CS 285 Final Project, Fall 2019

Coordination between Deep RL Agents for Virtual Football

Aakash Parikh, Amog Kamsetty, Kedar Thakkar

Extended Abstract:

In this paper, we take a look at various multi-agent reinforcement learning methods and their benefits within the context of the Google Research Football environment. To begin with, some context is required to describe the environment we worked within. The main function of the Google Research Football environment is to provide a simulator for reinforcement learning in an 11v11 football game. The environment, however, does also provide the ability to create custom simulations and has several pre-built scenarios to experiment in. These smaller environments can help with quicker iteration and experimentation due to the reduced compute cost. In our paper, we experimented on a combination of these smaller scenarios provided to us as well as our own custom-built scenarios in order to get a wide range of results to draw conclusions from.

We set up several different experiments in order to determine the performance of various methods for multi-agent RL. By default, the environment allows for an agent to control a single player on the field, but through modification we were able to allow for N different agents to control separate players on the field and be trained together. In another experiment, we instead had these N agents controlling the same players but trained them only individually (other players controlled by rule-based bots). Finally, in a third variant we had a single agent jointly controlling all N players and outputting N actions at each iteration. We then applied these experimental setups to the different scenarios we built, taking note of initial results and fine-tuning the algorithm as results came in.

From the data obtained, we were able to conclude that controlling N different agents that were trained simultaneously yielded the most stable, high-performance results. While both other methods had similar performance peaks to that of N simultaneous agents, in the case of a single global agent, the algorithm took a long time to stabilize or did not stabilize, and in the case of N agents trained individually, we saw much more variance upon evaluation as compared to the case in which they were trained simultaneously. Therefore, we found that controlling multiple agents and training them simultaneously was the best path going forward and continuing into extended work.

With regards to next steps, our primary aim would be to extend the idea of using multiple agents to incorporate defenders as our current setup only trains attackers. Doing so would yield greater variety in states encountered by the agents, resulting in more robust, generalizable agents.

Introduction and Related Work:

Motivation:

There are many tasks where collaboration and cooperation between agents is beneficial such as multi-player games or driving. In fact, it is predicted that in the future, many of the cars on the road will be autonomous [1]. Therefore it is crucial that we investigate the best methods to enable collaboration between independent autonomous agents. While it is relatively easy for humans to understand how to collaborate with each other, there are many additional complications when extend this same principle to artificial intelligence. In this paper, we explore different methods for multi-agent reinforcement learning on the Google Football Environment, and explore the tradeoffs for each approach.

Google Football:

Google AI research recently released an environment to simulate a football (soccer) game optimized for reinforcement learning work [2]. Their environment exposes a full 11-11 simulation as well as drills and mini-games such as corner kicks, counter attacks, and various game like scenarios. The game of soccer provides a great environment to explore multi-agent learning because of the necessary cooperation needed to perform well in the game. Each agent/player has a unique, specialized role which they must learn how to perform well in order to do well in the game. The complex structure of the game along with the intricacies involved with learning how to cooperate makes this environment a great candidate for multi-agent reinforcement learning experiments.

Coordination for RL agents:

There are various methods for multi-agent reinforcement learning that we describe below.

Centralized Learning: In this approach, a single controller controls all agents in the environment. The agents are not treated as independent, and this global controller takes in observations for all agents and outputs a joint action for all of them [3]. The benefit of this approach is that there is tight coordination between all of the agents in the environment. Because the global controller outputs a joint action for all the agents, the dynamics of the environment is stationary. However, the downside of this approach is that the state space and action space grows exponentially with the number of agents. For example, if there are A possible actions and n different agents, then the action space for the global controller has an action space of A^n. Thus the centralized learning approach is infeasible for situations where there are a large number of agents.

Concurrent Learning: In this approach, each agent in the environment concurrently learn their own independent policy [3]. Each agent has its own observation space and independently decide on actions to take. Since the agents are collaborating with each other, the reward is shared amongst all agents. In this method, the agents can learn a unique specialized responsibility since they learn their own policy, but are still incentivized to act in the best interest of the team since the reward is shared among all agents. This results in an advantage since this method best mimics how humans collaborate with each other and effectively distributes certain responsibilities amongst all of the agents. For example, in a soccer environment, certain agents can learn to take on a more defensive role, and other agents can learn to take on a more offensive one. In addition, since the agents all learn independently, there is no communication between the agents, meaning this approach performs well if there is a high bandwidth cost. However, the negatives of this approach is that the dynamics of the environment is not stationary from the perspective of each agent since each agent's observations are influenced by the actions of other agents. This could potentially limit learning with a large number of agents.

