AIL Project Report

1st Yifan Wu
yiw084@ucsd.edu

***Abstract*—The goal of this project is to reproduce and verify the result in "What Matters for Adversarial Imitation Learning?" [1]**

I. Reward Formulation and Bias

Discriminator $D = \sigma(f(s,a)) = \frac{\exp(f(s,a))}{\exp(f(s,a))+\pi(a|s)}$, where f is a discriminator logit (a learnable function represented as MLP).

A. Choice of reward function and its bias

1) Strictly positive reward (introduced in GAIL [2]) worked well for environments that require a survival bonus which encourages longer episodes.

$$r(s,a) = \log(1-D) = softplus(f) \in [0,\infty) \quad (1)$$

2) AIRL [3] flavor reward function is able to assign both positive and negative rewards for each time step. In environments with a survival bonus, this leading to sub-optimal policies (and training instability). Also, it can assign rewards with a negative bias(in the beginning of training). This is common for learned agents to finish an episode earlier. (to avoid additional negative penalty) instead of trying to imitate the expert.

$$r(s,a) = \log D - \log(1-D) = f \in (-\infty,\infty) \quad (2)$$

3) Strictly negative reward often used for tasks with a per step penalty. However, this variant assigns only negative rewards and cannot learn a survival bonus. It encourages shorter episodes.

$$r(s,a) = \log D = softplus(-f) \in (-\infty,0] \quad (3)$$

Notes that the choice of a specific reward function might already provide strong prior knowledge that helps the RL algorithm to move towards recovering the expert policy, irrespective of the quality of the learned reward.

TABLE I
Expert and random policy scores used to normalize the performance for all tasks

Task	Random policy score	Expert score
HalfCheetah-v2	-153	12851
Hopper-v3	21	3701
NavEnv-v0	-914	-43

II. Results

We run our experiments on 2 Mujoco environments, HalfCheetah-v2 and Hopper-v3, and a custom NavEnv-v0.

- HalfCheetah-v2 has true reward in range $[-\infty,\infty]$.
  The goal is to make the cheetah run as fast as possible. Note that there is no done signal in this environment.
- Hopper-v3 has true reward in range $[0,\infty]$.
  It has a similar objective as Halcheetah but with a survival bonus.
- NavEnv-v0 has true reward in range $[-\infty,0]$.
  This is a simple 2D-Navigation environment which has simple dynamics but complex reward function. It encourages to move to the center area as soon as possible to prevent additional penalty.

Expert score and random policy score for each environment are shown in Table 1. All experiment use trained SAC [4] agent as experts.

A. Best hyperparameter values

TABLE II
Best configuration for HalfCheetah-v2

Name	Best value
policy MLP depth	2
policy MLP width	256
critic MLP depth	2
critic MLP width	256
RL activation	ReLu
discount γ	0.99
batch size	256
RL Algorithm	SAC
replay ratio	256
RL replay buffer size	$3 \cdot 10^6$
reward function	AIRL
reward input choice	logsigmoid
absorbing state	False
absorbing reward formulation	-
discriminator input	(s,a)
discriminator MLP depth	1
discriminator MLP width	256
discriminator activation	ReLu
reward shaping	False
subtract log-pi	False
discriminator learning rate	$3 \cdot 10^{-5}$
discriminator reply buffer size	$3 \cdot 10^6$
discriminator to RL updates ratio	1
discriminator regularizer	spectral normalization
entropy	0.03
optimizer	Adam
max grad norm	10

TABLE III
BEST CONFIGURATION FOR HOPPER-V3

Name	Best value
policy MLP depth	2
policy MLP width	256
critic MLP depth	2
critic MLP width	256
RL activation	ReLu
discount γ	0.99
batch size	256
RL Algorithm	SAC
replay ratio	256
RL replay buffer size	$3 \cdot 10^6$
reward function	GAIL
reward input choice	logsigmoid
absorbing state	True
absorbing reward formulation	finite horizon
discriminator input	(s,a)
discriminator MLP depth	1
discriminator MLP width	256
discriminator activation	ReLu
reward shaping	False
subtract log-pi	False
discriminator learning rate	$7 \cdot 10^{-5}$
discriminator reply buffer size	$3 \cdot 10^6$
discriminator to RL updates ratio	1
discriminator regularizer	spectral normalization
entropy	0.03
optimizer	Adam
max grad norm	10

