Showcasing Imitation Learning using Bird Hunt Game

Bilal Siddique*, Nishant Gajjar†, Enosh Peter Ponraj‡ and Ayush Sharma§ Email :
{*bisi00001,†s8nigajj,‡enpe00001,§aysh00001}@stud.uni-saarland.de
Matriculation Number: *7015604, †2577584, ‡7012171, §7017384
Universität des Saarlandes, Saarbrücken

Abstract—Reinforcement Learning (RL) is a Machine learning algorithm that learns from interaction with its environment. But it brute forces its way to find the best policies. In order to help the agent to learn this faster, we use Imitation Learning (IL). In this project, we provide a comparison between different environment complexities with different RL and IL algorithms to showcase when IL is effective and when it is not. We have used the Unity Bird Hunt game environment provided by CodeMonkey [Codemonkey(2022)] as the baseline environment for all our experiments.

Keywords: Unity, Reinforcement Learning, Imitation Learning, Inverse Reinforcement Learning, Bird Hunt

1 Introduction

Reinforcement learning [Sutton and Barto(2018)] has been a promising machine learning method that has risen in popularity in the field of Autonomous Systems. These methods are becoming progressively more effective at optimizing reward functions in complex environments. However, it is the difficult to shape fitting reward functions for intricate tasks and incorporate all facets of it [Hadfield-Menell et al.(2017)]. Humans are habitually able to learn complex tasks much faster by observing and imitating the behavior of others. Similarly, autonomous agents have been capable enough to use the same notion, referred to as learning from demonstrations (LfD) [Argall et al.(2009)] and tackle the aforementioned challenges by the use of methods involving Imitation learning via expert demonstrations [Finn et al.(2016)] where the agent promptly learns desirable actions by imitating an expert. Behavioral cloning (BC) [Ross et al.(2013)] and Inverse RL are the primary methods that are used to deal with an imitation learning problem where the agent has access to both states and actions of the demonstration [Ho and Ermon(2016)].

We will be showcasing the performance, robustness and limitations of imitation learning when compared to traditional reinforcement learning. We have built on a pre-existing environment and have illustrated our findings. Motivation was to experiment how well an expert and sub optimal demo affects the performance with respect to traditional Reinforcement learning in different environment complexities. The base model is first evaluated to a model with color channels by comparing the performance on policy optimization method - Proximal Policy Approximation (PPO) [Schulman et al.(2017)] and Soft-Actor Critic (SAC) [Haarnoja et al.(2018)]. PPO is an on-policy algorithm that encourages exploration over exploitation whereas SAC is an off-policy method that balances between exploration and exploitation.

The rest of the paper is structured as follows, in section 2 we discuss previous work related to our procedure, section 3 discusses the complexities and the approach towards evaluation, in section 4 talk about our findings, in section 5 we review the challenges that we faced and at last in section 6 we put forward the general conclusion along with the related discussion.

2 Related Work

Advance research has been carried out in the field of Reinforcement Learning and Imitation Learning. [Torabi et al.(2018)] showcased an advanced adaptation of behavioral cloning called Behavioral Cloning from Observation where the agent only has access to demonstration states and not the demonstration actions. In [Taylor(2018)], the author provides a few methods in which the agent can best leverage the knowledge from a sub-optimal human to learn near-optimal or optimal behaviors. The methods introduced are Learning from Human Demonstrations and Learning from Human Feedback. But for our experiment, we require only the first method which is learning from human demonstrations. Since LfD methods always aim to mimic the demonstration which makes a requirement for expert demonstration, the paper aims to leverage the model from sub-optimal demonstrations. The techniques proposed are *Human-Agent Transfer* which uses the demonstration to form a classifier which is used to further boost the exploration phase and *Shaping IRL* which makes use of the environment rewards i.e., potential-based reward shaping. But in our experiments, since we have the expert demonstrations, we compare the models formed using expert demonstration. In [Verma et al.(2021)], the author provided how the agent can use the performance benefits of traditional reinforcement learning along with the reliability and accountability there without compromising scope and performance. More briefly, they come up with an meta algorithm called *Propel(Imitation-Projected Programmatic Reinforcement Learning)*.

