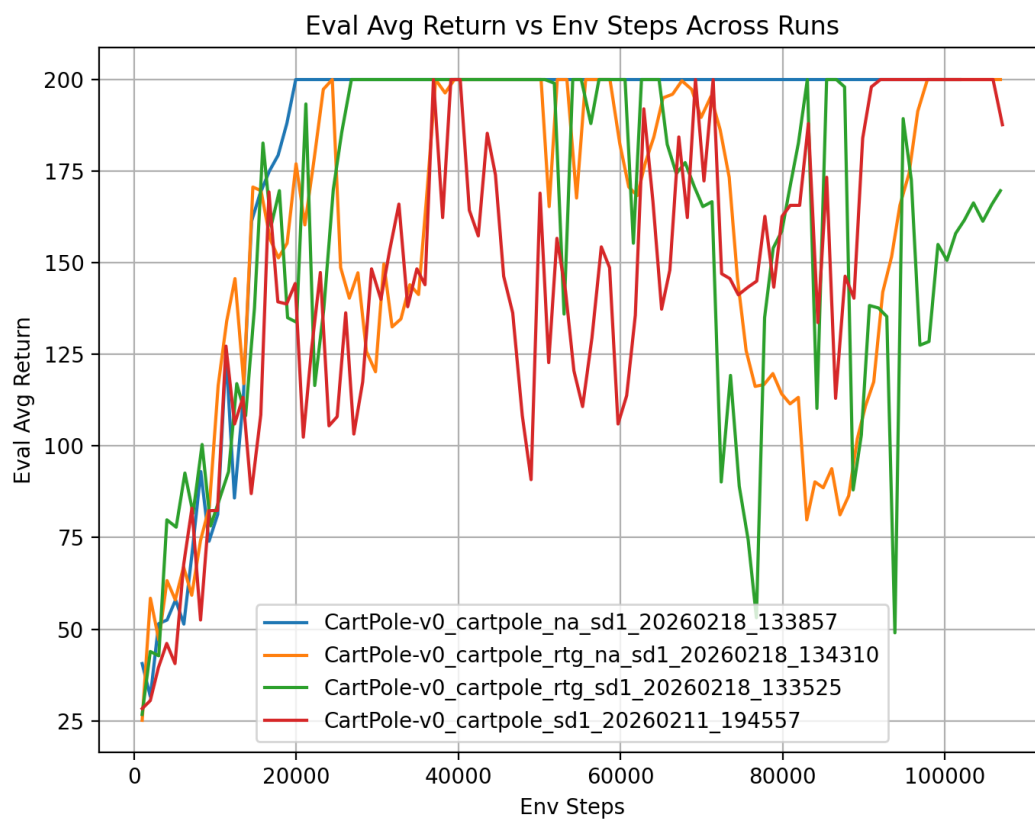
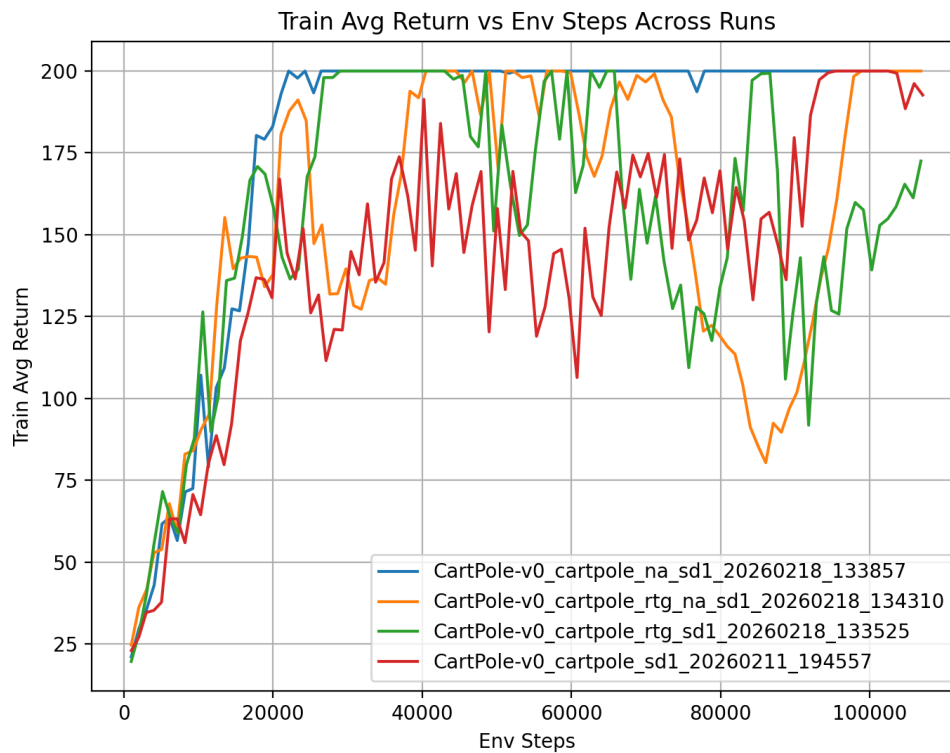
## Policy Gradients

