Reinforcement Learning Applications in Google Football Environment

Nikhil Karnwal **Aaron Luo** **Marcus Schaller**

Abstract

Advancements in AI have allowed for computer software that can play video games at levels more advanced than the most experienced humans. In this article we will research using Reinforcement Learning to create an algorithm that can successfully play soccer in the Google Research Football environment. To do so we will use existing techniques such as Actor Critic Methods to optimize our algorithm and save computational resources. If successful soccer coaches and players could simulate game states they could gain insightful information that could be used in their actual matches.

1 Introduction

The advancement of reinforcement learning (RL) has resulted in artificial intelligence (AI) that is capable of performing complex tasks at superhuman levels. AI algorithms are decades passed surpassing human intelligence in simple deterministic games such as chess and pong (1) . They are now proving to be more efficient in recent video games such as Dota 2 and Starcraft 2. These games have a sufficient degree of complexity and randomness and require optimized algorithms to efficiently simulate games. Characteristics such as Fog of War require the algorithm to succeed with incomplete information and require a large amount of computational power to competently play. Additionally, these games introduce multiple controllable characters and the AI must incorporate teamwork to successfully compete against human players (2). As AI has overtaken humans in these games, professional players and coaches are able to learn from well trained models and learn how to analytically improve their gameplay. Since AI is effective in showing how to better play video games, naturally the next step is to apply RL to sports to help teams gain an edge over their opponents. However, it is very difficult to simulate sports due to the physical world being naturally stochastic. This is the problem that Google AI and Manchester City Football Club has set out to solve with the release of Google Research Football.

2 Existing Works

Pong is an example of one of the easiest games for AI to play due to its limited movements and deterministic environment. RL research has since evolved to become superhuman in games such as Chess and Go (3). These games have much larger action sets and each game can have nearly infinite distinct paths. The next step was to implement AI in games that require cooperation between agents and have a much more stochastic environment such as Dota 2 . In order to successfully simulate these games optimization was done to save computational resources. Researchers at Open AI used a form a variant of Advantage Actor Critic to train their model to successfully play Dota 2(2). The work done in creating successful AI models in these games has inspired researchers to create models to simulate real life sports. Many limitations exist in creating such models, firstly, sports are extremely stochastic in nature and simulating a realistic environment can be very difficult. In addition, modeling such a complex simulation can require substantial computational resources due to the number of inputs required to accurately simulate the gameplay. Finally, simulating sports require advanced physics

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

simulators which often are unavailable for researchers. Google Research Football was introduced as a solution to this and is one of the first engines to support research in football/soccer.

3 Proposed Solution

To properly simulate real-world environments, we must introduce stochasticity to our simulated environment, such that the same action performed twice may not have the same result. We chose to use the Google Research Football Environment as it attempts to simulate a real soccer environment by simplifying the game into a set of moves centered around the agent who is currently controlling the ball. The engine characterizes all possible actions into 8 directions of movements as well as 4 ways to pass and shoot the ball. In addition, the simulator accounts for human stamina by allowing sprinting for a period of time depending on the agent's level of tiredness. This simplicity keeps computational costs low and makes it possible to develop a usable algorithm within a reasonable time frame with the amount of resources we have available (4). Because the game being played here is non-deterministic, there are virtually infinite possible game states and combinations of game states. Reinforcement Learning at its most basic form relies on assigning weight values to actions or states, a method which is computationally inefficient in this environment. Moreover, the stochasticity of the environment introduces uncertainty. We will be using the policy gradient method for incorporating stochasticity of the environment. Policy gradient updates its parameters by taking random samples which results in highly variation of policy distribution and cumulative reward values. Consequently, these issues lead to the instability and slow convergence of vanilla policy gradient method. To solve these shortcomings, we will use Advantage Actor-Critic methods (5) which reduces variance and instability by adjusting cumulative reward using different variants of baseline.