Parameter Sharing: The parameter sharing approach is a combination of the prior two methods [4]. The learning is done in a centralized mechanism where a single model is trained using the experiences of all the agents. However during the control phase, each agent act independently and the parameters of the model are broadcasted to all the agents. Though they have the same parameters, the agents make different decisions because each agent has different observations. This means that a single policy is learned during training time, but during evaluation each agent executes the policy with their independent observations. The advantage of this approach is that it is sample efficient since the experiences of all the agents can be used to train the single model. However, this also means that the states and rewards in addition to the actions have to be sent to the global model for each agent, thus making this method infeasible with limited bandwidth.

PPO:

The reinforcement learning algorithm that we use for all of our experiments is Proximal Policy Optimization, or PPO [5]. The PPO algorithm was first introduced by OpenAI in 2017 and is inspired from policy gradient methods.. PPO works by collecting small batches of the agent interacting with the environment, using these batches to update the policy, and then throwing away these batches. Since the actions that are executed are determined by only the current policy, PPO can be classified as an "on-policy" algorithm. The key contribution that PPO makes is that after each update, the new policy does not differ much from the previous policy. It ensures this by clipping the update to prevent it from being too large. This approach reduces the variance during training and makes the training process for the agent much smoother. Over the recent years, PPO has supplanted itself as the go-to reinforcement learning algorithm to use for most tasks. Thus, we decided to use PPO as well for our application.

Methodology:

We used the Google Football environment to design situations where players had to work together in a team to score a football. We designed three such situations with an ascending number of players to coordinate:

1. Single Attacking Players: Team A has one player that plays offense (Center Mid), and Team B has one defender (Center Back) and a Goalkeeper. The offensive player attempts to score a goal and gains reward as it advances the ball.
2. Two Attacking Players: Same as above except Team A has 2 offensive players (Left Mid, Right Mid).
3. Three Attacking Players: Same as above except Team A has 3 offensive players (Left Mid, Center Mid, and Right Mid).

In each of these cases, the attacking team begins with the ball in the center of the field. Each iteration ends after a set number of time steps or when either team scores, meaning that the only possible final scores are in {-1, 0, 1}. One timestep is equivalent to a 'frame' in which each agent receives an observation and each player (on both teams) executes a single action.
The observation space is an extraction that captures: the position of the (up to) 11 players on the left (attacking) team, the positions of the (up to) 11 players on the right (defensive) team, which player is active on the left team, and the position of the ball. The observation stores this for the last 4 time steps in a stacked manner to give an observation size of [96 x 72 x 4 x4], where 96 x 72 is the downsampled size of the field.

The action space for each agent is 21 discrete actions that include moving, passing, shooting, dribbling, specific variations of these, and also includes an idle action.

The attacking team was either controlled by:

1. N agents trained concurrently on the same observation space. Every agent shares the same observation space, and outputs its own action. We use the default action and observation space in this case.
2. N agents trained independently alongside 'bots' controlling the other attacking players. Each agent is trained on separate environments and uses the default action and observation space. The 'bots' are heuristic agents and do not learn. After training them individually, these agents are then evaluated together.
3. A single centralized agent that jointly controls the actions of the N players. We use a default observation space, but now have an action space of size N x 21.

We note that for the single attacking player situation, all of these reduce to a single agent controlling the action of a single player.
For consistency, we utilize the PPO2 algorithm from Stable Baselines in all these cases and keep the following constants:

Learning Rate	Entropy Coefficient	Discount Factor	Clip Range
0.00008	0.01	0.993	0.27

Table 1: Constants for PPO algorithm

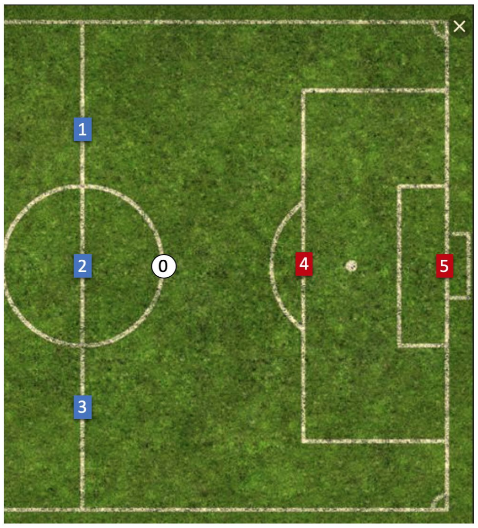


Figure 1: Initial Positions: Ball 0, Attacking Team: 1 (Left Mid), 2 (Center Mid), 3 (Right Mid), Defending team 4 (Center Back), 5 (Goalkeeper)

Results:

All graphs for the results are located in the Appendix.

