Adaptive Model-based Generative Adversarial Imitation Learning

David Orozco[1] Nicholas Ha[2]and Sayan Mondal [3]

***Abstract*— In this paper we introduce a new concept named, AMGAIL to solve the imitation learning problem in an environment in which rewards are available. AMGAIL is based on MGAIL, but it replaces bad expert trajectories with good ones that we generate. We make use of the total rewards of the trajectories to detect how good or bad they are. We tested for 3 MuJoCo environments- Hopper-v1, HalfCheetah-v1, InvertedPendulum-v1. We expect that AMGAIL should perform better than vanilla MGAIL when the expert trajectories are a mix of experts of varying level because the algorithm is able to replace the weaker experts and in turn lower the variance. Our results generally confirm this.**

I. INTRODUCTION

The goal in general for Reinforcement Learning is to speed up learning, because in real world applications, wear and tear on a physical system is an issue. Imitation learning is one possible approach to speed up learning. The goal of imitation learning is to learn a policy by copying an expert, when available expert data only includes state-action pairs (without rewards). Imitation learning is useful when the environment doesn't provide an explicit reward signal, thus only aims at copying expert behavior, regardless of what reward is gained (if there exists any).

We are trying to apply imitation learning for initializing a policy in an environment where reward signals are available, and the goal is to maximize reward. By copying a skilled expert, earlier stages of learning can be greatly accelerated. In particular, we are trying to solve the problem of imitation learning when the set of available expert trajectories has mixed skill levels. It may be desirable for policy robustness to include even the lower performing experts in order to increase dataset size. However these "bad" experts may hinder learning. AMGAIL aims to tackle the challenge by dynamically modifying the list of experts to train from.

II. BACKGROUND

A. Problem formulation

Given a set of expert trajectories in the form of (state,action) pairs, learn a policy that reaches the performance level of the expert (in terms of reward per episode) with as few interactions with the environment as possible.

B. Related Work

1) GAIL: It uses Generative Adversarial Networks (GANs) to imitate an expert in a model-free setup. It is effective, but slow for training stochastic policies because relies on gradient approximations. GANs is a method for training generative models. It uses a second neural network(D) to guide the generative model (G) towards producing patterns similar to those of the expert.

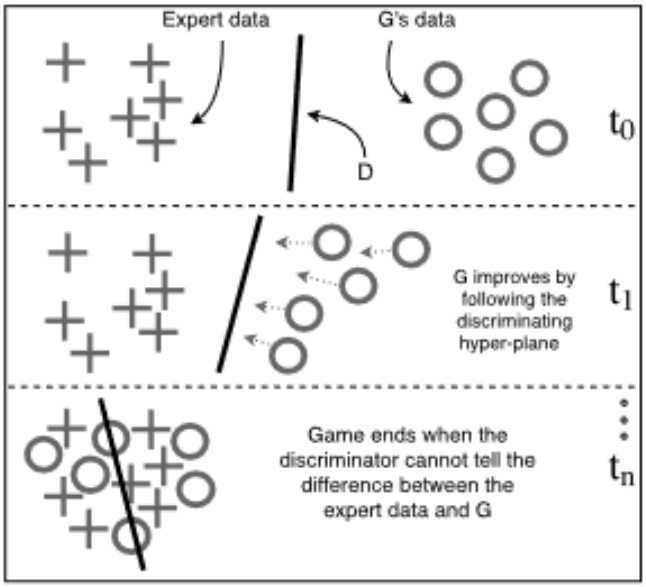

Fig. 1. Applying GAN to Policy Networks

The disadvantage of the model-free approach comes to light when training stochastic policies. The presence of stochastic elements breaks the ow of information (gradients from one neural network to the other,thus prohibiting the use of back propagation). In this situation,a standard solution is to use gradient estimation (Williams, 1992). This tends to suffer from high variance, resulting in a need for larger sample sizes as well as variance reduction methods.