Figure 1: Google Research Football Game-play Environment

3.1 Policy Gradient

Policy gradient methods are used to solve Reinforcement Learning problems. The policy gradient methods goal is to model and optimize the policy for achieving maximum rewards when taking action at a certain state. The policy can be modeled using a set of parameters θ and can find optimal parameters by minimizing the loss computed from the overall expected reward received by taking actions.

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t\right) G_t \tag{1}$$

$$where\ G_t = \sum_{t'=t+1}^{T} \gamma^{t'-t-1} r_{t'} \tag{2}$$

3.2 Advantage Actor Critic

Reinforcement algorithm implements using the policy gradient method's update policy parameters through Monte Carlo updates which uses random sample trajectories. This inherently causes high variance in the probability distribution and cumulative reward value because each trajectory can have high deviation during training. High variance in probability can lead to noise in gradient and thus to non optimal solution. Advantage Actor Critic method addresses these issues by subtracting baseline from G_t which reduces variance. It determines how much better it is to take specific a action relative to average and this is known as Advantage. (6)

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta log \pi_\theta(a_t|s_t) A(a_t, s_t) \tag{3}$$

$$where,\ A(a_t, s_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t) \tag{4}$$

and, V is modeled over parameters v

3.3 Network Architecture

The network architecture we have elected to use will be modeled off the 6th place Kaggle entry for this challenge. This model is relatively simplistic when compared to other models that were used in this competition. This model consist of 1 actor per CPU core which simulate matches send roll outs to the central learner.The learner will then updates the policy using this data. The actor will wait until the learner has updated the policy and will resume training after it has received the new policy. (12)

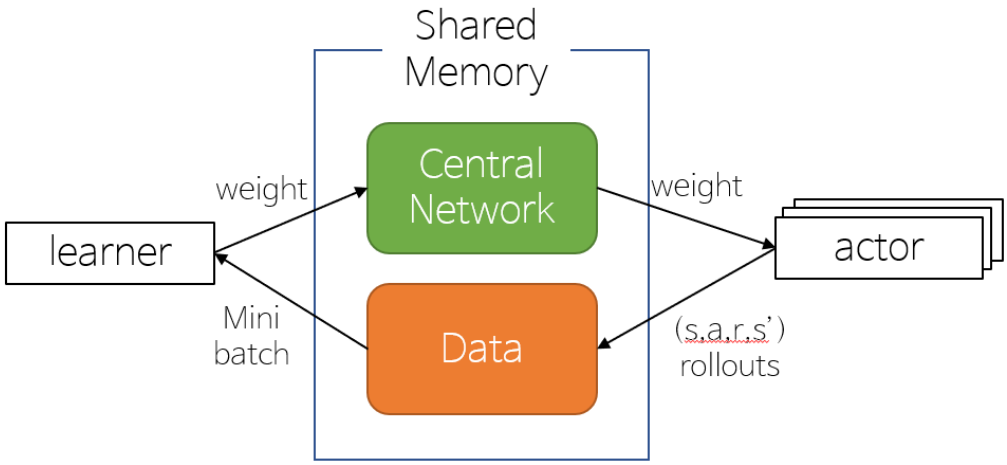


Figure 2: Architecture

4 Discussion

4.1 Actor Critic vs Policy Gradient

In order to validate that the Actor-Critic method would improve training run-time a test was done in Open AI Gym's simplistic Cart Pole environment in which the objective is to balance a pole upright earning +1 reward for every time step the pole remains upright. Both the Advantage Actor Critic as well as the Policy Gradient method was used with a learning rate of 0.001. The goal of this test was to prove that the Advantage Actor Critic method would indeed reach the 200 reward goal state at a quicker rate than the vanilla Policy Gradient algorithm. This can be seen clearly in Figure 3 as the A2C algorithm initially reaches the 200 point reward goal at episode 503 as compared to the Policy Gradient algorithm which reaches a reward of 200 at episode 851. (8).

