RL Project: Investigation into the use of DQN and PPO in a multi-agent environment

Steven Etches spe24@bath.ac.uk

Mark Hazell mph55@bath.ac.uk

Adam Rasool aar73@bath.ac.uk

Will Prior wp342@bath.ac.uk

1 Problem Definition

The reinforcement learning problem that we have chosen to address is from the multi-agent reinforcement learning (MARL) environments provided in PettingZoo.ml by Terry et al. (2020). These environments represent a generalisation of a predator and prey type environment that has good agents and adversaries, with some of the more complex environments also incorporating landmarks and objects. The environment that has been chosen is the Simple World Comm scenario Mordatch and Abbeel (2017).

Simple World Comm consists of six agents, two on the "good" team and four on the "adversary" team. Good agents are rewarded for proximity to a 'food' object(s), adversaries that are rewarded for collisions with good agents, and also there are two additional type of objects, a barrier that blocks the way and two 'shrubberies' that hide the position of the agents from observation. One of the adversary agents is leader adversary that can always see the location of the good agents and communicate that location to the other adversary agents.

Good agents will attempt to maximise their reward by remaining proximal to the 'food' while also attempting to minimise the number of collisions with adversaries. Adversaries on the other hand will be focused on attempting to collide and remain as close as possible with the good agents.

While the environment exists on an infinite plain, the good agents are incentivised to not move too far away from the origin point by an increasingly large negative reward every step once a threshold is crossed proportional to the distance from the origin. Adversarial agents are not penalised in this manner but since they are already rewarded for attempting to collide with the good agents, if the good agents remain within the bounded area so should all of the adversaries.

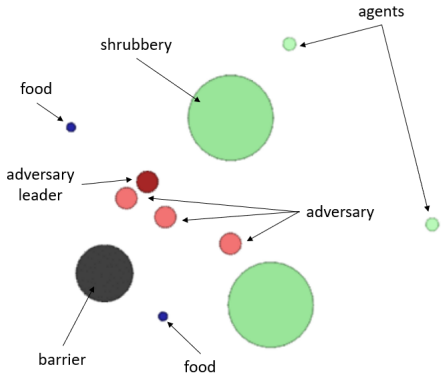

Figure 1: Simple Comm World Environment

The transition dynamics for the environment is a cycle as depicted in figure 3 found in Appendix A where each agent gets to take an action, once all agents have acted that step can be considered completed beginning a new step and cycle. The default episode length for the environments is 25 steps however this was extended in this project to a minimum length of 50.

The adversary and good agents both share the same action of space of either taking no action or, moving in the cardinal directions. The adversary leader's action space is the product of the movement actions that the other agents have and also the ability to either not communicate or communicate with one of the other adversary agents. These actions spaces are detailed in tables 3, 4, and 5 in Appendix A.

Good agents are rewarded for both colliding with the food objects and also mantaining a position within close proximity of the food. The reward for being proximal to the food decreasing with distance. Good agents are punished with a negative reward colliding with adversaries. Adveraries are rewarded for colliding with good agents while earning a punishment each step in the form of a negative reward proportional to how close good agents are to the food. More details on the reward sructure of the environment can be found in table 6 in Appendix A

This project focuses on the application of the Deep Q Learning (DQN) and Actor Critic algorithms to this environment, comparing their relative effectiveness to each other. Of particular interest will be contrasting the rate at which the two algorithms learn and also the rewards that their optimal policy results in.

2 Background

The first reinforcement methods we have consider for this problem are:

- Deep-Q Network
- REINFORCE
- Advantage Actor-Critic (A2C)
- Proximal Policy Optimisation (PPO)

The Deep-Q Network algorithm was first developed by Deep Mind in 2015 Mnih et al. (2015). This uses a combination of the reinforcement algorithm, Q-learning, and deep neural networks. It has been implemented on many Atari games in the OpenAI gym and achieved levels far exceeding human capabilities Mnih et al. (2015).

Algorithms that follow a value-function approach whilst performing well in some deterministic environments have been shown to be unable to converge to approximate a stochastic policy Sutton et al. (1999). The REINFORCE algorithm is a type of policy gradient algorithm that provide unbiased estimates of gradient but without the learned value function and are able to make weight adjustments in the direction of gradient reinforcement Williams (1992). Whilst this is appropriate for our stochastic environment, the main weakness is the long time required to train the agents. For example when implemented on Pong and Lunar Lander it reportedly took 96 hours when running on cloud GPU Noufal (2020).