The scenario with just a single offensive player can easily be learned by an agent. In our experiment it reaches its peak at around 100k timesteps of about 0.75 (average reward). In this 1v1 scenario, the agent is not able to exceed that limit.

Each of the 2v1 experiments takes considerably longer for an agent to learn. We do not reach a winning (0.5) result until about 800k timesteps for independently trained agents, 550k timesteps for simultaneously trained agents and 1.5M timesteps for a single agent. The simultaneously trained agents reach their peak of 0.65 and stabilizes there. While the individually trained players oscillate between a reward of 0.7 and 0.5., evaluated together, they reach a mean score of 0.655

over 200 evaluations, but the standard deviation of 0.486 is very high. The single agent does not stabilize within 2M timesteps and has a performance between 0.4 and 0.65 by the end of training.

The 3v1 scenario was evaluated over only 250k timesteps. When controlling a single player, the results varied depending on which player was controlled. The left player reached a peak of 0.9 in 200k timesteps, but did not stabilize, falling to as low as 0.6. The left player hit an average score of 0.5 after only 25k timesteps. The right player, took 60k timesteps to reach an average score of 0.5, but stabilized to an average reward of 0.7 at the end of the 250k timesteps. It also reached a maximum of 0.9 at 180k timesteps. The mid player, reached the highest peak of 0.95 at 155k timesteps and stabilized around 0.8 and 0.9 by the end of the training. It reached an average score of 0.5 at around 25k timesteps, when excluding an initial spike.

When evaluating the wings (left and right mid) together, with a heuristic-based bot controlling the middle player we reached an average score of 0.735 with a standard deviation of 0.45. When evaluating the left and middle agents together with a bot controlling the middle player we achieved an average reward of 0.79 and the standard deviation decreased to 0.41. When all three players were controlled by an agent, we reached a mean reward of 0.75 with a standard deviation of 0.43.

In simultaneously training 3 agents, each of the agents stabilized at an average reward of 0.8 after only 60k timesteps. A single agent controlling 3 players reached a score of 0.5 after 70k iterations and reached a peak average reward of 0.9 after 210k timesteps. This return remained variable between 0.6 and 0.9 by the end of training.

Detailed graphs for these results can be found in the appendix in *Figures 2 -11*.

Discussion:

From the data obtained, we were able to conclude that controlling N different agents that were trained simultaneously yielded the most stable, high-performance results. While both other methods had similar performance peaks to that of N simultaneous agents, in the case of a single global agent, the algorithm took a long time to stabilize or did not stabilize, and in the case of N agents trained individually, we saw much more variance upon evaluation as compared to the case in which they were trained simultaneously. Therefore, we found that controlling multiple agents and training them simultaneously was the best path going forward and continuing into extended work.

With regards to next steps, our primary aim would be to extend the idea of using multiple agents to incorporate defenders as our current setup only trains attackers. Doing so would yield greater variety in states encountered by the agents, resulting in more robust, generalizable agents.

Conclusion and Next Steps:

From our work, it is apparent that there are significant gains to be made through using the multi-agent setup. In the 2v1 scenarios in particular, the simultaneously trained agents peaked quicker and stabilized at that peak while the single agent failed to stabilize even after 2M timesteps. The results from individually trained agents were more promising than those of a single agent as well, but demonstrated more variance. This implies that training agents together rather than learning opposite a bot allowed for more stable results once evaluated. We expect the difference is due in part to the ways in which the agents trained individually learn to exploit the rule-based ways in which the bots they are trained alongside play. Separate agents trained simultaneously, however, are able to adapt and generalize better.

3v1 experiments yielded similar results as although a single agent controlling all three players had a similar peak in performance to three separate agents trained simultaneously, convergence was much quicker in the latter case (60k vs. 210k timesteps).