InfoGAIL (Information maximization Generative Adversary Imitation Learning) [Li et al.(2017)] algorithm follows in the footsteps of its predecessor Generative Adversary Imitation Learning (GAIL) [Ho and Ermon(2016)]. This imitation learning algorithm is capable of inferring the latent structure of expert demonstrations in an unsupervised way. This provides the means of interpreting and representing different complexities of behavioral data in a meaningful way. InfoGAIL follows a similar structure to an Information maximization Generative Adversary Network (InfoGAN) [Chen et al.(2016)] to recover semantically meaningful factors of variation in the data.

3 METHODOLOGY

Due to the analytical nature of our project, we break down our approach into miniature complexities added to the base environment in a cascading way, such that each added complexity introduces its own parameters and analytical corners on top of the one before it. Different model structures along with policy optimization techniques were used with every given complexity.

3.1 Base Environment

The initial environment is the simplest setting of all. It consists of a gray-scale backdrop where the bird is represented by a white box on a black background. The camera sensor was set to gray-scale which is limited to a scale of 50 pixels in the *x-axis* and 50 pixels in the *y-axis*. This corresponds to two discrete actions that the agent can perform – move along the *x-* and the *y-axis*. The total observation would therefore sum up to 2500 (50 x 50 x 1), where 1 represents a single channel image. The reward function was set in such a way that each bird that was hit would yield a reward of +1 whereas each bird that was missed would incur a negative reward of -0.01.

$$RewardFunction = \begin{cases} +1 & Bird - hit \\ -0.01 & Bird - missed \end{cases} \quad \ldots(1)$$

3.2 Colour Channels

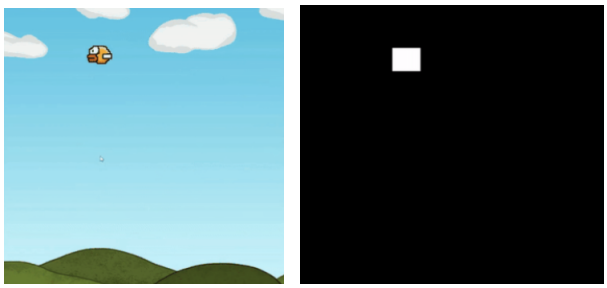
3.2.1 Environment Setup

This setting is an established on the initial environment where instead of using grayscale for the background all three-color channels (RGB) are used. The scaling and the action space is the same as base-environment whereas the observation space now increase to 7500 (50x50x3), where 3 represents the color-channels. The reward function is also similar to that of the base environment.

3.2.2 Models Used

The base environment was compared with the colored environment and for the purpose of evaluation the following model structures were used:

- Policy Optimization methods - PPO vs SAC
- Traditional RL vs Imitation Learning
- Good Expert vs Suboptimal Expert



(a) Base environment (b) Grayscale Camera

Fig. 1: (a) The base environment that consists of a bird that flies across the screen. (b) The bird is represented as a white box on a black background as seen by the gray-scale camera sensor.

3.3 Multiple Bird

3.3.1 Environment Setup

For this environment, we have introduced new complexity to the previous environment where we had provided colour to the agents' observation. Here we introduce, two new birds apart from the basic target bird. The complexity is introduced basically by the functionalities of these birds. The yellow bird will be the basic target while the red bird is added as a bonus where it spawns when two yellow birds are hit by the agent and the blackbird acts as a bombshell where it explodes when hit by the agent.The environment with all birds can be seen in Figure 2.

With respect to RL, we have the following basic inferences:

- *Action*: x,y coordinates
- *Environment*: 2D Birds on the Map
- *Observation*: Vector of each point in the Map
- *Rewards*: Reward function for each guess or hit

$$RewardFunction = \begin{cases} +1 & Hit = YellowBird \\ +1.4 & Hit = RedBird \\ -0.5 & Hit = Miss \\ -0.5 & Hit = BombShell \end{cases} \quad \ldots(2)$$

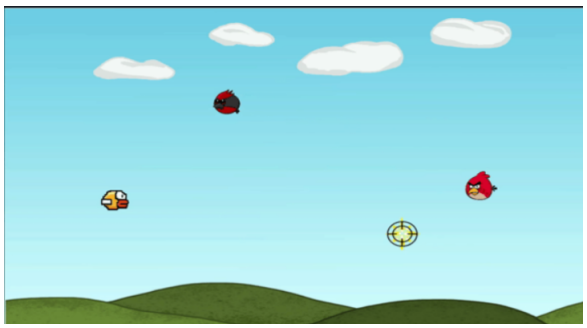


Fig. 2: Multiple Bird in the Environment and the Agent.