A further group of policy gradient based algorithms which may be applied to this problem are proximal policy optimisation (PPO) algorithms as discussed by Schulman et al. (2017). This family of algorithms avoids excessively large policy updates by considering a policy ratio between the new and old policies and ensuring that this ratio stays within an acceptable range $1 \pm \varepsilon$ where ε is a hyperparameter. Owing to their relative simplicity in addition to their effectiveness Yu et al. (2021), these algorithms are popular, with research available which extends their application to multi-agent systems such as that discussed in Schulman et al. (2017) which outlines the multi-agent PPO algorithm MAPPO.

In a like manner to the extension of the single-agent PPO algorithm to a multi-agent problem cited above, Paczolay and Harmati (2020) propose a modification to the advantage actor-critic (A2C) algorithm. The vanilla A2C algorithm utilises two neural networks, one to optimise on the basis of policy and a second to optimise value; respectively controlling agent behaviour and providing a measure of how 'good' the agent's actions are. The modified algorithm proposed in this work applies this philosophy to a cooperative multi-agent problem and produces very promising results in the

author's testing. It is noted however that "... the algorithm has the caveat of only being able to be used when the agents are fully cooperative without any special predefined roles between them". Terry et al. (2020)

3 Method

During the initial implementation of both of the DQN and A2C algorithms a simpler problem was used allow for easier testing for correctness of the implementations. The Simple Push environment is characterised by two particles, an adversary and an agent, which are attempting to earn as large a reward as possible for being close to a landmark. The agent is rewarded proportional to its closeness to the landmark while the adversary is rewarded in two parts. First the adversary receives a reward proportional to its closeness to the landmark and, second it is rewarded proportionally for how far away the agent is from the landmark. This environment incentivises that adversary to both keep close to the landmark and also to keep the agent as far away from the landmark as possible.

3.1 Deep Q-Learning

In traditional Q-learning the algorithm records the value of state-action pairs in what is known as a Q-table. Given a state the algorithm will first determine if it will explore randomly or taking an action which returns the maximum reward available (act greedily). If the choice to act greedily is made then a lookup will be made to the Q-table and if there are state-action pair values recorded it will choose the one in accordance to an action-value function.

However as the state and action spaces grow in both scale and complexity it becomes increasingly difficult and impractical to attempt to capture every state-action pair in a tabular form. Mnih et al. (2015) instead proposed utilising a neural network to instead approximate the action-value function. By presenting the state to a neural network and then utilising the ouputs to approximate the action-value function makes it possible to apply a behavioural policy typical to traditional Q-learning algorithms to these more complex problems.

The high levels of correlation present in reinforcement learning environments (each step is directly correlated to both the one before and after) can pose a problem to neural nets where they are prone to over fitting. To combat this experience replay was introduced where with each step a random batch of previous state, action, reward tuples is selected and used to train the net. By utilising this batch method of training it reduces the correlation between the data to reduce the chance of overfitting.

Another issue that can effect DQN algorithms is that they can be unstable and prone to sudden changes if the error is too large when using a normal certain loss formulae such as mean squared error (mse). This tendency for large changes to be made to the network is mitigated by utilising a different loss formula such as the Huber Loss utilised in this project which less severely punishes large errors.

A final technique to improve the accuracy of the neural net approximations is to use a fixed target network allowing for updates to be made towards a stationary target. This target network's weights are updated from the primary network every C steps.

3.1.1 Implementation

Deep Q-Learning (DQN) was first developed utilising the Simple Push environment described in Section 3 allowing for quicker testing of the algorithm.

For each environment each agent type shared a neural net. In Simple World Comm the adversary leader did not share a net, the three adversaries shared a single net, and the good agents shared a single net. By sharing the neural net between agent types it allowed the teams the benefit of learning from the actions that every agent on that team took and the rewards that earned.