Name Of Student: Chong Wei Kee

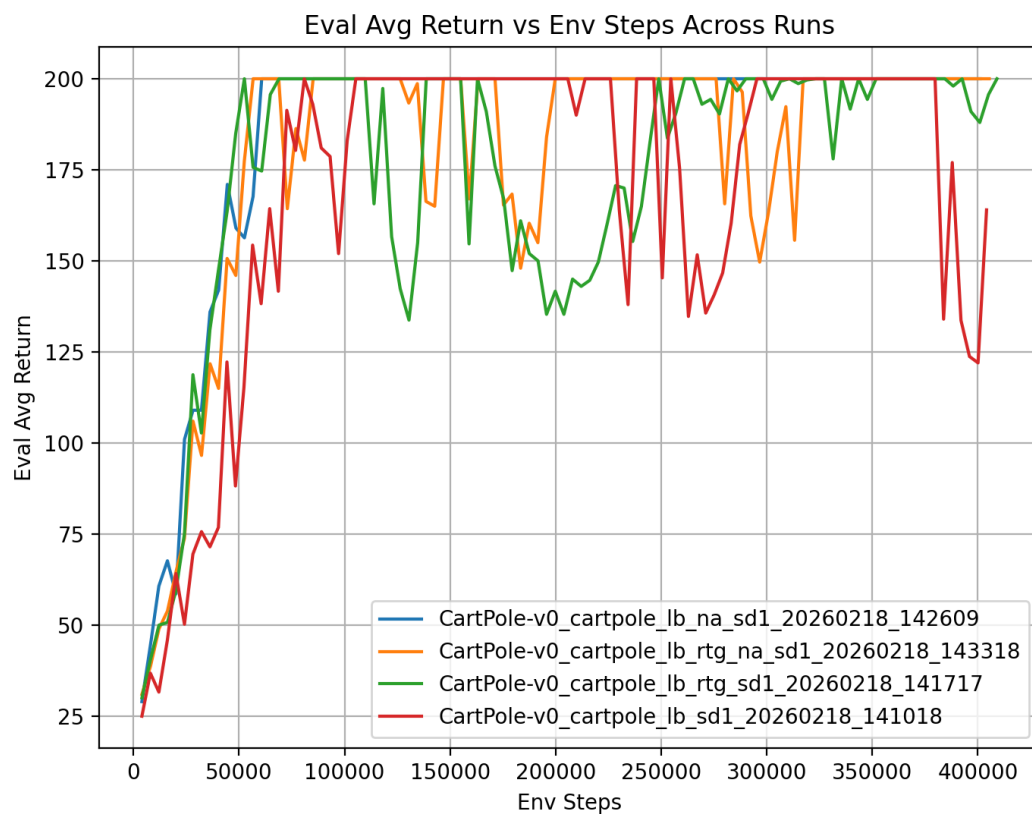
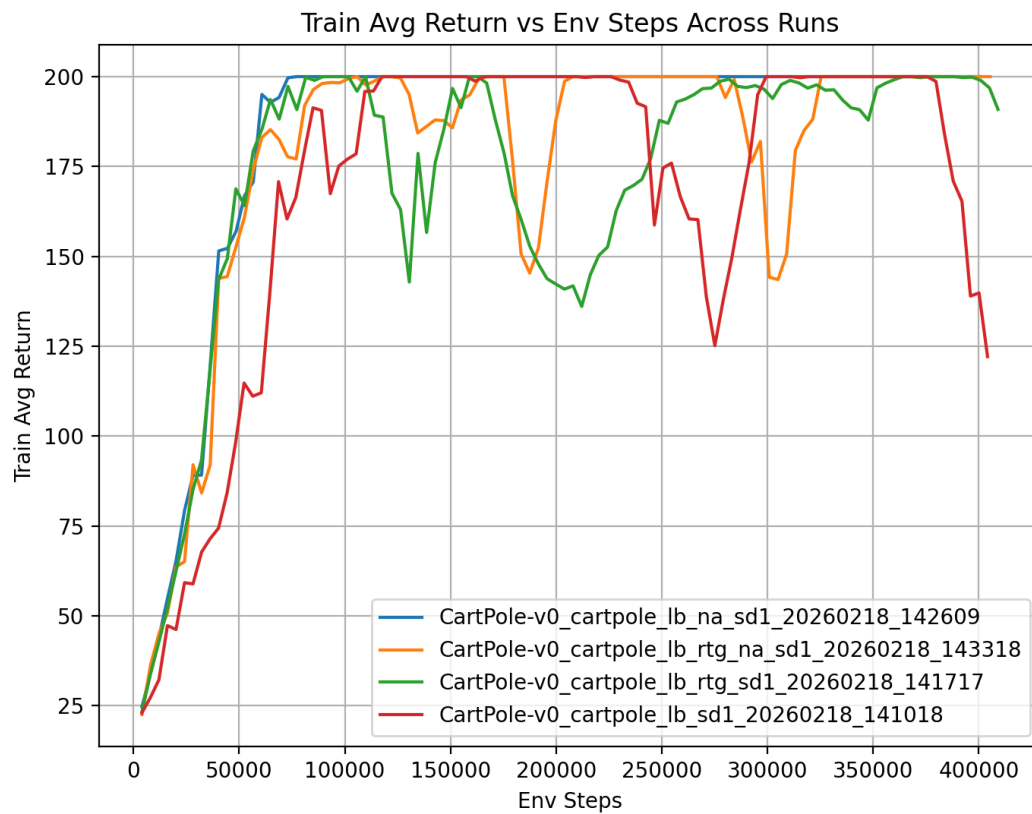
SID: 3041940996