3.3.2 Model Used

For the above environment setup we compare the following RL models:

- Traditional RL with PPO
- Behavioural Cloning
- GAIL with PPO
- BC and GAIL together

We start with the traditional RL, which takes a lot of training time due to the explore phase. We use this as a basic metric to see how Imitation learning (BC) has its effect on the above-introduced complexities. We also have a model with GAIL so that we could see if there is any improvement GAIL can bring by introducing new policies. At last, we use BC along with GAIL to see if the model can have train fast as well as produce a better result from GAIL as it should theoretically.

3.4 Limited Ammunition

3.4.1 Environment Setup

For this environment, we introduce new parameters to the game environment in the hopes of pushing the agent to converge easier and quicker on pinpoint accuracy. A new parameter introduced, $Clip\_Size$, determined a preset amount of ammunition that enabled the agent (or the user) to shoot. Another virtual dependent parameter was $Ammo_t$ which specified the ammunition available to the player at time t. The final parameter is Reload Duration T_{reload} which was used to determine the time steps required to finish a reload action. At every time step t, the agent is forced to shoot if ammunition is available ($Ammo_t > 0$), otherwise, a reload is forced on the agent, resetting the ammunition available to $Clip\_Size$ after T_{reload} time steps, such that:

$$Ammo_t = \begin{cases} Ammo_{t-1} - 1 & Ammo_{t-1} > 0 \\ Clip\_Size & t \pmod{(Clip\_Size + T_{reload})} = 0 \\ 0 & o.w \end{cases} \quad \ldots (3)$$

$$RewardFunction = \begin{cases} +1 & Hit = YellowBird \\ +2 & Hit = RedBird \\ -0.01 & Hit = Miss \\ -0.5 & Hit = BombShell \end{cases} \quad \ldots (4)$$

3.4.2 Models Used

For the above environment, we compare the following RL models:

- Traditional RL with SAC policy optimization
- Behavioral Cloning with SAC policy optimization
- GAIL with SAC policy optimization

For both behavioral cloning and GAIL, expert demonstration were provided by recording a duration of 5000 iterations of a team member playing the game. SAC is used as the sole policy optimization technique to guarantee fairness of comparison. As with the previous experiments, we set the traditional RL algorithm as the baseline for our analysis. We also perform a comparison between behavioral cloning and GAIL without crossing one with the other.

4 Results

In this section, we will be discussing the results that were obtained in different environment settings with different models. The comparison of these models in the respective environment is based on the following evaluation metrics of the models:

- *Step count*: Number of steps required for the agent to converge.
- *Cumulative Reward Function*: Mean reward obtained by the agent in a particular number of steps.
- *Episode length*: Time taken for the agent to complete an episode. Episodes end when a bird is shot (even bombs for the multiple bird environment).
- *Entropy*: Measure for how much the agent is uncertain about choosing an action for the provided observation.

For all the results we would see the graphs where rewards, episodes and entropy are compared against the number of steps. All the graphs are normalized for ease of visual comparison.

4.1 Colour Channels

Here, the base environment was evaluated against the colored environment in three different scenarios. The ideal reward that the agent can obtain was 1.0. The outcome metrics of the comparisons can be seen in tables 1,2 and 3.

4.1.1 PPO vs SAC

In the figure 3, from the upper two graphs we can see that SAC converged much faster than PPO. This shows that SAC is more sample efficient when compared with PPO. In the entropy graph, we see that PPO was more confident in choosing an action which shows that PPO is relatively more stable than SAC.