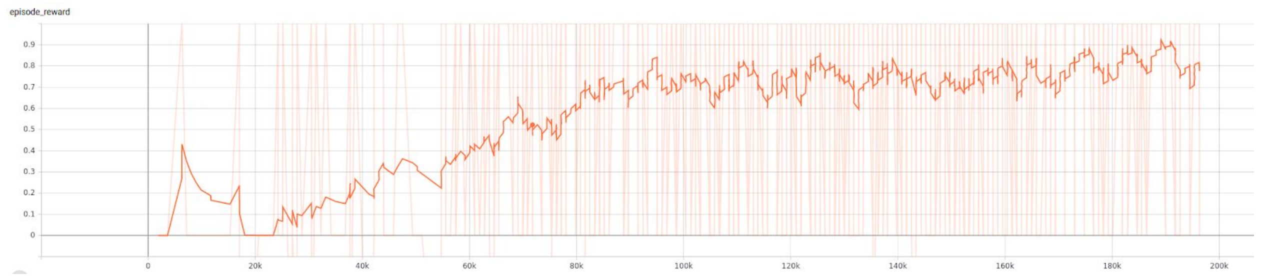
There are a number of next steps we could take from here, particularly with more time and computational resources. There are clearly gains to be made by training multiple agents simultaneously which we could take advantage of by training both attacking and defensive players together, rather than just attacking players. Currently, agents for attacking players are able to exploit defensive behavior by bots and would instead be able to learn more generally by seeing defensive variation. Such a change would introduce entropy and variety in the states seen by our agents which would allow them to improve much more quickly and to higher peaks.

Furthermore, we've currently only used PPO but could further finetune the algorithm for our use, particularly as we focus on a multiagent case. After doing so, we could expand this to include more agents in a full 11 v 11 game rather than restricting the space to the drills given.

Appendix

Figure 2:

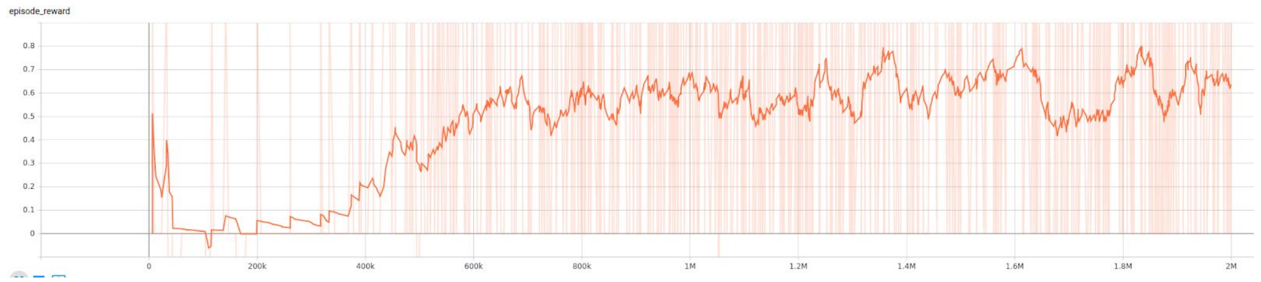
1v1 Single Agent, Single Player



Eval time: 11 min, 200K timesteps, 303 timesteps / second

Figure 3:

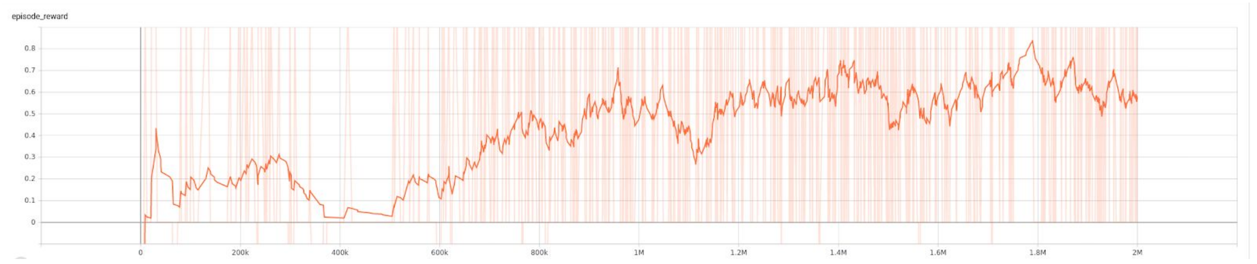
2v1 Single Agent controlling left player



Eval time: 1hr 33 min 2M iterations, 358 timesteps/second

Figure 4:

2v1 Single Agent controlling right player



Eval time: 1hr 28 min, 2M iterations, 379 timesteps/second

Table 2:

2v1 Evaluation of Separately trained agents

Iterations	Mean Reward	Reward Std Dev	Max Reward	Min Reward
200	0.655	0.486	1.0	-1.