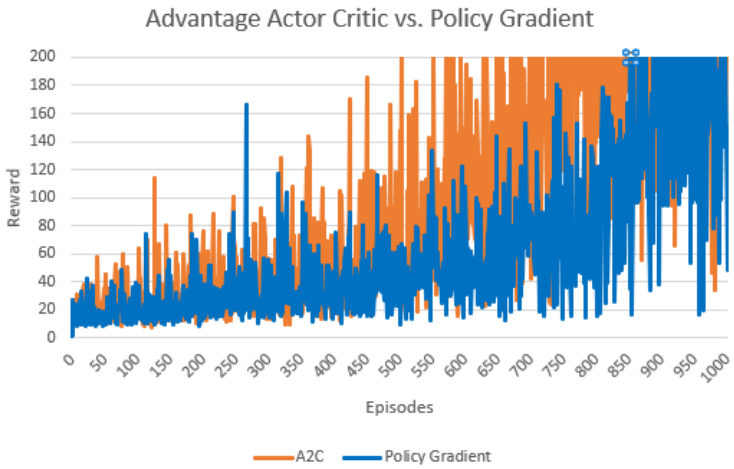


Figure 3: A2C vs Policy Gradient in Cart Pole Environment

In a simple environment such has the cart pole problem this faster convergence only accounts for a couple minutes of saved time between the Actor Critic and the Policy Gradient model. However when training a more complicated model such as a Google Football simulator team in which there are many actors and a larger selection of possible actions this time save will be crucial to ensuring that the model can be trained in an appropriate amount of time.

4.2 Advantages and Disadvantages of Advantage Actor Critic (A2C) Methods

As discussed previously, one of the advantages Actor Critic methods have over Policy Gradient is variance reduction. General variance reduction can be achieved by subtracting cumulative reward G_t by a baseline $b(s_t)$. A2C methods implement this idea by replacing the cumulative reward value G_t with an advantage value $A(a_t, s_t)$. This value is estimated by the "Critic" by subtracting the action value, or Q-value $Q_w(s_t, a_t)$ by the state value, or V-value $V_v(s_t)$(6). Below is a diagram of the Actor-Critic structure, with the Value function being the Critic model.

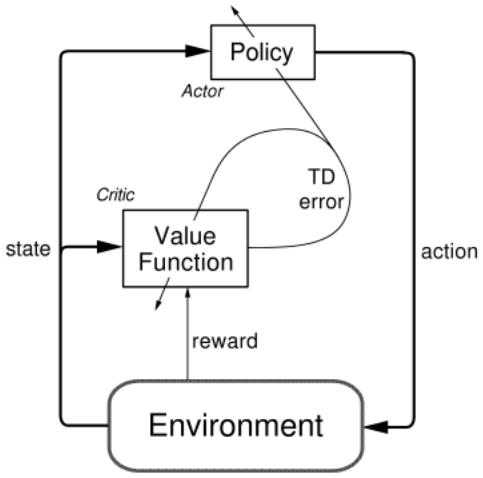


Figure 4: Basic Actor Critic Model (9)

The way the advantage value is calculated is important when discussing the global reward problem, and how A2C methods handles it better than policy gradient methods. The global reward problem manifests in games and activities that involve cooperative entities that work towards a singular goal, as it can be difficult to decide how the reward should be distributed. In policy gradient methods, the reward is only granted at the final sequence, which makes it difficult to assess the contributions of

each individual agent. One could adopt a shaped reward approach, that is, assigning the reward to all agents equally, but such a method could lead to slow convergence, as it can be difficult to assess the how positive or negative individual actions in specific states are until a large amount of games are played. A2C methods solve this problem by updating its parameters after each step as well as considering the value of actions given the environment state they occur in, making it easier to assess the individual contributions of each agent. This makes it possible to implement the local reward approach, or weighted reward distribution. However, this also plays into the main disadvantage of A2C methods in that it takes a lot more time to calculate estimated reward values and thus reach ground truth values (10).

