ECEN 689 Reinforcement Learning Project Report, Spring 2022

Analyzing Independent vs Cooperative Multi-agent RL on Google Research's Football environment

Brent Arnold Basiano, Subrahmanyam Arunachalam, Sai Namith Garapati

Abstract:

In this paper, we explore and analyze Independent Multi-agent RL vs Cooperative Multi-agent RL within the context of Google Research Football environment. We start by explaining a little about the environment that has been used for analysis. Google Research Football environment [5] is a virtual environment provided by tech giant Google to implement reinforcement learning algorithms where training of agents is done in an advanced state-of-the-art 3D simulator. This environment is compatible with Open-AI Gym and provides support for multi-agent reinforcement learning. While there are many other football environments for reinforcement learning, this environment is unique as there are various in-built scenarios to experiment with, which helps in faster implementation and cutting down of computation costs. In this project, some of these scenarios are taken to experiment for analyzing the independent and cooperative multi-agent setup.

Scenarios like academy empty goal and2 academy counterattack is set up to analyze the performance of independent multi-agent RL using A3C (Asynchronous Advantage Actor Critic) algorithm and cooperative multi-agent RL using PPO (Proximal Policy Optimization). Like in FIFA, the environment can control a single player on the pitch by default. For independent setting, N different agents are trained together to control each player. For the cooperative setting, there are still N different agents, but they communicate with each other through some parameter sharing. The parameter sharing, we have implemented in this project is sharing of policies after some T timesteps of training. This idea of parameter sharing was taken from Tan et.al's [1] paper which introduces the idea of cooperative learning in multi-agent settings. We have used this paper as the base for implementing our code for Google Research's Football Environment. Using this idea of independent and cooperative learning, these multi-agent setups were applied to the academy counterattack, academy empty goal and other scenarios where we understand different parameters and improve the algorithm from the obtained parameters.

From the results, it can be observed that cooperative multi-agent RL setting works better than the independent multi-agent RL setup but with a tradeoff on the training time.

Introduction and Related Work:

Motivation:

Multi-agent RL has widespread applications like autonomous driving, consensus of multiple drones for a particular task, multi-player games, etc. It is estimated that usage of autonomous electric driving cars will be ubiquitous in future. Hence, it is essential to understand and interpret several such multi-agent applications. Learning is one of the innate abilities of human life which is driven by the existence of intelligent behavior. Even though humans have the capacity to act on their own actions, sometimes individual effort may not help in getting the work done. This is where the idea of collaboration comes in. Like how humans collaborate as a team to get the task done, agents also collaborate by sharing the knowledge and learning from each other to complete a task or achieve higher values of cumulative rewards in a reinforcement learning setting. Hence, the obvious fact is that multiple agents can outperform single agent in most of the scenarios. Therefore, logical discussion would be to compare how N agents approach a task by acting independently versus how N agents cooperate with each other and collaborate to get the work done. This idea has motivated us to understand and analyze independent and cooperative multi agent learning on Google Football environment.

Google Research's Football Environment:

Football involves a lot of strategy and coordination which makes it an ideal platform to work on multi-agent settings. Football provides the opportunity to analyze scenarios where players can behave independently without showing much cooperation and scenarios where they can play with cooperation. Hence, we used the Google Football environment that provides a full 11 vs 11 simulation as well as drills and mini games such as corner kicks, counter attacks, and various scenarios to analyze the independent vs cooperative multi-agent RL. Google Research Football environment is compatible with Open-AI Gym and provides the opportunity to perform and experiment on various reinforcement learning tasks. Let us understand about the environment a little. Environment has an action set of 21 actions comprising of the standard move actions in 8 directions, different ways to kick and pass the ball like long pass, short pass. Etc. Definition of state in the environment contains information about ball position, possession, co-ordinates of all players, the active player, the game state, and the current pixel frame. Size of the state space is 115 x 1. Regarding rewards, a reward of +1 is obtained when scoring a goal and -1 reward when a goal is conceded. This is scoring reward. There is also a checkpoint reward that addresses the sparsity of scoring by obtaining partial rewards for advancing across the pitch. This reward structure and the state action space provides an ideal platform to experiment on.

Review of Multi-agent RL:

Research in the field of Reinforcement Learning (RL) has seen significant advancements in the past few years. Classic vanilla RL algorithms try to maximize the cumulative reward in the

environment involving only a single agent. However, in multi-agent RL, multiple agents either co-ordinate or compete to maximize the cumulative rewards.