2) MGAIL: This predecessor uses a forward model to make the computation fully differentiable, which enables training policies using the exact gradient of the discriminator. The resulting algorithm trains competent policies using relatively fewer expert samples and interactions with the environment than GAIL. Our AMGAIL algorithm is an extension to this algorithm and attempts to solve the problem of what to do with high variance in expert trajectory quality. Since MGAIL uses every trajectory that is in the expert buffer, it will imitate every expert, even if the expert is of questionable quality and therefore should take longer to train.

3) InfoGail: Recent work on adversarial learning has adopted a different approach by learning semantically meaningful factors of variation in the data. InfoGail aims to capture the latent structure underlying expert demonstrations in an unsupervised way. Although this algorithm aims to solve the problem of variability in expert demonstrations, This algorithm does not use any reward signals to determine the variations in the data, or determine the quality of the

expert. InfoGail does propose a way of "reward augmentation" to provide additional incentives to the agent using external rewards without interfering with imitation learning. However, the results of this were never explored the original paper. They proposed using a surrogate state-based reward

$$\zeta(\pi_\theta) = E_{s\sim\pi_\theta}[r(s)]$$

that reflects the bias over the desired agents behavior, optimization for InfoGail is as follows.

$$\min_{\theta,\psi}\max_{\omega}\mathbb{E}_{\pi_\theta}[\log D_\omega(s,a)] + \mathbb{E}_{\pi_E}[\log(1-D_\omega(s,a))] - \lambda_0\eta(\pi_\theta) - \lambda_1 L_I(\pi_\theta, Q_\psi) - \lambda_2 H(\pi_\theta)$$

It can be seen that although InfoGail proposed a way of augmenting the data using external rewards, the method still uses imitation learning on all expert data, and does not use the rewards as a way to remove or eliminate expert trajectories, unlike AMGAIL.

III. METHODS

We start this section by analyzing the characteristics of the discriminator. Then, we explain how the forward model proposed by our predecessor (MGAIL) alleviates problems that arise when using GANs for imitation learning. Last, we present the changes made for implementing our Adaptive model-based adversarial imitation algorithm (AMGAIL).

A. Discriminator

The discriminator network is trained to predict the conditional distribution: $D(s,a) = p(y|s,a)$ where y $\varepsilon\{\pi_E, \pi\}$. This conditional distribution represents the likelihood ratio that a given state-action pair was generated by our generative network π rather than by an expert π_E. Using Bayes rule and the law of total probability we can simplify the joint distribution into the following form.

$$D(s,a) = \frac{p(s,a|\pi)}{p(s,a|\pi)+p(s,a|\pi_E)} = \frac{1}{1+\frac{p(s,a|\pi_E)}{p(s,a|\pi)}}$$

$$= \frac{1}{1+\frac{p(a|s,\pi_E)}{p(a|s,\pi)}\frac{p(s|\pi_E)}{p(s|\pi)}}$$

The last substitution is an approximation for AMGAIL because the size of the expert buffer and generator buffer is kept to almost the same length +/- one episode. Thus the probability of the trajectory coming from the expert or generator is on average approximately 1/2, rather than being enforced such as in MGAIL. By defining

$$\Phi(s,a) = \frac{p(a|s,\pi_E)}{p(a|s,\pi)}, \Psi(s) = \frac{p(s|\pi_E)}{p(s|\pi)}$$

and upon further inspection we see that $\Phi(s,a)$ represents the likelihood ratio that a sample came from expert rather than the generator. Additionally we see that $\Psi(s)$ represents the state likelihood ratio, i.e. the likelihood that the state belongs to an expert trajectory rather than a generated trajectory. At a high level, the prior deals with how likely the action was taken by an expert, while the latter expresses how likely it is the expert would have encountered that trajectory.