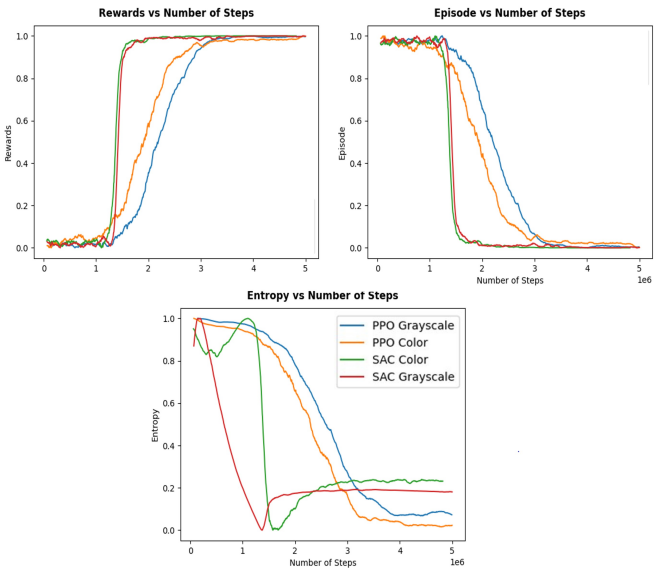


Fig. 3: Comparing base environment with the colored environment in terms of policy optimization methods.

Metric	PPO Gray	PPO color	SAC Gray	SAC color
Step Count	162k	246k	97k	106k
Cum. Reward	0.98	0.99	0.99	0.99
Entropy	0.66	0.45	0.87	0.9

TABLE 1: Comparison of policy optimization methods on grayscale and color environments.

4.1.2 *Traditional RL vs Imitation Learning*

All the models in this setting were trained using the SAC method. For imitation learning methods, a total of two models were trained - one model was trained only using BC and the other model was trained by adding GAIL to BC. As observed in figure 4, traditional RL converged much faster than IL methods but at the same time from the entropy graphs we saw that IL methods were more confident in choosing an action over the time.

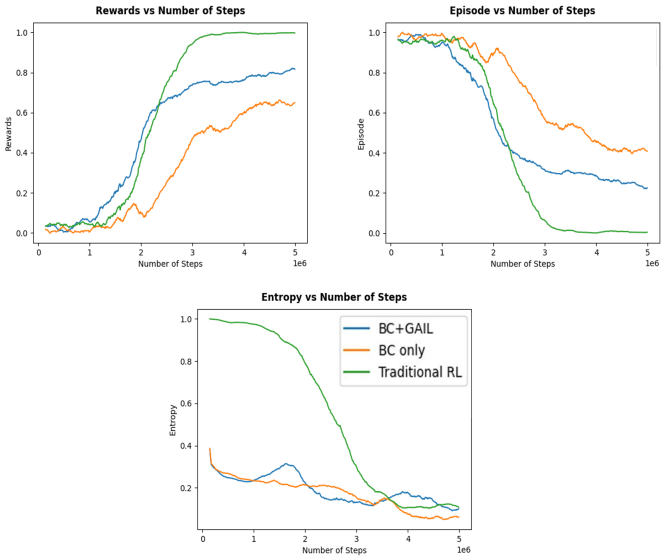


Fig. 4: Tradition RL methods and Imitation learning methods are compared on the colored environment.

Metric	RL	BC Only	GAIL Only	BC + GAIL
Step Count	162k	>500k	≫1M	500k
Cum. Reward	0.98	0.89	-	0.95
Entropy	0.66	0.45	-	0.63

TABLE 2: Comparison of Traditional RL and Imitation Learning methods.

4.1.3 *Good Expert vs Sub-optimal Expert*

For this the colored environment was assessed on two different kinds of expert demonstrations. The purpose of evaluating in this setting was to see how different kinds of experts function on the exact same environment. One of the demonstrations involved a good expert with a mean reward of 0.997 which can shoot the bird instantly without missing and another demonstration which had a sub-optimal expert with a mean reward of 0.81 which misses hitting a bird frequently. As seen in figure 5, we observed that with a good expert demonstration the agent learnt much faster and was more consistent and confident when it came to choosing an action whereas with a sub-optimal expert demonstration it did converge at some point, but it took almost double the time than the good-expert to even start converging.