## Question 4

### Small Batch Experiments Learning Curve



## Large Batch Experiment Learning Curves



## **Best Configuration for Small and Large Batch: CartPole-na**

– Which value estimator has better performance without advantage normalization: the trajectory centric one, or the one using reward-to-go?

**For the small batch experiments, the trajectory centric value estimator seems to perform better as it converges closer to the max return of 200.**

**For large batch-experiments, the reward-to-go value estimator, performs better.**

**Theoretically, this might be because rtg leads to unstable policy gradient updates as the return term changes at each timestep when using rtg. This would be more significant in small batch experiments as there are less rollouts for the unstable policy gradient update to be averaged over.**

– Between the two value estimators, why do you think one is generally preferred over the other?

**The reward-to-go value estimator would be more preferred as it removes variance due to irrelevant rewards and better estimates the return of an action by only considering current and potential future rewards.**

– Did advantage normalization help?

**Yes. Generally, advantage normalization stabilized the policy gradient updates and lead to earlier convergence to the max reward of 200.**

– Did the batch size make an impact?

**It affected the effect of rtg but with regard to the effectiveness of the return estimator, the larger batch size led to convergence in more steps. Though this might be because the large batch policy performs gradient updates less frequently.**

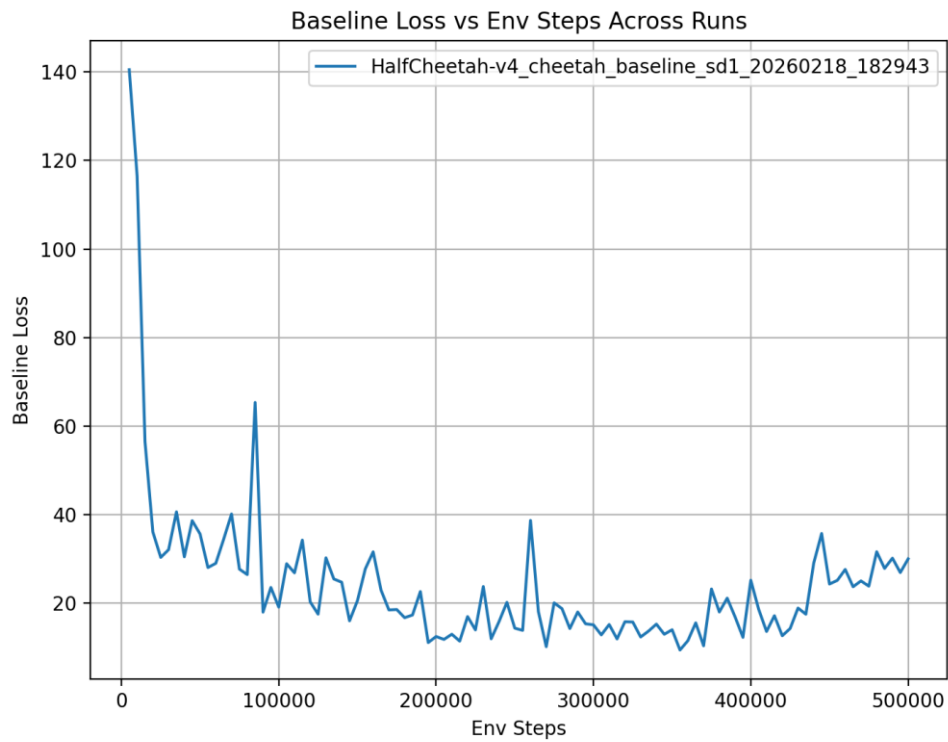
## **Command-line Configurations**

1. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 1000 \`  
`--exp_name cartpole`
2. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 1000 \`  
`-rtg --exp_name cartpole_rtg`
3. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 1000 \`  
`-na --exp_name cartpole_na`
4. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 1000 \`  
`-rtg -na --exp_name cartpole_rtg_na`
5. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 4000 \`  
`--exp_name cartpole_lb`
6. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 4000 \`  
`-rtg --exp_name cartpole_lb_rtg`
7. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 4000 \`  
`-na --exp_name cartpole_lb_na`
8. `uv run src/scripts/run.py --env_name CartPole-v0 -n 100 -b 4000 \`  
`-rtg -na --exp_name cartpole_lb_rtg_na`