B. MGAIL approach to GAN

1) Stochastic Unit Approximation : For the continuous environments used by MGAIL and AMGAIL we use the reparameterization trick to back propagate through the stochastic elements of this algorithm. MGAIL and AMGAIL both do this in the following way; first assume a stochastic policy with a Gaussian distribution, where the mean and variance are given by some deterministic functions $\mu_\theta and \sigma_\theta$, respectively: $\pi_\theta(a|s) \sim N(\mu_\sigma(s), \sigma^2_\theta(s))$. This is then rewritten as π as $\pi_\theta(a|s) = \mu_\theta(s) + \zeta\sigma_\theta(s)$, where $\zeta \sim N(0,1)$. By doing this, MGAIL was able to use the monte-carlo estimator of the derivative which is defined in MGAIL as follows:

$$\nabla_\theta E_{\pi(a|s)} D(s,a) = E_{\rho(\zeta)} \nabla_a D(a,s) \nabla_\theta \ \pi_\theta(a|s) \approx$$

$$\sum_{i=1}^{M} \nabla_a D(s,a) \nabla_\theta \pi_\theta(a|s) \big\|_{\zeta=\zeta_i}$$

2) MGAIL forward model: The main contribution for MGAIL was the use of its forward model, hence, model-based GAIL. Since originally we treated the environment dynamics as a black box we do not know how actions taken will affect future states. In order to back propagate error through entire episodes, a differentiable model mapping state-actions to future states is needed. Thus, the forward model in MGAIL bridges the gap over the "black box" dynamics of the environment. This is visually demonstrated in figure 2 below.

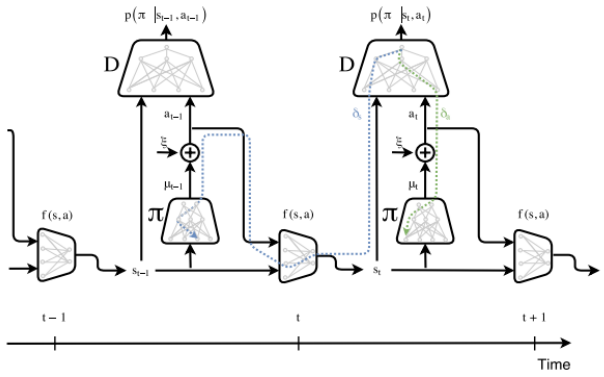

Fig. 2. Block diagram of model-based adversarial imitation learning.

$$J(\theta) = E[\sum_{t=0} \gamma^t D(s_t, a_t)|\theta].$$

In order to differentiate $J(\theta)$ over a trajectory of (s,a,s_0) transitions, MGAIL and AMGAIL rely on the results of Heess [5]:

$$J_s = \mathbb{E}_{p(a|s)}\mathbb{E}_{p(s'|s,a)}\left[D_s + D_a\pi_s + \gamma J'_{s'}(f_s + f_a\pi_s)\right]$$

$$J_\theta = \mathbb{E}_{p(a|s)}\mathbb{E}_{p(s'|s,a)}\left[D_a\pi_\theta + \gamma(J'_{s'}f_a\pi_\theta + J'_\theta)\right]$$

Fig. 3. Calculated Derivatives for Forward Model

The final policy gradient $\nabla_\theta J$ is calculated by applying Eq. 12 and 13 recursively, starting from $t = T$ all the way down to $t = 0$.

C. AMGAIL Algorithm & Pseudo Code

1) Buffer Replacement & Memory Insertion: The main contribution of AMGAIL is the use of a dynamically changing buffer and the algorithm's use of a rating system to replace trajectories in the expert buffer with experiences from the generated buffer. We begin by using the same algorithm as MGAIL, we then replace the fixed buffer size of 50,0000, with a buffer which is allowed to change in size, the method for replacing memories is as follows: Start at the current position in the buffer at the end of the last entered memory. If adding the current episode makes the total length of the buffer greater than 50,000 iterations remove the next entire episode in the buffer, otherwise simply append the entire episode. If the buffer is still larger than 50,000 iterations, remove the next entire episode in the buffer and continue until the buffer is less than 50,000.