The final hyperparemters that the algorithm was run with were:

3.2 Actor Critic

Touched upon in Section 2, PPO algorithms are policy gradient algorithms which aim to produce better results by limiting an agent's learning rate. These algorithms curb changes to policy, preventing a new

Hyperparameter	**Value**
Initial Exploration Rate ϵ	1
Exploration Decay Rate	0.05
Minimum Exploration Rate ϵ	0.1
Decay Rate γ	0.95
Learning Rate α	0.00025
Batch Size	8
Target Network Update	Every 10 steps

Table 1: Deep Q-Learning hyperparameters

policy from deviating too far from the one that preceded it. This is achieved through modification of the objective function. For the PPO-clip algorithm this is given by:

$$L^{CLIP}\theta = \hat{\mathbb{E}}[min(r(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{1}$$

Clipped Surrogate Objective Schulman et al. (2017)

Where the advantage function A^t is given by a truncated version of the Generalized Advantage Function (GAE)

$$\begin{aligned} A^t &= \delta_t + (\gamma\lambda)\delta_{t+1} + ... + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \\ &where\, \delta_t = r_t + \gamma V(S_{t+1}) - V(s_t) \\ &\lambda = smoothing factor \end{aligned} \tag{2}$$

Truncated GAE Schulman et al. (2017)

And $R_t(\theta)$ is the ratio

$$\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{3}$$

of the new policy to the previous. The first term in the above Clipped Surrogate Objective function gives the normal policy gradient objective, the second gives us a clipped policy gradient objective. The latter is equivalent to the former with the additional application of a clipping operation within the range of $1 \pm \epsilon$ for hyperparameter ϵ. This clipping operation is visualised below.

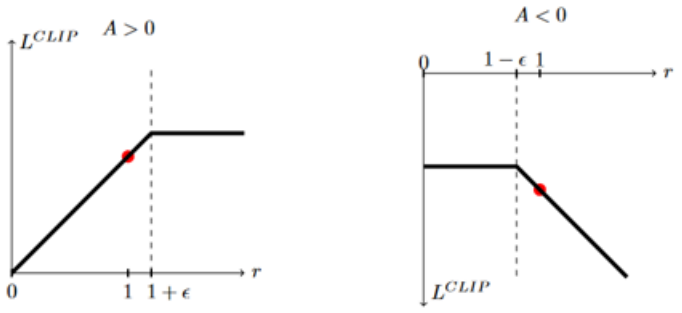

Figure 2: Plots showing one term (i.e., a single timestep) of the surrogate function LCLIP as a function of the probability ratio r, for positive advantages (left) and negative advantages (right). The red circle on each plot shows the starting point for the optimization, i.e., r = 1. Note that LCLIP sums many of these terms." Schulman et al. (2017)

A minimum is then computed over these two terms; in this way, the growth of $r_t(\theta)$ is constrained by an upper bound and the policy cannot change excessively after a single alteration. This helps to stabilize training via constraining the policy changes at each step. This is useful as our gradient is only an approximation so trajectories which have high variance and large steps could be harmful to the policy. Because of this PPO can performs multiple epochs of minibatch updates, compared to a single policy gradient update per data sample in usual policy gradient methods. This allows for greater learning per data sample and therefore a greater learning performance.

This algorithm has a number of strengths which make it particularly useful. Maintaining conservative policy updates as detailed above provides better sample efficiency allowing the algorithm to converge more easily on an optimum. Further, this algorithm was designed to be simpler to implement than the comparable Trust Region Policy Optimisation (TRPO) algorithm Schulman et al. (2015) which also provides better sample efficiency than previous policy gradient algorithms "and have better overall performance" Schulman et al. (2017). More relevant to this project, Baker et al. (2019) have explored the application of this algorithm to a 'hide and seek' problem, producing positive results in an environment similar to that which is explored here. Given that this algorithm is known to perform well in a similar environment and the various strengths of the PPO-clip algorithm, it was chosen for further investigation in this project.

3.2.1 Implementation

As with the DQN algorithm discussed in subsection 3.1.1 the A2C algorithm was initially applied to the Simple Push. When applied to Simple Push the episode length was 250 steps.

The algorithm was run with the following hyper parameters for both Simple Push and Simple World Comm:

The Simple World Comm environment was set up as described in Section 2. Episodes were set to a length of 500 steps. Each type of the agent type in the environment uses the same actor and critic network. Therefore, in the implementation of 1 lead adversary, 3 adversaries and 2 (good) agents there are 3 sets of actor and critic networks.