TABLE IV
BEST CONFIGURATION FOR NAVENV-V0

Name	Best value
policy MLP depth	2
policy MLP width	256
critic MLP depth	2
critic MLP width	256
RL activation	ReLu
discount γ	0.99
batch size	256
RL Algorithm	SAC
replay ratio	256
RL replay buffer size	$3 \cdot 10^6$
reward function	Inverse GAIL
reward input choice	logsigmoid
absorbing state	True
absorbing reward formulation	finite horizon
discriminator input	(s,a)
discriminator MLP depth	1
discriminator MLP width	256
discriminator activation	ReLu
reward shaping	False
subtract log-pi	False
discriminator learning rate	$7 \cdot 10^{-5}$
discriminator reply buffer size	$3 \cdot 10^6$
discriminator to RL updates ratio	1
discriminator regularizer	spectral normalization
entropy	0.03
optimizer	Adam
max grad norm	10

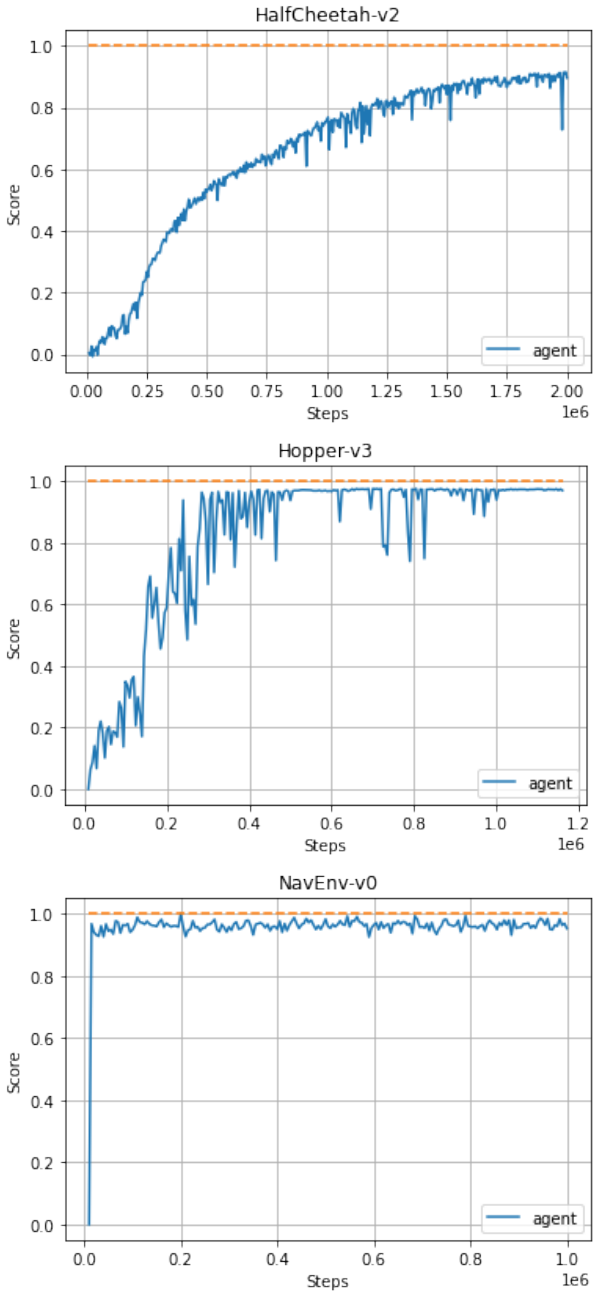

Fig. 1. *Learning curve for AIL agents under 3 environments. These plots shows the averages score of agent across 3 random seeds.*

III. DISCUSSION

In this section, we will discuss what we find about the hyperparameter illustrated above.

A. Discriminator MLP Size

Initially we use two-layer MLP with 256 width in each layer for the discriminator design. We are able to obtain good results but the overall training process is not stable even with the help of different regularization. As we gradually reduce the width and depth of the neural network, we found that smaller

parameters in neural networks is more stable and works better for AIL. Therefore, we suggest using a one-layer design with large hidden size $(256 - 512)$ works best in most of cases. This might also be the reason that dropout are suggested in "What Matters for Adversarial Imitation Learning?" to help on the training. However, dropout significantly decrease the performance in our experiments.

B. Training AIL with off-policy generator

Overall SAC [4] as the state-of-the-arts off-policy RL algorithm outperform on-policy RL algorithm in sample efficiency and achieves a much higher reward in different environment. Instead of sampling trajectories from current policy directly, we sample transitions from a replay buffer R collected, while performing off-policy training. This can be viewed as a mixture of all policy distributions appears during training instead of latest trained policy. In order to recover the original on-policy expectation, one needs to use importance sampling:

$$\max_D E_R\left[\frac{p_{\pi_\theta}(s,a)}{p_R(s,a)}\log(D(s,a))\right] + E_{\pi_E}[log(1-D(s,a))] \tag{4}$$

It is difficult to estimate the importance weight and we found the algorithm works well with the importance weight omitted.