In order to add memories into the expert trajectories there is a slight modification to the add method. Before we add to the buffer, we first sort the expert buffer episodes by the total reward of each episode, thus placing the worst trajectories at the beginning. Before adding any trajectory into the buffer we set the location of the last entered location to be the start. The rest of the replace algorithm is as the same as above.

2) Reward Rating Methods: In the case that we have an explicit reward signal from the environment, i.e. the reward produced for a given state-action pair by the environment is a defined function, we first calculate the direct reward received for every state-action pair in the buffer. Once each pair has an associated reward signal we group every trajectory by episode, which is then used to calculate the total reward for the entire episode. This scoring process is only done once every 15 iterations, which was determined empirically through experimentation. The reason that the total episodic reward is used to rate each step in the trajectory rather than it's instant reward is so that our algorithm can take long term rewards into account. In other words, the quality of an expert action is determined by overall how well the expert did, and not how it did in the very short term. This should overcome a bias of choosing actions that produce high rewards in the short term, which is particularly important for environments with sparse delayed reward signals. One set back of this rating system is that we must be able to generate an associated reward for every state-action pair, which may be a problem for some cases of imitation learning. This problem is out of the scope of this paper because we aim to assist imitation learning problems in which taking advantage of an available reward signal could speed up learning. However, a future addition to this paper would be to create a network that would recover these reward signals similar to the inverse reinforcement learning problem.

3) Pseudo Code: The figure below shows a comparison of the vanilla MGAIL algorithm and the AMGAIL additions (shown in blue).

Algorithm 1 Model-based Generative Adversarial Imitation Learning

Input: Expert trajectories τ_E, experience buffer $\mathcal{B}$, initial policy and discriminator parameters θ_g, θ_d
for $trajectory = 0$ **to** ∞ **do**
  for $t = 0$ **to** T **do**
    Act on environment: $a = \pi(s, \xi; \theta_g)$
    Push (s, a, s') into $\mathcal{B}$
  end for
  train forward model f using $\mathcal{B}$
  train discriminator model D_{θ_d} using $\mathcal{B}$
  set: $j'_s = 0, j'_{\theta_g} = 0$
  for $t = T$ **down to** 0 **do**
    $j_{\theta_g} = [D_a\pi_{\theta_g} + \gamma(j'_{s'}f_a\pi_{\theta_g} + j'_{\theta_g})]\big|_\xi$
    $j_s = [D_s + D_a\pi_s + \gamma j'_{s'}(f_s + f_a\pi_{\theta_g})]\big|_\xi$
  end for
  Apply gradient update using $j^0_{\theta_g}$
  Calculate total rewards for expert and agent buffer.
  Rate each trajectory based on total rewards per episode.
  If max(Rating(B)) > min(Rating(T)):
    Replace min(T) with max(B)
end for

Vanilla MGAIL

AMGAIL

Fig. 4. AMGAIL vs MGAIL Pseudo Code Comparison

IV. RESULTS

We analyze performance between MGAIL and AMGAIL each with 50,000 expert samples, on the environments InvertedPendulum, HalfCheetah, and Hopper. In addition we do these tests on 3 different expert datasets that name Tier 1, 2 and 3. Tier 1 contains only the best experts, Tier 3 contains only the worst, and Tier 2 has an equal combination of both. The histograms of expert datasets for each environment are shown in figure 5. In addition, for tier 2 expert datasets, we also compare MGAIL and AMGAIL to "1/2MGAIL" which uses only the best 25000 expert samples from Tier 2. We suspect that 1/2MGAIL may be less robust since the dataset to learn from is smaller, but it may be faster since it doesn't have bad trajectories to slow it down.