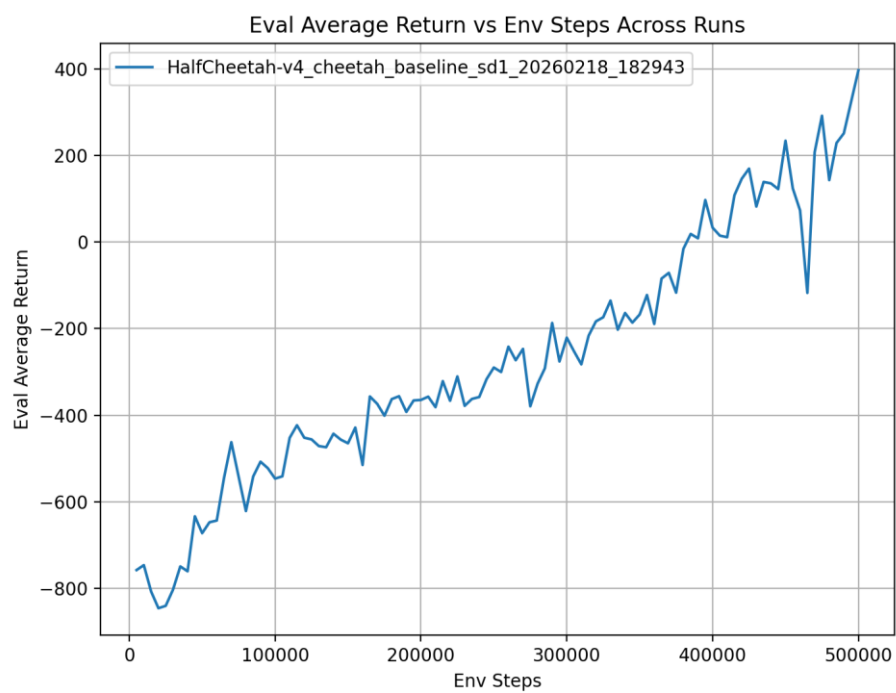
## Question 5

### Vanilla Baseline Experiment

#### Baseline Loss Learning Curve

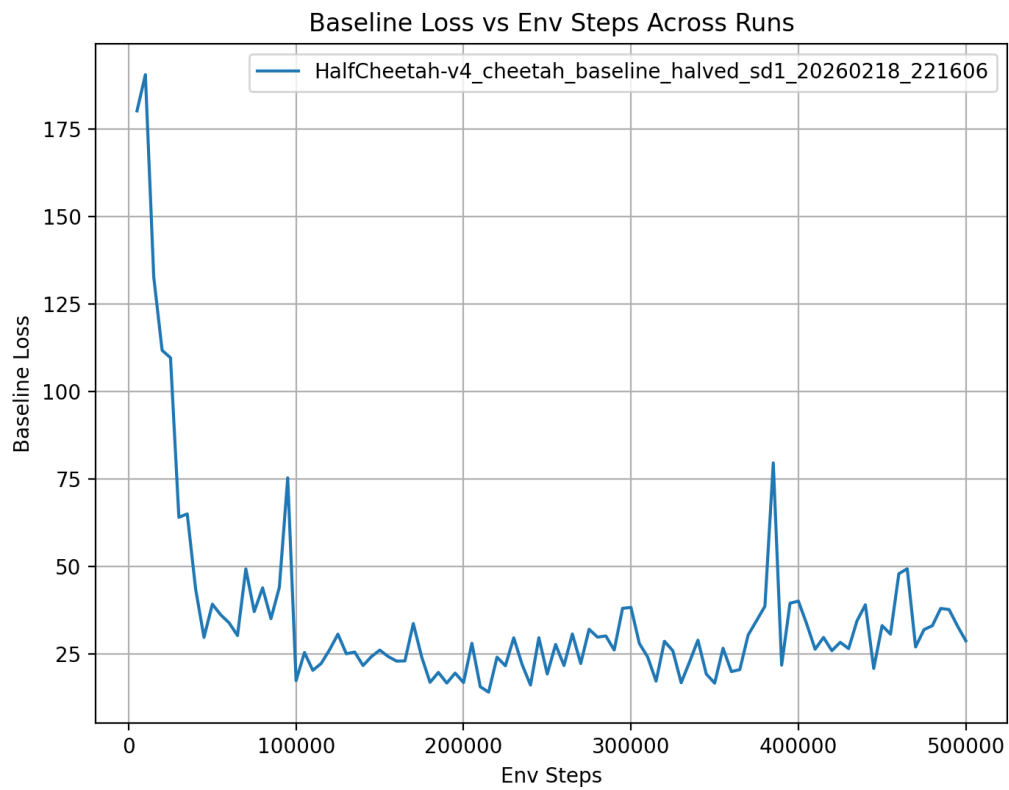


#### Evaluation Reward Learning Curve



## Halved Baseline Learning Rate Experiment

### Baseline Loss Learning Curve

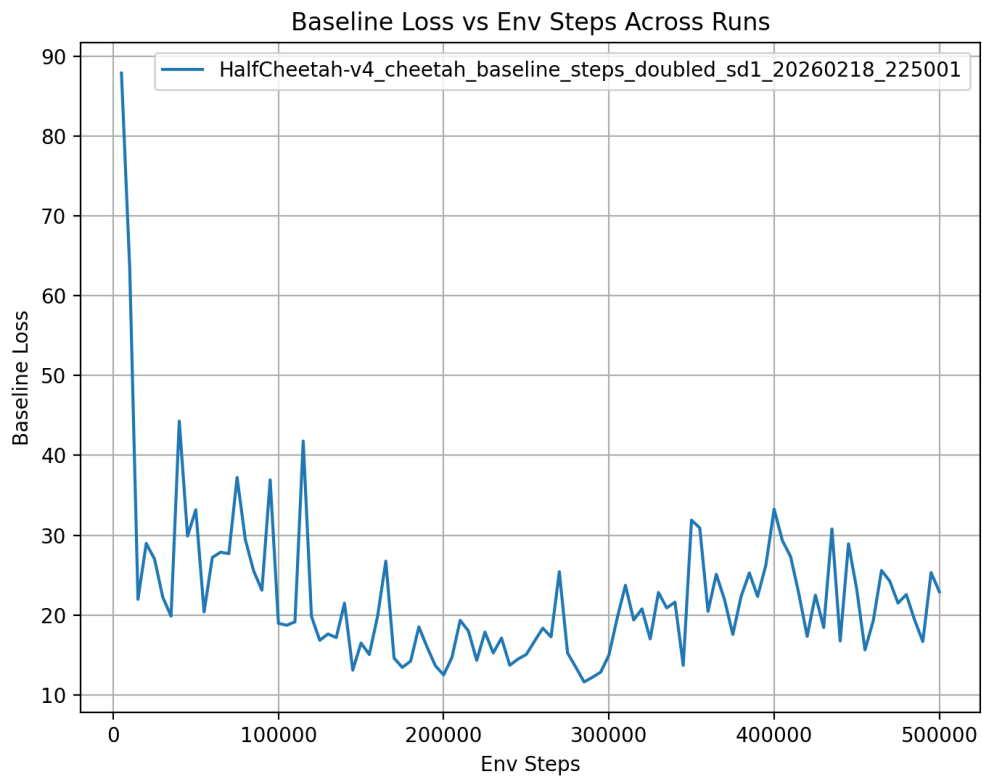


### Evaluation Reward Learning Curve

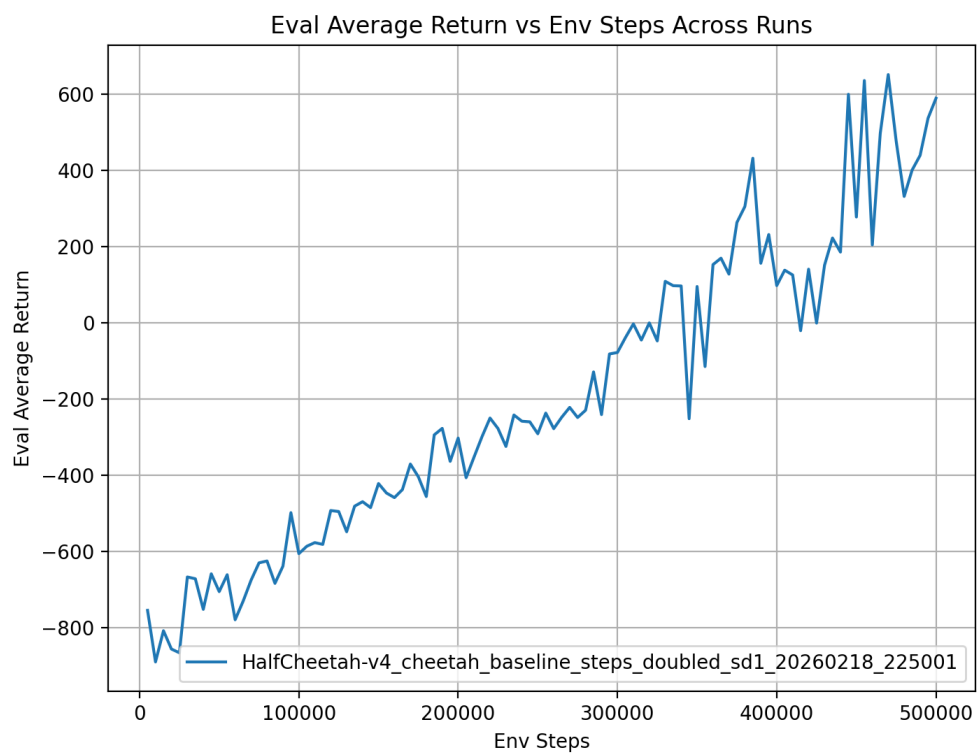