4.3 Novelty

Because Actor Critic and its variations are well established in Reinforcement Learning, it is difficult to establish novelty in terms of algorithms used without coming up with an entirely new implementation. Novelty can be found in other things, however, specifically in the way we shape our rewards. Anyone who has watched or played soccer before knows that actual goals are rather uncommon. To account for this sparsity, we will be assigning awards for the following actions. Point values for each are subject to change.

1. Successful Goal: +10, Getting Scored On: -10
2. Successful Pass: +0.1, Unsuccessful Pass: -0.1
3. Time Spent With Ball Control
    - +0.0001/step when Inside Own Penalty Box
    - +0.0002/step when Inside Own Side of Field
    - +0.00025/step when In center of field
    - +0.0003/step when Inside Opponent Side of Field
    - +0.0005/step when Inside Opponent Penalty Box
4. Time Spent Without Ball Control
    - -0.0005/step when Inside Own Penalty Box
    - -0.0003/step when Inside Own Side of Field
    - -0.00025/step when In center of field
    - -0.0002/step when Inside Opponent Side of Field
    - -0.0001/step when Inside Opponent Penalty Box
5. Gaining Control Over Ball: +0.1, Losing Control Over Ball: -0.1

In the Kaggle competition this project is based on, many contestants had similar reward shaping ideas. The first place team, like us, assigned rewards for passing as well as gaining and retaining possession of the ball(11). The sixth place team also assigned rewards based on the current position of the ball(12). However, unlike other teams, we will not explicitly deduct points for penalties such as fouls or kicking the ball out of bounds. This is because there are situations where doing these things can be beneficial, such as kicking a ball out of bounds to slow down the opponent's offense or fouling a player to force a pass. We do not want to discourage these actions outright; rather, we want the algorithm to figure out when performing these actions are more beneficial than harmful. Another reward that other contestants don't appear to assign is shooting an unsuccessful shot. Landing a successful goal is typically uncommon, so this comes as an alternate way to encourage shooting. This can be somewhat dangerous, however, as including this as a reward may encourage taking poor but close shots. This reward may be removed as the project develops should this problem manifest.

5 Experiments and Dataset

For this project, we will be demonstrating the effectiveness of Actor Critic methods over Policy Gradient methods in complex game environments by making two different reinforcement models: one that utilizes the vanilla policy gradient algorithm, and one that utilizes the advantage actor critic algorithm. We do this to support the earlier discussions comparing A2C and policy gradients with additional empirical evidence, as well as to gain a better understanding of the implementation of

these algorithms. For this project, we will be training the model against the built-in hard AI. Much of the code used for this project is based in and taken from the 6th place winner of the Google Research Football Kaggle Competition. (12)

First, we preprocess the raw observations of each game state. Raw observations consist of location information such as the 2D coordinates of each left team and right team players, the 3D coordinates of the ball, and the direction each player is moving in. They also contain player and ball state information such as which player currently has the ball, which team has control over the ball, which players have been kicked out of the game, the tiredness of each player, and each player's roles (goalie, forward, etc.). For more details on these raw observations, visit this url: `https://github.com/google-research/football/blob/master/gfootball/doc/observation.md`.
We process these raw observations by compiling them into relevant information, namely the position and state of the allied player who has the ball, the position and movement information of the ball, position and state of everyone on the allied team, the position and state of everyone on the opposing team, and the position of every player on the field relative to the ball. During this process we will also be performing action blocking calculations, that is, locking out certain action choices from being chosen depending on such things as the current position of the ball or who has possession of the ball. So for example, shooting is blocked if the ball is not within the penalty zone of the field, and passing is blocked if the ball is not in possession. Note that, all preprocessing actually takes place during training, as we are making decisions and getting the output observations of these decision in real-time. Each new observation is processed before being placed into the network.