In figure 6 we see a typical learning pattern for a single trial of MGAIL and AMGAIL for each environment with different Tier expert sets. It is evident that AMGAIL and MGAIL do similarly for Tier 1 experts, and while AMGAIL typically learns quicker for Tiers 2 and 3 as expected. For the Hopper environment, the rewards seem to have very high variance and it's hard to see whether AMGAIL helps. Note the "iterations" refers to the number of iterations of the algorithm. Training on different networks are done every iteration, but an episode of MGAIL/AMGAIL is run only every 15 iterations.

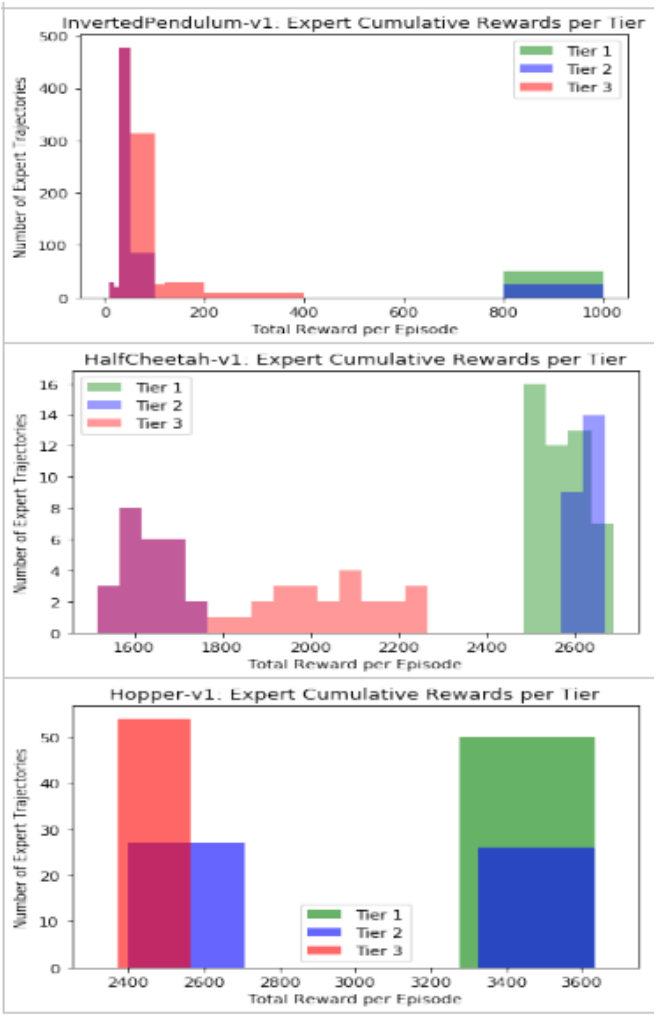

Fig. 5. Histogram rewards per trajectory for Expert Tiers

We also do a more detailed comparison between the algorithms, now including 1/2 MGAIL for Tier 2. In these tests, we run each algorithm many times and record the number of times it took solve the environment. During each run of MGAIL/AMGAIL, every ”test interval” we check the average reward over 5 episodes. If the average is above a specified termination condition, we consider that environment solved and record the number of iterations it took. We can see that for most environments and Tiers AMGAIL either converges faster or about the same as MGAIL on average. In an external note for Tier 3 Inverted Pendulum we see that there is a clear advantage of AMGAIL over MGAIL. However, we see that typically AMGAIL and MGAIL have a very large standard deviation and thus the results are not as concrete to give a definitive answer which algorithm performs better. The number of runs n, average, and standard deviation for each algorithm, expert dataset, and environment are shown in table 7.Parameters for each environment are shown in Table 8. In addition histograms of learning time are shown in figs 9 to 11.