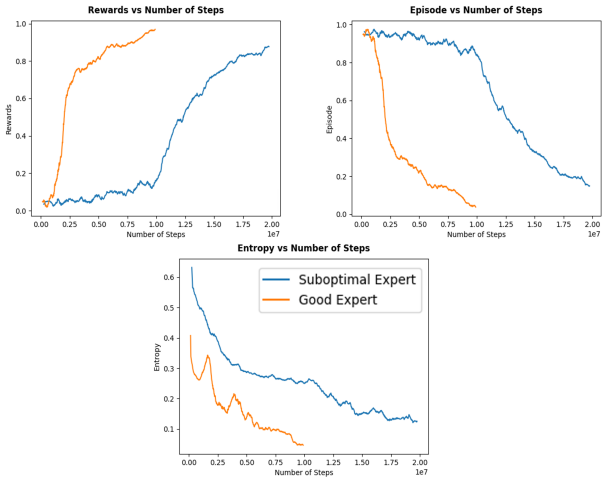


Fig. 5: Comparison of Good expert $\overline{reward} = 0.997$ demonstration and sub-optimal expert $\overline{reward} = 0.81$ demonstration on the colored environment.

Metric	Good Expert	Sub-optimal Expert
Mean Reward (Demo)	0.997	0.81
Step Count	500k	≫1M
Cum. Reward	0.91	0.77
Entropy	0.31	1.02

TABLE 3: Comparison of Good expert demonstration and Sub-optimal expert demonstration on the same environment

4.2 Multiple Bird

For the Multiple Bird setup, we trained the agent with 4 different models. The ideal reward that can be obtained by the agent for this setup is 1.1. Three of the models used the expert demo which had a 5k step count and had a cumulative reward of 1.092. The metrics of the models can be seen in Table 4 and the comparison in figure 6 and 7.

4.2.1 *Traditional RL with PPO*

From the results, it can be seen that the RL got a reward of 1.031 by going over 1.2 Million steps and since it took more than $500k$ steps to obtain a small increase further, it can be concluded that traditional RL takes a large number of steps to get a good reward.

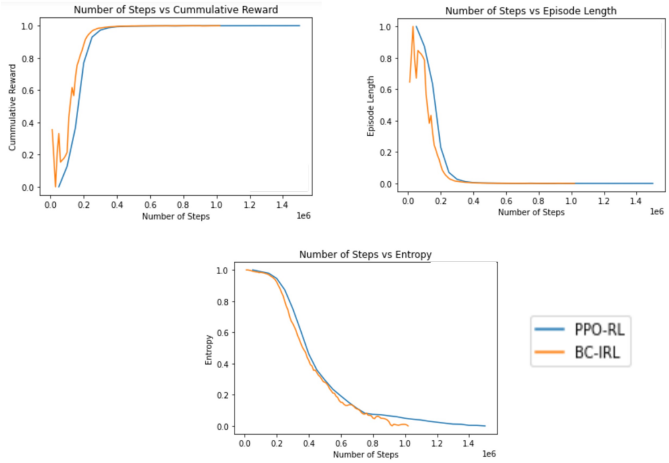


Fig. 6: Comparing Cumulative Reward and Episode length vs number of steps of RL with PPO and BC

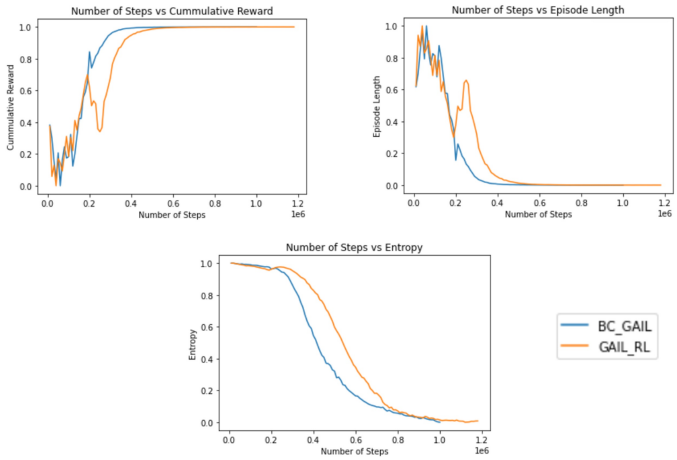


Fig. 7: Comparing Cumulative Reward and Episode length vs number of steps of GAIL with PPO and BC ith GAIL

4.2.2 BC

When BC is used to train the model with the expert demonstration, it can be seen that this Imitation learning method always provides a head-start for the agent and is able to learn the provided demonstration 20 25 percent faster than traditional RL. And we can see that the uncertainty is less when compared to RL with PPO.