## Doubled Baseline Gradient Steps Experiment

### Baseline Loss Learning Curve



### Evaluation Reward Learning Curve



## **Command-line Configurations**

# No baseline

1. `uv run src/scripts/run.py --env_name HalfCheetah-v4 -n 100 -b 5000 -eb 3000 -rtg --discount 0.95 -lr 0.01 --exp_name cheetah`

# Baseline

2. `uv run src/scripts/run.py --env_name HalfCheetah-v4 -n 100 -b 5000 -eb 3000 -rtg --discount 0.95 -lr 0.01 --use_baseline -blr 0.01 -bgs 5 --exp_name cheetah_baseline`

## **Lowered Baseline Learning Rate to 0.005**

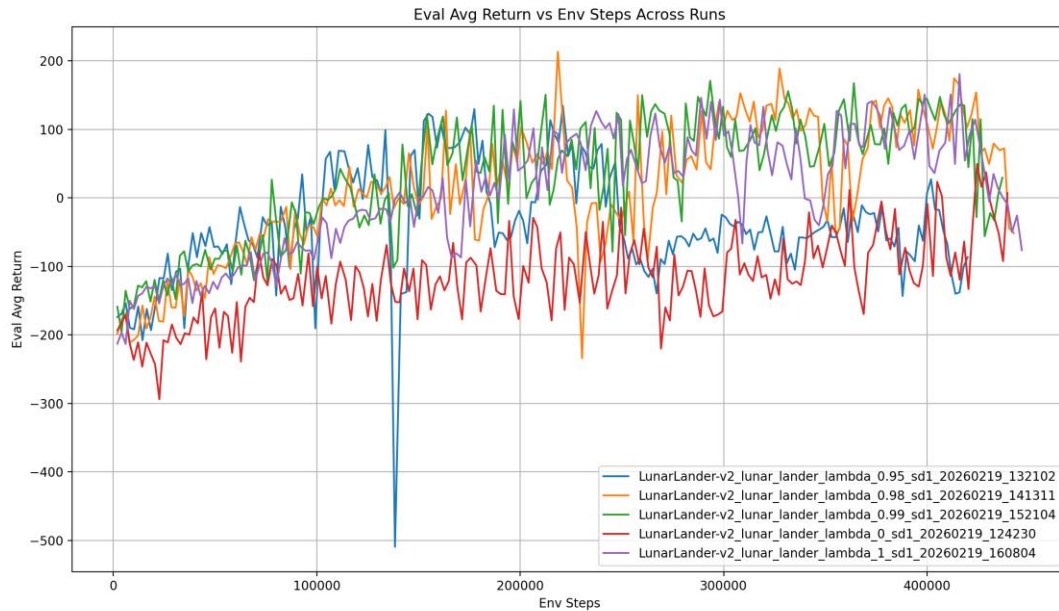
3. `uv run src/scripts/run.py --env_name HalfCheetah-v4 -n 100 -b 5000 -eb 3000 -rtg --discount 0.95 -lr 0.01 --use_baseline -blr 0.005 -bgs 5 --exp_name cheetah_baseline_halved`