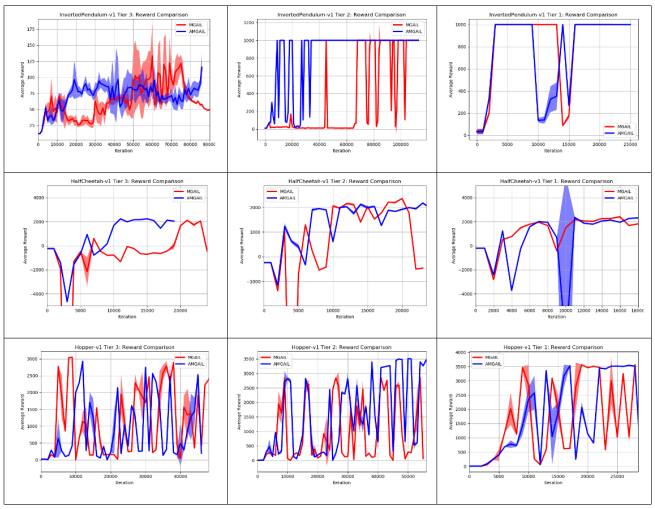

Fig. 6. Single run of MGAIL and AMGAIL for each tier and each environment. Each column is a different tier, and each row is a different environment

ITERATIONS UNTIL CONVERGENCE

Tier 1

	Inverted Pendulum		Half Cheetah		Hopper	
	MGAIL	AMGAIL	MGAIL	AMGAIL	MGAIL	AMGAIL
n	20	20	19	16	18	18
ave	2280	**2185**	**10500**	10666	**9200**	9333
std	518	513	1886	1618	2195	1945

Tier 2

	Inverted Pendulum			Half Cheetah			Hopper		
	MGAIL	AMGAIL	½ MGAIL	MGAIL	AMGAIL	½ MGAIL	MGAIL	AMGAIL	½ MGAIL
n	21	21	16	18	17	18	16	17	18
ave	43500	10400	**2475**	14200	17400	**10900**	8800	**7600**	9011
std	27700	3600	1090	5800	12100	1700	1700	1600	1640

Tier 3

	Inverted Pendulum		Half Cheetah		Hopper	
	MGAIL	AMGAIL	MGAIL	AMGAIL	MGAIL	AMGAIL
n	6	6	12	12	15	16
ave	*		101400	**57100**	**9000**	9600
std			111800	100800	2800	4100

Fig. 7. Average number of iterations until convergence. *for Inverted Pendulum with bad experts, data could not be collected well because convergence was rarely reached. After 6 trials, MGAIL only converged once after 440,000 iterations. AMGAIL converged in 80,000,215,000, and 351,000 iterations, but didn't converge the 3 other times. For display the the histogram, non-convergence is represented 500,000 iterations

TEST PARAMETERS:

	Inverted Pendulum	Half Cheetah	Hopper
Test interval	100	400,500	400
Term condition	990	2000	2500