4.2.3 GAIL with PPO

GAIL is used so that it will generate its own policies based on the demonstrations received and can have more actions that were not seen in the demonstration. Although this model had more uncertainty than other models because of its unseen generated policies, this model produced a better cumulative reward than any other models.

4.2.4 BC with GAIL

Since the previous model had a better cumulative reward, this has experimented with the assumption that incorporating GAIL to BC will increase efficiency and decrease uncertainty. But this model resulted in a less efficient but more certain model than the others. And it can be seen that this model had given a head-start because of BC when the GAIL had to explore more of its generated policies.

Metric	RL-PPO	BC	PPO-GAIL	BC-GAIL
Step Count	1.2M	**900k**	1.1M	1M
Cumulative Reward	1.031	1.005	**1.055**	1.021
Entropy	1.271	1.24	1.6929	**1.1436**

TABLE 4: Comparing the traditional RL with PPO, BC, GAIL with PPO, and BC with GAIL based on the comparison metrics.

4.3 Limited Ammunition

For the Limited Ammo environment, the agent was trained with 3 different models. The ideal reward obtainable was 2.0. The expert demonstration provided for the imitation learning algorithms spanned 5000 iterations with a cumulative reward of 1.5. The resultant metrics of the experiment is shown in Table 5. The results are also visualized in figure 8

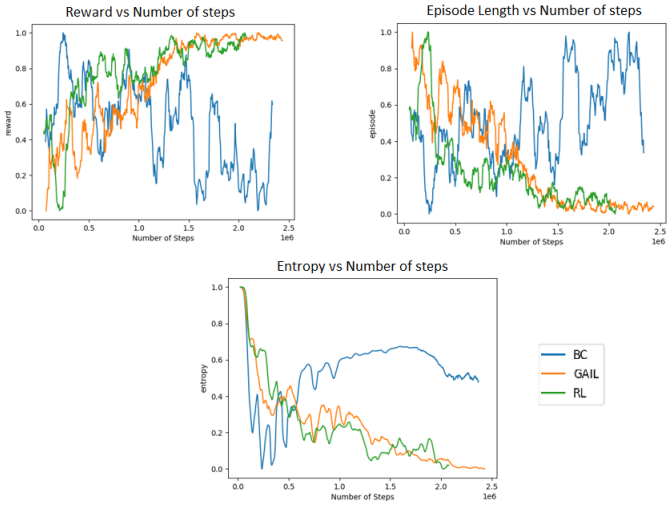


Fig. 8: Comparison of average cumulative reward, episode length and model entropy between RL, BC and GAIL in the Limited Ammo environment

4.3.1 Traditional RL

Due to the significantly increased complexity of the environment, the traditional reinforcement learning algorithm was unable to capture a proper way of shooting the birds. Instead, the agent was able to ”cheat” the environment by learning the average positions of the spots where the red and the yellow birds spawned. The agent then proceeded to only shoot at these locations, barely moving the cursor. The traditional RL agent was able to almost reach the score of a human player using this method. This testifies for the capability of RL algorithms to find loopholes given enough time to do so.

4.3.2 Behavioural Cloning

On the other hand, the BC algorithm struggled greatly to try and reach the human player score. Since the recorded demonstration did not utilize the environment loophole, but instead moved the cursor around and tried to aim for the red and yellow birds while avoiding the black birds; the agent was unable to properly replicate the demonstrated behaviour and failed to converge (or even improve) after a significant number of iterations. This showcases the limitation of imitation learning algorithms in general, since their usual approach involves blindly trusting the expert demonstration and doing less ”exploratory” approaches (when compared to traditional RL).