Figure 5:

2v1 Two Agents Simultaneously trained

Figure 5.1: Combined Rewards

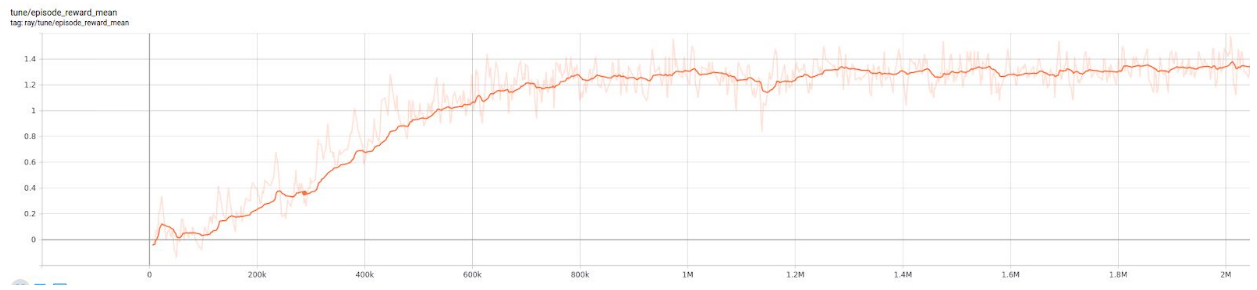


Figure 5.2: Left Player Rewards

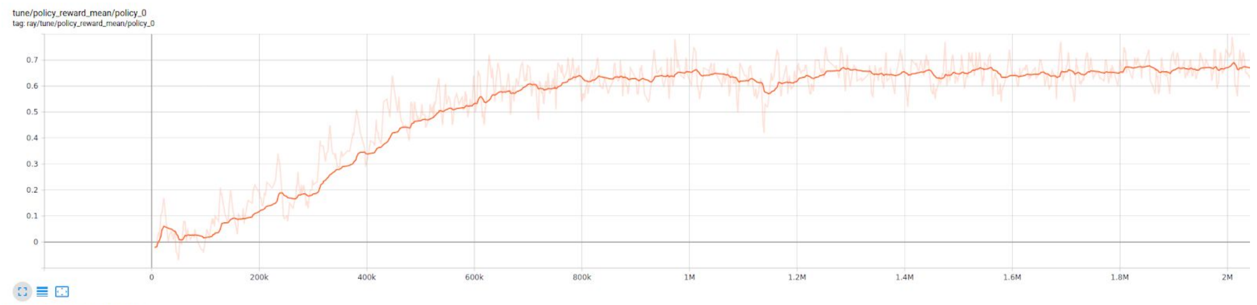
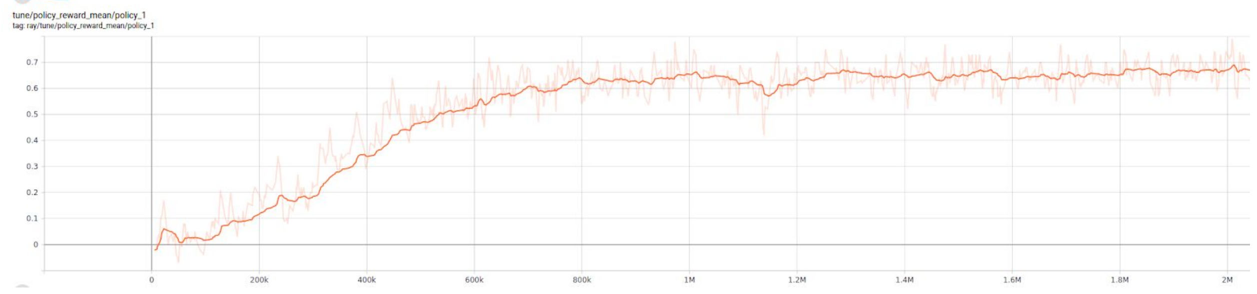