Next, we set up a Policy Gradient algorithm. We first read in a raw observation then preprocess it. We put it into a 1D Convolutional Network, which outputs an array of probability weights for 11 different action options. By default, there are 19 available action options, but we group the eight movement options into one action. If the movement option is selected, then we have a separate set of probability weights for choosing one of eight movement options that is also trained by the Convolutional Network. After an action is chosen, we insert this back into the environment to get a new observation as a result of that step. We use this new observation to calculate rewards as well as the vanilla loss optimization function, described in Section 3.1. We then loop back to the beginning of the algorithm, using this new observation as input.

To develop an A2C algorithm we, merely need to add a Critic to our Policy Gradient algorithm. The Critic takes the action, the probability of said action, the current and previous observations, and rewards. It then uses these values to calculate its own gradient ascent optimization function. In this case, we are using the Proximal Policy Optimization (PPO) algorithm, which means our critic function will be back-propagating over multiple epochs of minibatch updates. The optimized model is then passed back into the actor for further training.

To evaluate performance, we will be tracking the win-rate, score, and reward of the model over the course of 160 games, with each game consisting of 3000 observation states.

6 Results

The following is a visualization of one frame of a game. The left team, represented by orange numbers is the trained agent, while the right team, represented by the blue numbers, is the hard mode AI opponent.

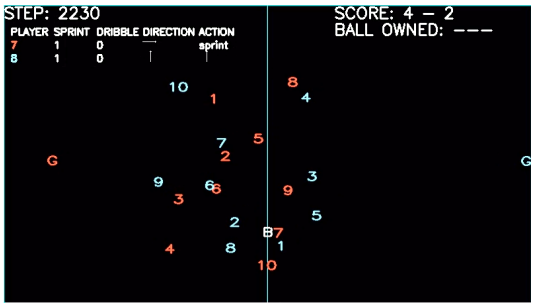
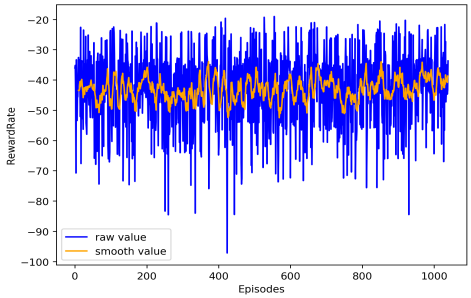
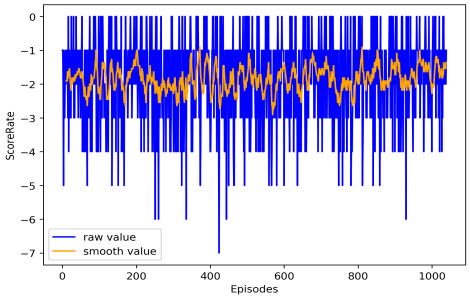


Figure 5: Visualized Results



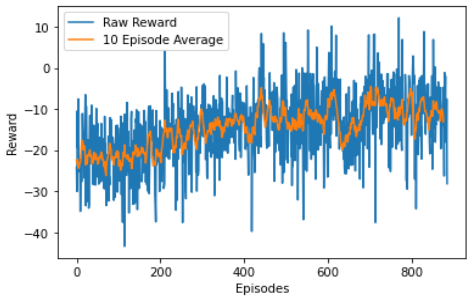
(a) Reward Value Over Games Played



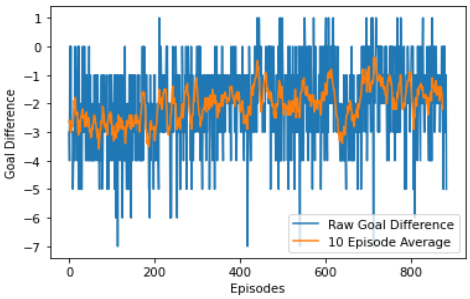
(b) Score Difference Over Games Played

Figure 6: Policy Gradient Reward and Score Difference, No Pretrained Weights

This graph represents the reward and score difference over 700 games of an agent using a policy gradient algorithm, trained without pretrained starting weights. To clarify, score difference refers to the difference in the amount of goals scored by the allied team and the amount of goals scored by the opposing team. Positive scores mean the game was won, negative scores mean the game was lost. Also, raw values refers to raw output datapoints, while smoothed values refers to the average of the said values over every 10 games. As demonstrated by the graph, the policy gradient agent doesn't seem to be able to win even one game over 1000 games, and overall doesn't see an improvement in performance over time.