## **Doubled Baseline Gradient Steps**

4. `uv run src/scripts/run.py --env_name HalfCheetah-v4 -n 100 -b 5000 -eb 3000 -rtg --discount 0.95 -lr 0.01 --use_baseline -blr 0.01 -bgs 10 --exp_name cheetah_baseline_steps_doubled`

## Question 5

- Provide a single plot with the learning curves for the LunarLander-v2 experiments that you tried. Describe in words how  $\lambda$  affected task performance.



Increasing lambda generally lead to a higher average return. This can be seen by the curve for lambda = 0 having the lowest Eval\_AverageReturn. This correlation might be due to the finetuning of lambda leading to a better estimation of advantage for a certain state,action which would allow the policy to weigh actions more accurately.

- Consider the parameter  $\lambda$ . What does  $\lambda = 0$  correspond to? What about  $\lambda = 1$ ? Relate this to the task performance in LunarLander-v2 in one or two sentences.

$$A^\pi(s_t, a_t) \approx \delta_t = r(s_t, a_t) + \gamma V_\varphi^\pi(s_{t+1}) - V_\varphi^\pi(s_t),$$

$$A_{\text{GAE}}^\pi(s_t, a_t) = \sum_{t'=t}^{H-1} (\gamma\lambda)^{t'-t} \delta_{t'},$$

$\lambda$  adjusts the emphasis placed on advantage estimates derived from multiple  $n$ -step Monte Carlo returns depending on the value of  $n$ . A higher  $\lambda$  implies more emphasis placed on  $n$ -step Monte Carlo returns with a greater  $n$ .

$\lambda = 0$  corresponds to the case where the advantage estimate corresponds to the 1-step Temporal Difference (TD) advantage estimator. This maximises Bias as the return is solely dependent on the bootstrapped return output by the value function prediction for the rollout beyond state  $t$ . This also minimises Variance since the value function reduces the noise from random rewards by replacing it with a learned deterministic function estimate.