In single agent RL, the agent is modelled in such a way that it makes sequential decisions after interacting with the environment which is usually formulated in the form of a Markov Decision Process (MDP). Value based methods like value iteration and Q-Learning are some of the techniques developed that help in getting a fair estimate of the value function obtained from state-action pairs. The optimal policy is obtained by extracting the best action from the estimate we obtained through value-based methods. Other set of RL algorithms like REINFORCE, actor-critic are Policy-based methods that use the whole policy space to estimate the optimal policy. The best policy is obtained with the help of parametrized function approximators like neural networks. The parameters are obtained using gradient descent methods by constantly moving to the direction of gradient that achieves higher value of cumulative reward. Since convergence in policy-based algorithms is faster than value-based algorithms, policy-based algorithms are preferred in the literature.

Analogous to the single agent setting, multi-agent RL also tries to come up with sequential decisions based upon the interaction with the environment. The change of the system dynamics like state and the amount of individual reward achieved by each agent is also influenced by the combined action of the rest of the agents. Here each individual agent tries to maximize its cumulative long-term rewards, which is now a function of other agent's policies as well.

The primary challenge of solving multi-agent RL systems include non-unique learning goals, non-stationarity and scalability. In single agent RL, the main goal is to achieve maximum cumulative rewards whereas in multi-agent RL the learning goals become vague. When multiple agents try to learn simultaneously, the environment is affected by each individual agent and the environment changes constantly resulting in a non-stationary environment. Action taken by an agent significantly affects the reward and state of other agents. Hence the agents should constantly adjust to the changing environment. To counter non-stationarity, each agent must act upon the joint action space which results in exponential increase in the dimensions with all other agents. These challenges make multi-agent RL an exciting field to conduct extensive research. Multi-agent Settings like Centralized setting, Decentralized setting with network agents and fully decentralized setting can be used to tackle these issues effectively [2].

Agents in multi-agent setting can interact with the environment with cooperation, competition, and a mix of both. Usually, cooperation is the ideal method to interact with the environment since agents can share knowledge with each other making the learning much more effective. In this project, we will be discussing more regarding the cooperation in multi-agent RL.

Cooperation in Multi-Agent RL:

There are three possible ways in which agents can communicate with cooperation, according to [1]. The first method is sharing instantaneous information like actions and rewards. Here the information which is shared is only useful when the information obtained is relevant and beneficial for learning. The second method involves agents communicating episodes as sequences of state, action, reward. Here the state space increases exponentially. In the third method, agents try to share the policy that they learnt. These are usually the settings in which multiple agents can learn and cooperate with each other.

As seen above, agents learn through cooperation by exchanging instantaneous information, through episodic experience and shared knowledge. Hence, in this project, we plan to analyze whether cooperative agents will outperform independent agents provided with same number of agents to learn. Asynchronous Advantage Actor Critic (A3C) algorithm is used to train the independent scenarios whereas Proximal Policy Optimization (PPO) is used to train the cooperative setting. Let us understand the algorithms further.

Asynchronous Advantage Actor-Critic (A3C):

Algorithm 1 A3C Pseudocode [20].

//Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
//Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Syncronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get sate s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Recieve reward r_t and new state s_{t+1}
    $t \leftarrow t+1$
    $T \leftarrow T+1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 \text{ for terminal } s_t \\ V(s_t, \theta'_v) \text{for non-terminal } s_t \text{ // Bootstrap from last state} \end{cases}$
  for $i \in t-1, \ldots, t_{start}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ' : $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v : $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_i;\theta'_v))^2/\partial\theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$
until $T > T_{max}$

Source: https://www.mdpi.com/2076-3417/11/11/4948/pdf

Asynchronous Advantage Actor Critic (A3C) (given above) algorithm is used to implement independent multi-agent RL settings in our project. A3C is one of the prominent policy gradient algorithms where a policy $\pi(a_t \mid s_t; \theta')$ and an estimate of the value function $V(S_t; \theta'_v)$

is maintained. In our assignment 4, we implemented A2C (Advantage Actor Critic) where the critic learned an optimal value function given the states as input while the actor learned a policy in terms of the states. Both the actor and critic were trained using a parameter called advantage which is the difference between Q-value function and the value function at each timestep. The extension to A3C is just the existence of multiple independent agents working on a different copy of the environment in parallel.

Proximal Policy Optimization (PPO):

Algorithm 1 Modified multiagent PPO with cooperation

Input: initial policy parameters $\theta_0^{(m)}$, initial value function parameters $\phi_0^{(m)}$, $T = 100$

for $k = 0, 1, 2, \ldots$ **do**

  for $m = 0, 1, 2, \ldots$ A **do**

    Collect set of trajectories $\mathcal{D}_k^{(m)} = \tau_i$ by running policy, $\pi_k^{(m)} = \pi(\theta_k^{(m)})$ in the environment.