4.3.3 GAIL

Finally, the GAIL algorithm suffered a similar struggle to the BC algorithm at first. But since GAIL combines the best of both worlds, it was able to break free of the recorded behaviour and find the same loophole that the traditional RL algorithm found. In the end, the GAIL algorithm scored higher than any other algorithm, including the recorded human demonstrations, while getting the lowest model entropy of all. This falls in line with the idea that GAIL is recommended when the agent environment has a high level of complexity and dimensions.

Metric	RL	BC	GAIL
Converged Step Count	2.5M	∞	**1.1M**
Cumulative Reward	1.4	-0.77	**1.67**
Entropy	0.68	6.64	**0.23**

TABLE 5: Comparison between RL, BC and GAIL in the Limited Ammo environment

5 Challenges Faced

Since our experiment was done in Unity and utilized ml-agents package, we faced a number of challenges in environment construction, model design and connectivity due to the team’s lack of expertise with Unity. Added to that, the fact that our experimentation involved building an environment that can accommodate both a human player (for recording) and an AI agent; this coexistence proved challenging to implement.

Due to the time constraint of the project, and also the dramatically increased training time after adding colour channels (which effectively tripled the input size to the model), our experiments did not cover all combinations of model architectures, environment complexities and policy optimization techniques.

Bridging the gap between the PyTorch backend of ml-agents and the unity environment was far from straightforward. This pushed us to focus our efforts on developing our experiments on only one side (unity) of the pipeline. Finally, designing the reward function proved quite a challenge for the environments with higher complexities such as Multiple Birds or Limited Ammo.

6 Conclusion and Discussion

We compared different policy optimization techniques, PPO and SAC, and observed that PPO had a more stable convergence when compared to SAC and resulted in a lower model entropy at convergence signifying that the agent was more confident in choosing an action. However, what SAC lacked in stability, it made up for with sample efficiency. SAC converges much faster than PPO, making it an optimal policy in some of our environments due to our time constraint.

When comparing the different model architectures on different environment complexities, we observed that imitation learning algorithms (on average) converge slower but reaches a lower model entropy resulting in agents with more confidence than the ones trained with traditional RL algorithms. However, this comes at the cost of binding the agent’s ”thinking” to the expert’s demonstration behaviour which might steer it away from finding loopholes in the environment like RL algorithms can. Finally, GAIL was able to bring the best of both worlds in our final environment complexity, showcasing its power at capturing the gist of expert demonstrations while exploring the environment and bringing the advantages of RL algorithms to the table.

References

[Argall et al.(2009)] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57 (05 2009), 469–483.

[Chen et al.(2016)] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. arXiv:1606.03657 [cs.LG]

[Codemonkey(2022)] Codemonkey. 2022. Code monkey.

[Finn et al.(2016)] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization. arXiv:1603.00448 [cs.LG]

[Haarnoja et al.(2018)] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs.LG]

[Hadfield-Menell et al.(2017)] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart Russell, and Anca Dragan. 2017. Inverse Reward Design. arXiv:1711.02827 [cs.AI]

[Ho and Ermon(2016)] Jonathan Ho and Stefano Ermon. 2016. Generative Adversarial Imitation Learning. arXiv:1606.03476 [cs.LG]

[Li et al.(2017)] Yunzhu Li, Jiaming Song, and Stefano Ermon. 2017. InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations. arXiv:1703.08840 [cs.LG]

[Ross et al.(2013)] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. 2013. Learning monocular reactive UAV control in cluttered natural environments. , 1765-1772 pages.

[Schulman et al.(2017)] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

[Sutton and Barto(2018)] Richard S. Sutton and Andrew G. Barto. 2018. Reinforcement Learning: An Introduction.

[Taylor(2018)] Matthew E. Taylor. 2018. Improving Reinforcement Learning with Human Input. , 5724–5728 pages.

[Torabi et al.(2018)] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral Cloning from Observation. arXiv:1805.01954 [cs.AI]

[Verma et al.(2021)] Abhinav Verma, Hoang M. Le, Yisong Yue, and Swarat Chaudhuri. 2021. Imitation-Projected Programmatic Reinforcement Learning. arXiv:1907.05431 [cs.LG]