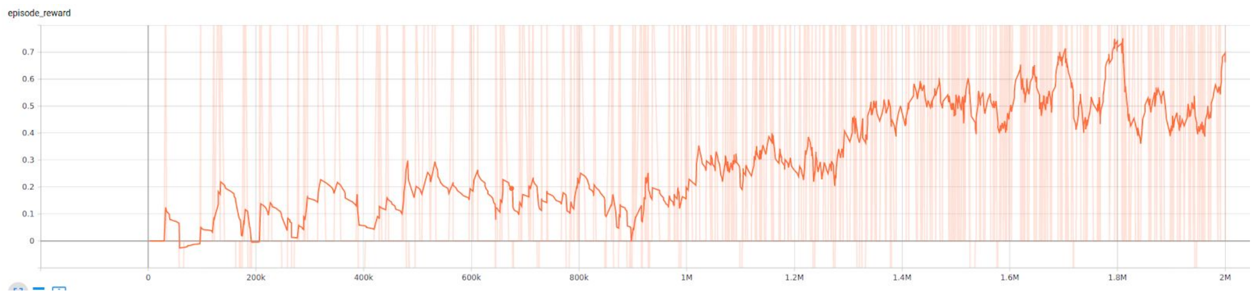
Figure 5.2 Right Player Rewards



Eval Time: 1hr 34 min 2M iterations, 354 timesteps /second

Figure 6

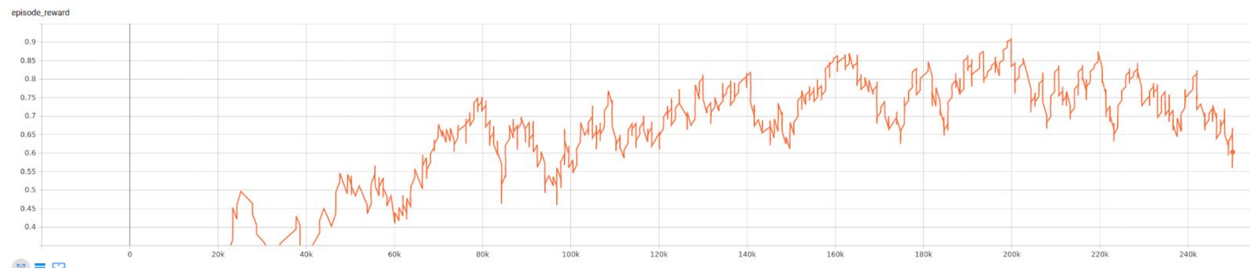
2v1 Single Agent Controlling 2 Players



Eval Time: 1hr 27 min 2M iterations, 383 timesteps/second

Figure 7

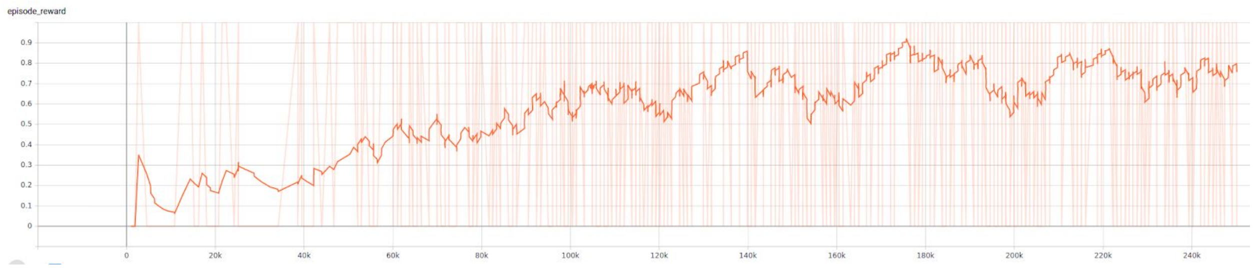
3v1 Single Agent controlling Left player