Fig. 8. Parameters used to determine iterations until convergence

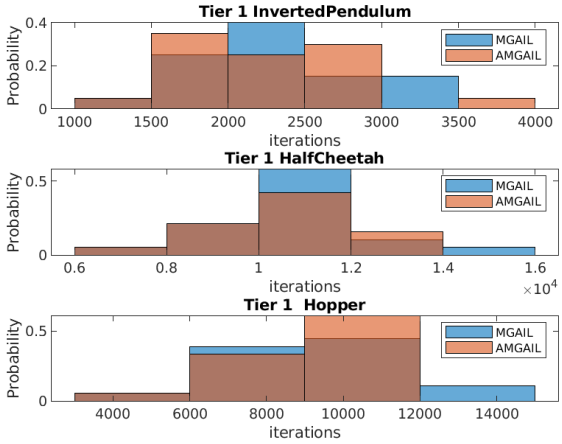

Fig. 9. Histogram for iterations until convergence for Tier 1

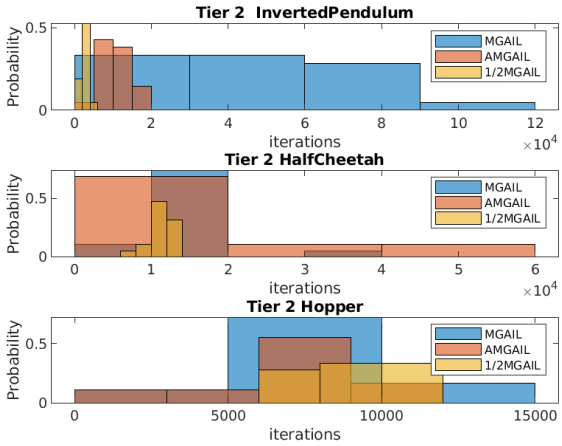

Fig. 10. Histogram for iterations until convergence for Tier 2

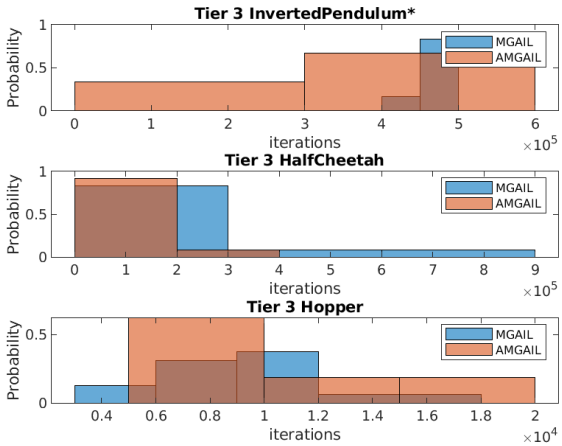

Fig. 11. Histogram for iterations until convergence for Tier 3

By comparing the histograms and tables, we can see that MGAIL and AMGAIL have similar performance for Tier 1 experts. In Tier 2, AMGAIL typically does better, but there is sometimes higher variance. AMGAIL is better in Tier 3, especially for Inverted Pendulum. 1/2 MGAIL is very good with Tier 2.

One explanation that 1/2MGAIL tends to do the best is the fact that it has very low variance and all of the experts are only from Tier 1, in this scenario we expect that this is the result we would find. Since the expert trajectory is already ”perfect” (i.e. no variance) AMGAIL has little to no advantage over MGAIL. This also reinforces our idea that a weakness of MGAIL is that it performs worse with high variance in expert trajectories, and thus when we remove this barrier we see it performs better.

V. CONCLUSIONS

In general, AMGAIL shows an improvement over vanilla MGAIL in nearly all environments where the expert trajectory sets that are high in variance and lower in performance. The tier 2 expert trajectories show that variance in expert trajectories shows a drop in performance for vanilla MGAIL while Adaptive MGAIL filters tends to filter out the poor performers earlier on, thus leading to faster improvement. The tier 3 expert trajectories show that AMGAIL is able to slightly improve in performance after it has managed to perform as good as the experts, while vanilla MGAIL cannot become better than the experts it was trained on and is hard capped to only perform as good as the experts. In the case where the expert trajectories are of nearly perfect quality as in tier 1, we see that AMGAIL performs about the same as vanilla MGAIL. This suggests that AMGAIL could be used as an extension to MGAIL as it may only improve performance but not inhibit it. By comparing MGAIL's performance on tier 2 experts as opposed to the 1/2MGAIL experiment, the evidence reinforces our idea that the high variance in our expert data is what causes a drop in performance for MGAIL rather than a smaller set of tier 1 expert trajectories.

REFERENCES

[1] Generative Adversarial Imitation Learning https://arxiv.org/pdf/1606.03476.pdf
[2] End-to-End Differentiable Adversarial Imitation Learning, http://proceedings.mlr.press/v70/baram17a/baram17a.pdf, https://github.com/itaicaspi/mgail
[3] InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations, http://people.csail.mit.edu/liyunzhu/posters/infogail-poster.pdf
[4] Trust Region Policy Optimization with TensorFlow and OpenAI Gym, https://github.com/pat-coady/trpo
[5] Learning Continuous Control Policies by Stochastic Value Gradients, https://arxiv.org/pdf/1510.09142.pdf