    Compute rewards to go $\hat{R}_t^{(m)}$

    Compute $\hat{A_t^{(m)}}$ based on the current value function $V_{\phi k}^{(m)}$

    Update the policy by maximizing the modified PPO objective given below:

$$\theta_{k+1}^{(m)} = arg\max_{\theta^{(m)}} \frac{1}{|D_k|^{(m)} T^{(m)}} \sum_{T \in D_k^{(m)}} \sum_{t=0}^{T^{(m)}} min(\frac{\pi_\theta^{(m)}(a_i|s_t)}{\pi_\theta^{(m)}(a_t|s_t)}) A^{\pi_\theta}{}_k^{(m)}(s_t, a_t), g(\epsilon, A^{\pi_\theta}{}_k^{(m)}(s_t, a_t))$$

    typically via stochastic gradient ascent with Adam.

    Fit value function by regression on mean-squared error:

$$\phi_{k+1}^{(m)} = arg\min_{\phi^{(m)}} \frac{1}{|D_k|^{(m)} T^{(m)}} \sum_{T \in D_k^{(m)}} \sum_{t=0}^{T^{(m)}} (V_{\phi^{(m)}}(s_t) - \hat{R}_t{}^{(m)})^2$$

    typically via some gradient descent algorithm.

  end for

  if k%T == 0:

    $temp \leftarrow \theta_k^{(1)}$

    $\theta_k^{(1)} \leftarrow \theta_k^{(2)}$

    $\vdots$

    $\theta_k^{(A)} \leftarrow temp$

end for

Modified from: https://towardsdatascience.com/elegantrl-mastering-the-ppo-algorithm-part-i-9f36bc47b791

The algorithm (given above) used to implement the cooperative multi-agent setup for our project is Proximal Policy Optimization (PPO). The traditional PPO is modified for multi-agents by repeating the same algorithm for the N different agents. In PPO, various small batches of the agent's interaction with the environment are collected and the collected batches are used to update the policy. In PPO after each update, it is ensured that the newly obtained policy does not differ much from the previous policy by clipping the updated policy thereby preventing it from becoming too big. A considerable amount of variance is reduced during training by following this approach which makes the training process much smoother making it ideal for the cooperative setting. So, in order to incorporate cooperative approach, after 100 timesteps, the policies are exchanged in a clockwise order (i.e., first agent gets second agent's policy, second agent gets third agent's policy and so on).

Novelty:

The novelty for this project is that we aimed to analyze and compare independent vs. cooperative multi-agent RL by evaluating different scenarios of Google Research's football environment. To our knowledge, only one paper [6] used coordinated learning instead of cooperative learning and there are limitations in their paper which we have tried to address using the base paper [1]. Also, there are no code implementations of the base paper [1], which is our main novel work. Coordinated learning is using a single agent which takes in states of all the N agents and spits out an N action vector which corresponds to the N different agents. This does not take into consideration the fact that only some of the states are observable by each agent. Hence, we improve the realistic scenario by sharing just policies, which is a novel idea from the base paper [1]. We have also implemented A3C algorithm from scratch, which was an extension of the code from Assignment 4 in class. But we had to shift the implementation to TensorFlow to accommodate Google Research's Football environment's dependency issues. We, however, used a ChainerRL library to implement PPO algorithm and included the cooperative code on top of that due to time constraints and the lack of optimized code hurting the training time severely.

Hardness:

Multi-agent RL increases in difficulty when more agents are being trained. This problem can cause longer training time for the agents. The state action space of football is also large, and the nature of football requires complex strategies for the agents to learn. Regarding exploration, since more agents are exploring the environment, the amount of information gathered can be large which consumes substantial amounts of computing resources. We also faced multiple issues while working on the project which are listed below.

Issues:

- HPRC – We originally wanted to use the HPRC to take advantage of its resources. It would allow us to train the model faster; however, the packages required to run the environment on the HPRC cannot be installed. We first asked support to install these packages through the 'sudo' command, but they kept persisting that the packages are already installed, and the football environment can be installed through the Conda environment. Instead, we used a laptop that contained a GPU for training. On top of this, we could not render our model while it is getting trained in a headless server like HPRC, so we avoided using the server and trained on Brent's laptop.

- Outdated packages - The environment provided by Google Research appeared to use outdated packages. This problem caused our recent versions of Python and TensorFlow to have compatibility errors. To resolve this issue, we used a Conda environment and installed older versions of Python and TensorFlow to install the packages.