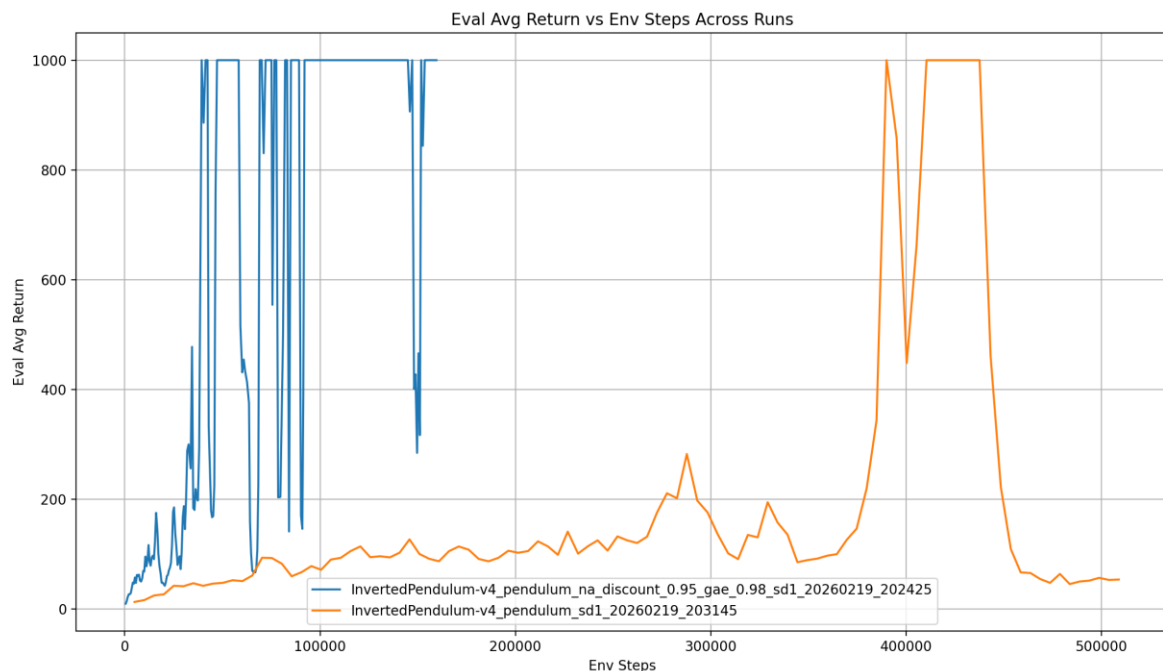
$\lambda = 1$  corresponds to the case where the advantage estimate corresponds to the full Monte Carlo return. This minimises Bias as the return is fully determined from the rollout and maximises Variance as there is less Variance reduction from using the learned value function to predict future returns.

## **Command Line Configurations**

1. `uv run src/scripts/run.py --env_name LunarLander-v2 --ep_len 1000 --discount 0.99 -n 200 -b 2000 -eb 2000 -l 3 -s 128 -lr 0.001 --use_reward_to_go --use_baseline --gae_lambda 0 --exp_name lunar_lander_lambda_0`
2. `uv run src/scripts/run.py --env_name LunarLander-v2 --ep_len 1000 --discount 0.99 -n 200 -b 2000 -eb 2000 -l 3 -s 128 -lr 0.001 --use_reward_to_go --use_baseline --gae_lambda 0.95 --exp_name lunar_lander_lambda_0.95`
3. `uv run src/scripts/run.py --env_name LunarLander-v2 --ep_len 1000 --discount 0.99 -n 200 -b 2000 -eb 2000 -l 3 -s 128 -lr 0.001 --use_reward_to_go --use_baseline --gae_lambda 0.98 --exp_name lunar_lander_lambda_0.98`
4. `uv run src/scripts/run.py --env_name LunarLander-v2 --ep_len 1000 --discount 0.99 -n 200 -b 2000 -eb 2000 -l 3 -s 128 -lr 0.001 --use_reward_to_go --use_baseline --gae_lambda 0.99 --exp_name lunar_lander_lambda_0.99`
5. `uv run src/scripts/run.py --env_name LunarLander-v2 --ep_len 1000 --discount 0.99 -n 200 -b 2000 -eb 2000 -l 3 -s 128 -lr 0.001 --use_reward_to_go --use_baseline --gae_lambda 1 --exp_name lunar_lander_lambda_1`

## Question 6

### Learning Curves



Provide the best set of hyperparameters on InvertedPendulum-v4 and the exact command line configuration. Briefly discuss which hyperparameters mattered in your tuning process.

**I adjusted batch size to be 500 rather than 5000 so that the policy gradients update over a smaller number of accumulated environment steps, even though this increased variance, I believe it improved learning in the policy as it would explore more rapidly and collect more useful trajectory samples with more frequent updates.**

**Using reward-to-go would improve the policy gradient updates by removing noise from earlier, irrelevant rewards.**

**Normalising Advantages helps to reduce variance and improve policy gradient updates**

**GAE lambda helps to improve the associated Advantage with actions, allowing policy to update distribution more accurately.**

**Discount Factor of 0.95 guides policy to focus on short-term rewards and reduces variance of Advantage**

**- n\_iterations = 200**

- batch\_size = 500
- eval\_batch\_size = 1000
- rtg
- gae\_lambda = 0.98
- discount = 0.95

### **Command Line Configuration**

```
uv run src/scripts/run.py --env_name InvertedPendulum-v4 -n 100 -b 5000 -eb 1000 \
--exp_name pendulum
```

```
uv run src/scripts/run.py --env_name InvertedPendulum-v4 -n 200 -b 500 -eb 1000 -
rtg -na --gae_lambda 0.98 --discount 0.95 --exp_name
pendulum_na_discount_0.95_gae_0.98
```