The actor and critic network had the same input neuron number, being the size of the agent's state space. They also had the same hidden layer structure consisting of 2, 256 neuron layers both with

Hyperparameter	Value
Discount Factor, γ	0.99
Learning Rate, α	0.0003
Smoothing Factor, λ	0.95
Policy Clip, ϵ	0.2
Batch Size	20
Mini-batch Size	5
Number of epochs	4

Table 2: Actor Critic Hyper Parameter Values

Rectified Linear Unit (relu) activation functions. The actor network having a dense output layer with the same number of neurons as the number of actions in the agent's action space and a SoftMax activation function giving an effective probability distribution of taking each action. The critic network has a single output with no activation function as it evaluates a particular state. They were both compiled and optimized with an Adam optimizer Abadi et al. (2015) with the above learning rate, α.

A high-level program flow shows each agent to take an action based on their current policy. After this the current state, output of each actor and critic network, action taken and reward from taking the action are recorded. It was chosen that the learning part of the algorithm would be carried out every 20-time steps, using minibatches of 5 which are run for 4 epochs. This means that for every 20 steps taken the lead adversary has a single dataset to learn from, the adversaries have 3 and the agents have 2. It was chosen to be implemented this way instead of individual networks per agent, for the obvious reason that each policy network will learn more in the same number of learning steps for the agent and adversaries.

4 Results

5 Discussion

We have produced two algorithms that have demonstrated reinforcement learning in a multi-agent, complex and competitive environment. The results also suggests the development of new strategies to maximise rewards, that is then countered/overcome by the opposing adversarial agents. The potentially mimics the evolutionary 'arms race' seen between competing organisms directed by natural selection Dawkins and Krebs (1979).

6 Future Work

Future work on this project could involve analysis of the strategies that have developed at specific episode intervals as has been documented in Baker et al. (2019). Possible examples of strategies might be the running and chasing of the agents and adversaries, the agents hiding in the 'shrubberies' to avoid detection, development of the adversary leader communicating locations to the other adversaries. At the moment only the rewards for each episode are recorded, but to allow analysis of the strategies monitoring of other in-episode parameters could be applied. Examples might be recording the amount of movement of the agents/ adversaries, distance of the agents to the 'shrubberies', communication and response of the adversaries.

7 Personal Experience

7.1 Steven Etches

7.2 Mark Hazell

In many ways a primary hallmark of my experience in this project has been one of frustration and waiting. I am used to being able to rapidly iterate on ideas and code however while implementing the DQN aglorithm I was often faced with long training times in the double digit hours to achieve a

reasonable number of episodes. This often led to the completion of a training session before noticing some manner in which the algorithm could be improved for performance or not quite correct and having to repeat this training multiple times. However in spite of this I have learnt a lot about how to implement the algorithms discussed in this report as well as how to break down a more complex reinforcement learning problem into constituate parts (going from Simple World Comm to Simple Push) to prove that what is being implemented is indeed functional.

7.3 Adam Rasool

7.4 Will Prior

My personal experience with the project started with deciding the project. We went for the later found to be rather ambitious task of a multi-agent environment. I was tasked with looking into the effectiveness of policy gradient algorithms on environment. After implementing both the REINFORCE and a basic actor-critic algorithm with no noticeable learning/ consistency in the results, we went back to the drawing board. In this we found the hide and seek paper that has been mentioned previously in the report which was a very similar experiment to that of our problem. In this it noted about how the use of PPO with GAE was an algorithm that worked well in the environment - it seemed logical to implement this in our environment. Which I did, giving the pleasant surprise of some nice learning curves as seen in the results section. Giving good performance for both simple push and simple world comm. This project was a very steep learning curve in how to implement neural nets using TensorFlow and keras, which took some time to get my head round the syntax and what was required in terms of activation functions number of neurons etc. However, I now feel I have a good grasp of this along with a rather advanced policy gradient method!

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2015. TensorFlow adam optimiser. Tensorflow's Adam Optimiser. Available from: `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam`.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B. and Mordatch, I., 2019. Emergent tool use from multi-agent autocurricula. Available from: `https://doi.org/10.48550/ARXIV.1909.07528`.