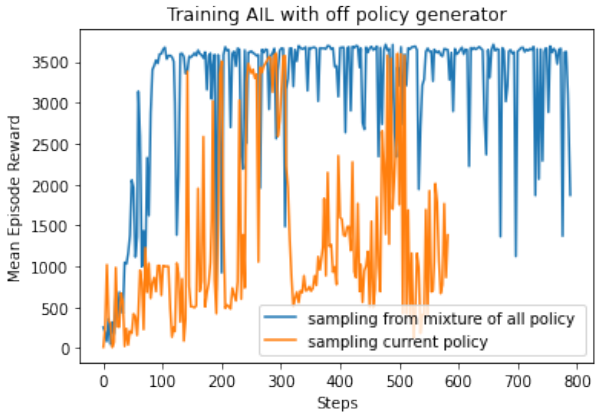

Fig. 2. *Comparison between sampling from current policy and form a mixture of all policy in Hopper*

C. Spectral Normalization

Applying Spectral Normalization [5] to discriminator significantly improve the stability of GAN training. However, there are trade-offs between stability and performance. In some situation, Spectral Normalization can hinder the performance which stuck at a local maxima.

D. Absorbing States

Adding absorbing states is extremely important for Adversarial imitation learning. This idea is introduced in DAC [6]. In their approach, they use TD3 [7] as the generator in GAIL. We modify the algorithm to make it work with SAC as the generator. As mention before, all reward formulation are biased. In order to resolve the issues, we explicitly

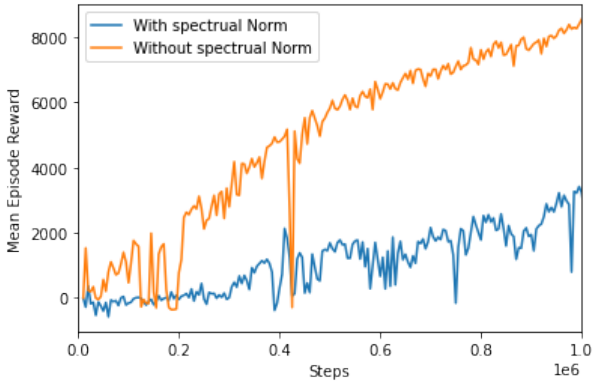

Fig. 3. *Spectral Normalization lead to sub-optimal solution*

learn rewards for absorbing states fro expert demonstrations and trajectories produced by the agent. The return for final states are now $R_T = r(s_T a_t) + \sum_{t=T+1}^{\infty} \gamma^{t-T} r(s_a, \cdot)$, where $r(s_a, \cdot)$ is a learned reward for absorbing state (absorbing reward). The discriminator can distinguish whether reaching an absorbing state is a desirable behavior from the experts' perspective and assign the rewards accordingly. This does not guarantee to cancel the reward bias but significantly moderate the effect. In our experiment, adding absorbing state does not help on HalfCheetah. Since there is no done signal in this environment, we will never encounter absorbing states. When naively adding absorbing state to the trajectory, discriminator will output zero for absorbing reward which is the same as not applying absorbing states. Overall, adding absorbing states with adversarial imitation learning has a faster and more stable convergence.

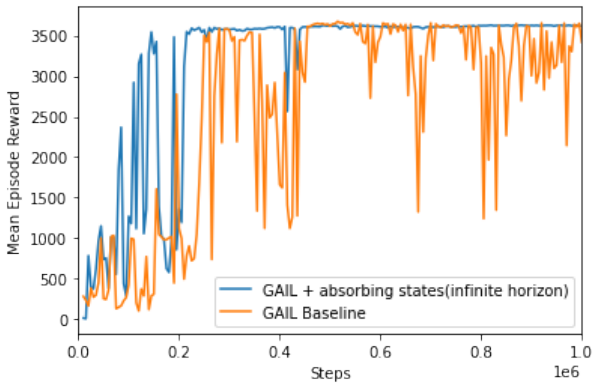

Fig. 4. *Applying absorbing state with GAIL on Hopper*

REFERENCES

[1] M. Orsini, A. Raichuk, L. Hussenot, D. Vincent, R. Dadashi, S. Girgin, M. Geist, O. Bachem, O. Pietquin, and M. Andrychowicz, "What matters for adversarial imitation learning?" *arXiv preprint arXiv:2106.00672*, 2021.

[2] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016. [Online]. Available:

https://proceedings.neurips.cc/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf

[3] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248*, 2017.

[4] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[5] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv preprint arXiv:1802.05957*, 2018.

[6] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, "Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning," *arXiv preprint arXiv:1809.02925*, 2018.

[7] S. Dankwa and W. Zheng, "Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019, pp. 1–5.