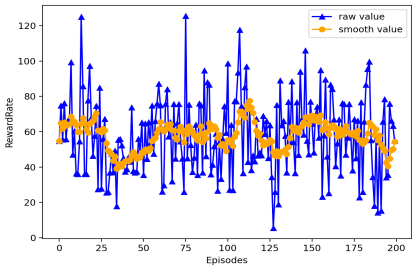
(a) Reward Value Over Games Played



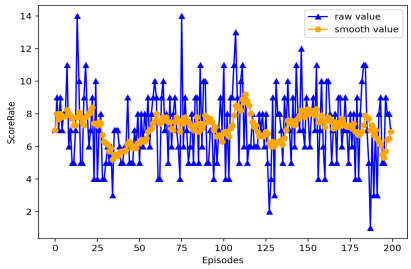
(b) Score Difference Over Games Played

Figure 7: Advantage Actor Critic Reward and Score Difference, No Pretrained Weights

We contrast this with our Advantage Actor Critic model, which also doesn't seem to pull out any wins over around 900 games but does at least demonstrate improvement in performance over time.



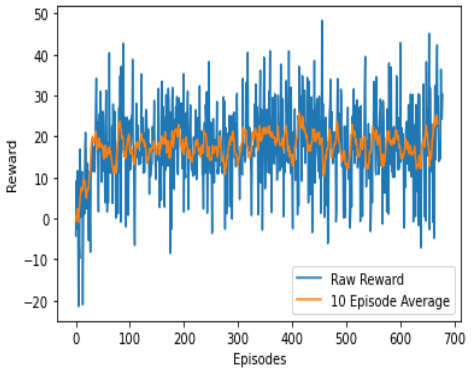
(a) Reward Value Over Games Played



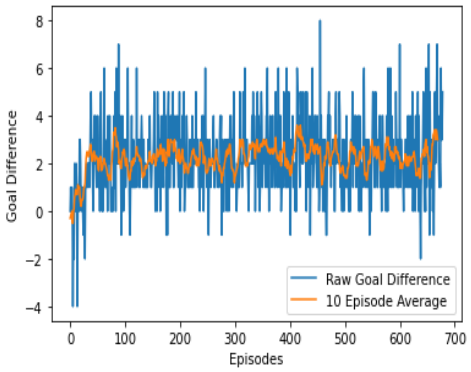
(b) Score Difference Over Games Played

Figure 8: Policy Gradient Reward and Score Difference, Pretrained Weights

Here is our Policy Gradient model trained on top of pretrained weights. Given that starting from pretrained weights is essentially as a continuation of training from previous models, we can see that the Policy Gradient model does eventually converge at a positive winrate given enough games.



(a) Reward Value Over Games Played



(b) Score Difference Over Games Played

Figure 9: Advantage Actor Critic Reward and Score Difference, Pretrained Weights

When we look at our Advantage Actor Critic model trained on top of pretrained weights, we can see the same phenomenon occurring here as with the Policy Gradient model above. Note that our Actor Critic model has been trained over significantly more games due to time constraints.

6.1 Analysis of Results

As discussed previously, the primary disadvantage of policy gradient is its high variance, such that it converges slower than other reinforcement methods. It is possible that this is the case here, and that given enough games, policy gradient will start to win games over the hard AI. However, it may also be the case that the policy gradient has already converged, as evidenced by the fact that the score and reward value doesn't seem to move very much over time. Answering this question would likely require further training, something that we unfortunately don't have enough time for within the scope of this project, as this model had already been trained for around six hours. Given the fact that our A2C model does eventually start to trend upwards after around 200 games, in addition to the knowledge that policy gradient models converge slower than Actor Critic models, we can safely assume that the former is probably more likely. The fact that our policy gradient model trained from pretrained weights returns positive scores supports this assumption.