- One local Machine – Since we are not sure how to put the environment to the HPRC, we used one local machine which contained an Nvidia GPU for training the models.

- Ray Module - The training using Google Research's football existing code contained using the framework Ray to accelerate hyperparameter tuning on top of the RL Lib environment. But the process initialization had some issues and cooperative multi-agent RL took more than 1.5 days to train the entire model.

- JSON - for the cooperative multi-agent RL, the checkpoint files were stored in JSON format which was hard to infer from. So, inferring the plots and getting the final rendering video was hard.

- Cooperative Training Time - The cooperative multi-agent RL, although gives better results and cooperation between the individual agents, takes a long time to train as consensus between different agents cannot be achieved quickly, that too with random initialization of parameters.

Execution:

For our setup, we used Google Research's Football Environment to test our models. The environment contained 11 scenarios for training and evaluation. We used 6 of the 11 scenarios for training. The following scenarios used are:

- Academy empty goal close – The agent needs to score against an empty goal. (Independent)
- Academy run to score – The agent starts at the center and needs to score a goal against 5 defenders (Independent)
- Academy pass and shoot with keeper – Two players try to score against two defenders – 1 is a keeper. (Independent + Cooperative)
- Academy run pass and shoot with keeper – Two players try to score where one defender is at the center with one agent and the keeper is facing the other agent. (Independent + Cooperative)
- Academy 3 vs 1 with keeper – Three players try to score. One is at the left side of the pitch, at the right, and at the center. The center player has the ball against a defender. A keeper is present at the goal. (Independent + Cooperative)
- Academy counterattack – Four players against 1 defender counterattacking. (Independent + Cooperative)

The environment is compatible with Open-AI Gym, and our model is built using Python and TensorFlow.

To determine the better case, we observed the return of the training. The return is based not only on the goal difference but also on ball possession, travel, and successful passes.

Results:

The results have shown that cooperative performance is slightly better than that of independent agents with some outliers. The cooperative agents in the counterattack scenario had a challenging time scoring a goal while the independent agent was able to possess the ball more often in the run to score scenario. The other scenarios showed that the cooperative agents were able to hold on their own. We also showed the entropy for all the scenarios. The entropies are shown because the higher the entropy is, the more random the actions will be. This allows the agent to use other actions to explore better ways to increase the return. The plots for the independent agent's scenarios are shown in the following:

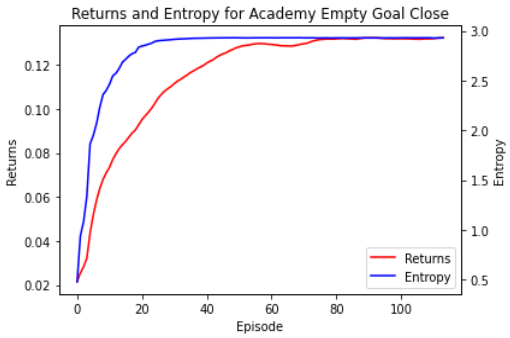


Figure 1a. Returns and Entropy for Academy Empty Goal Close

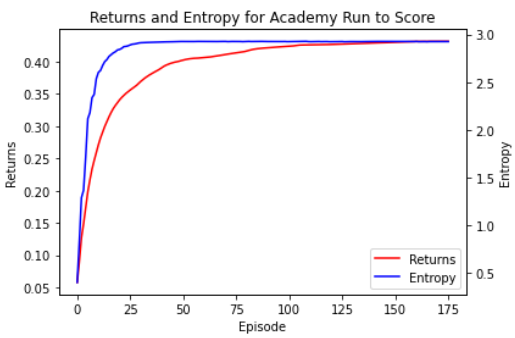


Figure 1b. Returns and Entropy for Academy Run to Score

The plots for the cooperative agent's scenario are shown as follows:

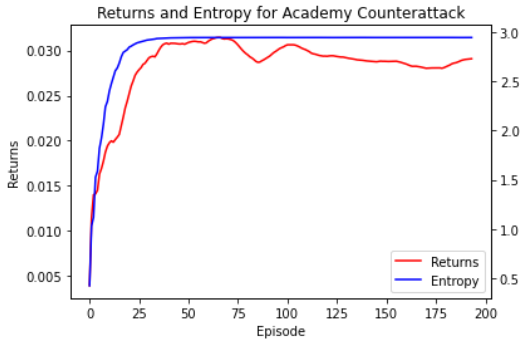


Figure 2a. Returns and Entropy for Academy Counterattack

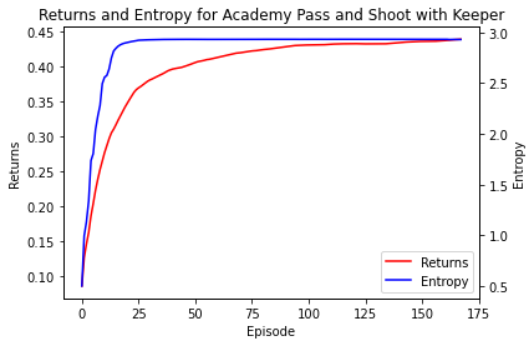
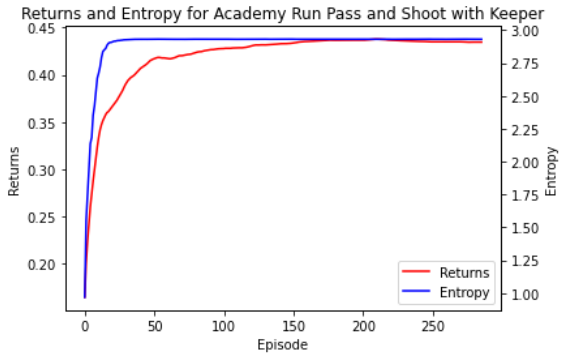


Figure 2b. Returns and Entropy for Academy Pass and Shoot with Keeper



3a. Returns and Entropy for Run Pass and Shoot w/ Keeper

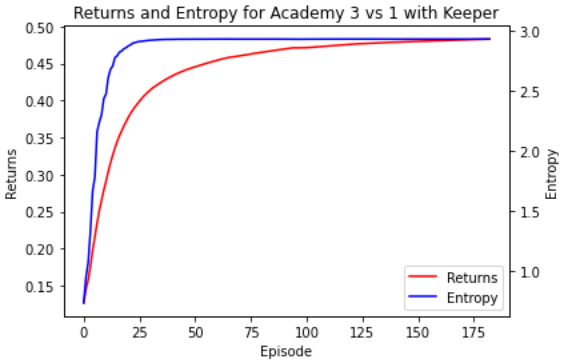


Figure 3b. Returns and Entropy for 3 vs 1 w/ Keeper

Conclusion and Future Work:

- As mentioned in the issues, the cooperative multi-agent RL takes a long time to achieve consensus. There might be better way to initialize weights of the neural network.

- The base paper [1] utilizes cooperative learning through sharing of rewards as well as states in addition to the policies. We have not utilized the sharing of rewards and states as this is skewing the game. In real life, one player will not know where exactly all his teammates are located on the field. He has only partial observability through his peripheral vision. As the states of the other agents are only partially observable to one agent, one extension of our work is to incorporate POMDP (Partially Observable Markov Decision Processes) to the given football environment.

- Another extension of the work is to try out the coordinated learning approach, where a single agent tries to learn for all N agents, i.e., a single agent outputs N actions instead of just one. Again, we have not tried this approach as the states observable to different agents are not same. We found an interesting work from UC Berkeley on the same, which is listed in the references [6].

- One of the extensions possible is to try imitation learning (mimic human behavior) in multi-agent football. As professional football grows more strategic with strategies like Tiki-taka, Gegenpressing, counter attacking football etc., we could have an expert policy used to train the individual agents through imitation learning or behavior cloning.

- To conclude, we have implemented independent and cooperative multi-agent reinforcement learning using policy gradient algorithms like PPO and A3C on Google Research's football environment. The results have shown that cooperative performance is slightly better than that of independent agents with a trade-off in training time.

References:

[1] Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. Cooperative Agents. *Machine Learning Proceedings 1993*, 330–337. https://doi.org/10.1016/b978-1-55860-307-3.50049-6

[2] Zhang, K., Yang, Z., & Başar, T. (2021). Multi-Agent Reinforcement Learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321–384. https://doi.org/10.1007/978-3-030-60990-0_12

[3] Sutton, R. S., Bach, F., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT Press Ltd.

[4] Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., & Gelly, S. (2020). Google Research Football: A novel reinforcement learning environment. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(04), 4501–4510. https://doi.org/10.1609/aaai.v34i04.5878

[5] Google-Research. (2019, May 26). *Google-Research/Football: Check Out the new game server:* GitHub. Retrieved May 7, 2022, from https://github.com/google-research/football

[6] Parikh, A., Kamsetty, A., & Thakkar, K. (2019). *APARIKH98/multi-agent-coordination-google-football: Coordination between Deep RL agents for Virtual Football*. GitHub. Retrieved May 7, 2022, from https://github.com/aparikh98/Multi-Agent-Coordination-Google-Football