Dawkins, R. and Krebs, J.R., 1979. Arms races between and within species. *Proceedings of the royal society of london. series b. biological sciences. royal society, 205(1161),*. pp.489–511. Available from: `https://doi.org/10.1098/rspb.1979.0081`.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540), Feb, pp.529–533. Available from: `https://doi.org/10.1038/nature14236`.

Mordatch, I. and Abbeel, P., 2017. Emergence of grounded compositional language in multi-agent populations. *arxiv preprint arxiv:1703.04908*.

Noufal, S., 2020. Reinforce algorithm: Taking baby steps in reinforcement learning. Accessed: 3rd May 2022. Available from: `https://www.analyticsvidhya.com/blog/2020/11/reinforce-algorithm-taking-baby-steps-in-reinforcement-learning/`.

Paczolay, G. and Harmati, I., 2020. A new advantage actor-critic algorithm for multi-agent environments. *2020 23rd international symposium on measurement and control in robotics (ismcr)*. pp.1–6. Available from: `https://doi.org/10.1109/ISMCR51255.2020.9263738`.

Schulman, J., Levine, S., Moritz, P., Jordan, M.I. and Abbeel, P., 2015. Trust region policy optimization. Available from: `https://doi.org/10.48550/ARXIV.1502.05477`.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms. Available from: https://doi.org/10.48550/ARXIV.1707.06347.

Sutton, R.S., McAllester, D., Singh, S. and Mansour, Y., 1999. Policy gradient methods for reinforcement learning with function approximation. In: S. Solla, T. Leen and K. Müller, eds. *Advances in neural information processing systems*. MIT Press, vol. 12. Available from: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Terry, J.K., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sulivan, R., Santos, L., Perez, R., Horsch, C., Dieffendahl, C., Williams, N.L., Lokesh, Y., Sullivan, R. and Ravi, P., 2020. Pettingzoo: Gym for multi-agent reinforcement learning. *arxiv preprint arxiv:2009.14471*.

Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3), May, pp.229–256. Available from: https://doi.org/10.1007/BF00992696.

Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A. and Wu, Y., 2021. The surprising effectiveness of ppo in cooperative, multi-agent games. Available from: https://doi.org/10.48550/ARXIV.2103.01955.

Appendices

Appendix A: Environment Cycle

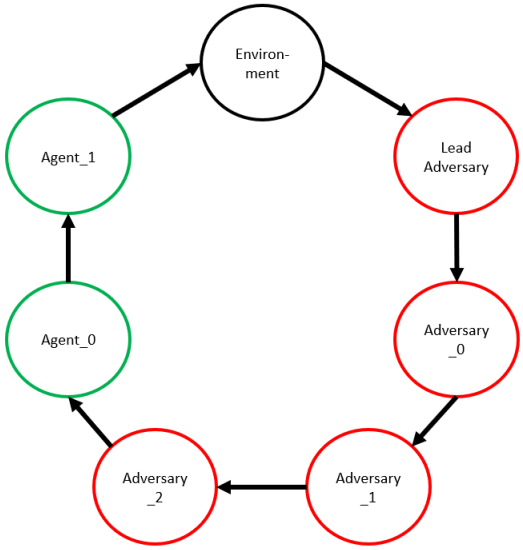

Figure 3: Environment transition dynamics.

Good Agent
no_action
move_left
move_right
move_up
move_down

Table 3: table

Good agent action space

Adversary Agent
no_action
move_left
move_right
move_up
move_down

Table 4: Adversary agent action space

Adversary Leader	say_0	say_1	say_2	say_3
no_action	(no_action, say_0)	(no_action, say_1)	(no_action, say_2)	(no_action, say_3)
move_left	(move_left, say_0)	(move_left, say_1)	(move_left, say_2)	(move_left, say_3)
move_right	(move_right, say_0)	(move_right, say_1)	(move_right, say_2)	(move_right, say_3)
move_up	(move_up, say_0)	(move_up, say_1)	(move_up, say_2)	(move_up, say_3)
move_down	(move_down, say_0)	(move_down, say_1)	(move_down, say_2)	(move_down, say_3)

Table 5: Adversary leader action space

Good Agent	-5 per collision with an adversary agent	+2 per collision with food object	-0.05 x min distance to a good object
Adversary Agent	+5 per collision with a good agent.	-0.1 x min distance to a good agent	

Table 6: Agent Rewards

Appendix B: Example Appendix 2