Based on our results, we can also declare that our novel rewards policy does have a positive relationship with improved model performance. When analyzing the graphs, we can see that smoothed datapoints of reward values and score difference almost have a one-to-one correlation, suggesting that following reinforced behaviors leads to the model winning more games, or at least losing by less points. This however, may simply be a result of the reward model giving points for winning games and deducting points for losing games. In future tests, we can remove winning and losing from the reward model to observe the effect short term reward systems and to see if such systems are even necessary to begin with.

7 Conclusion

The purpose of this project was to demonstrate the effectiveness of Actor Critic models over policy gradient models in high complexity reinforcement environments. We knew prior to experimentation that policy gradient methods tend to have higher variance due to the randomness of its Monte Carlo update policy. Initial experimentation using a simple pole cart model demonstrated this idea, with the A2C model converging faster than the policy gradient model. Further experimentation with a Google Football environment reinforced this further, with policy gradient not even converging after many hours of training, and A2C converging within a much more reasonable time frame. In conclusion, we can assert that A2C models are much more effective in reinforcement learning than policy gradient ones, as they reach better performance over the course of fewer games due to lower variance.

References

[1] Campbell, M., Hoane, A. and Hsu, F., 2002. Deep Blue. Artificial Intelligence, 134(1-2), pp.57-83.

[2] Berner, Christopher & Brockman, Greg & Chan, Brooke & Cheung, Vicki & Dębiak, Przemysław & Dennison, Christy & Farhi, David & Fischer, Quirin & Hashme, Shariq & Hesse, Chris & Józefowicz, Rafal & Gray, Scott & Olsson, Catherine & Pachocki, Jakub & Petrov, Michael & Pinto, Henrique & Raiman, Jonathan & Salimans, Tim & Schlatter, Jeremy & Zhang, Susan. (2019). Dota 2 with Large Scale Deep Reinforcement Learning.

[3] Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D. Mastering the game of Go with deep neural networks and tree search. Nature. 2016 Jan 28;529(7587):484-9. doi: 10.1038/nature16961. PMID: 26819042.

[4] Kurach, K., Raichuk, A., Stańczyk, P., Zając, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O. and Gelly, S., 2020. Google Research Football: A Novel Reinforcement Learning Environment. Proceedings of the AAAI Conference on Artificial Intelligence, 34(04), pp.4501-4510.

[5] https://arxiv.org/abs/1602.01783v2 , (Async Advantage Actor Critic , A3C)

[6] Yoon, C., 2019. Understanding Actor Critic Methods. [online] Medium. Available at: https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f .

[7] "Google Research Football with Manchester City F.C.," Kaggle. [Online]. Available: https://www.kaggle.com/c/google-football/discussion/203412. [Accessed: 18-Feb-2021].

[8] A. G. Barto, "A toolkit for developing and comparing reinforcement learning algorithms," Open AI Gym. [Online]. Available: https://gym.openai.com/envs/CartPole-v1/. [Accessed: 17-Feb-2021].

[9] Lee, Mark. "6.6 Actor-Critic Methods." Incomplete Ideas, 4 Jan. 2005, incompleteideas.net/book/first/ebook/node66.html#:%7E:text=The%20policy%20structure%20is%20known,being%20followed%20by%20the%20actor.

[10] Keneshloo, Yaser Shi, Tian Reddy, Chandan Ramakrishnan, Naren. (2018). Deep Reinforcement Learning For Sequence to Sequence Models.

[11] "WeKick: Temporary 1st Place Solution." Kaggle, www.kaggle.com/c/google-football/discussion/202232.

[12] "Solutions of Team 'Liveinparis' with Codes (6th Place)." Kaggle, www.kaggle.com/c/google-football/discussion/201376.