Eval time: 16 min 34 sec 250K iterations, 252 timesteps/second

Figure 8

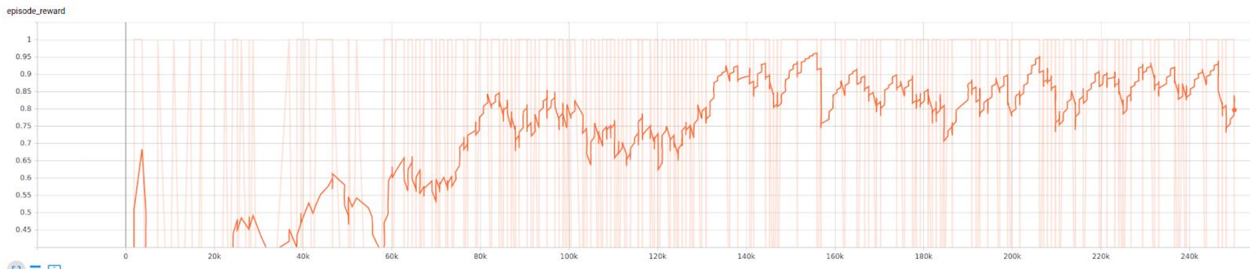
3v1 Single Agent controlling Right player



Eval Time: 15 min 42 sec 250K iterations, 265 timesteps/second

Figure 9

3v1 Single Agent controlling Middle player



Eval time: 17 min 2 sec, 250K iterations, 245 timesteps /second

Table 3:

3v1 Evaluation of Separately trained Left, Right agents

Iterations	Mean Reward	Reward Std Dev	Max Reward	Min Reward
200	0.735	0.453	1.0	-1.0

Table 4:

3v1 Evaluation of Separately trained Left, Middle agents

Iterations	Mean Reward	Reward Std Dev	Max Reward	Min Reward

200	0.79	0.407	1.0	-1.0

Table 5:

3v1 Evaluation of Separately trained Left, Right agents

Iterations	Mean Reward	Reward Std Dev	Max Reward	Min Reward
200	0.750	0.433	1.0	-1.0

Figure 10

3v1 Three Agents Simultaneously trained

Figure 10.1 Combined Rewards

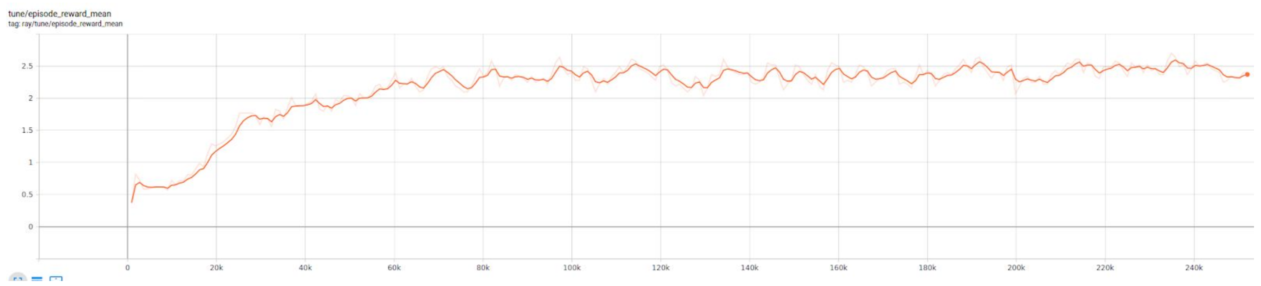
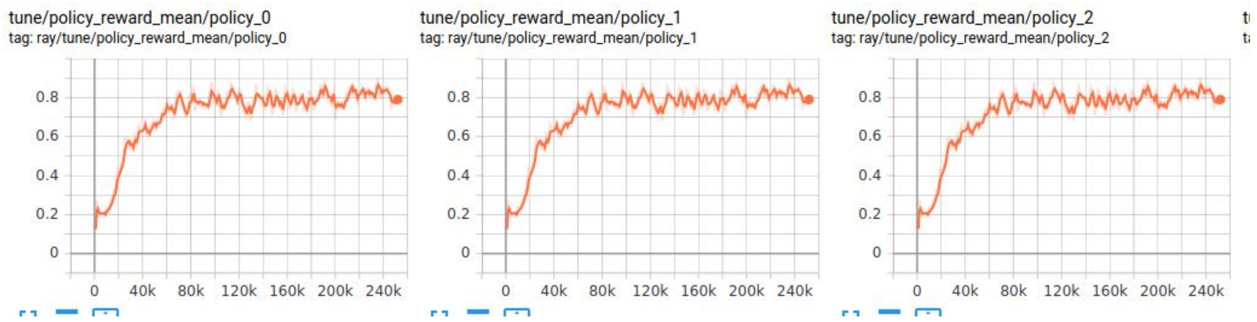


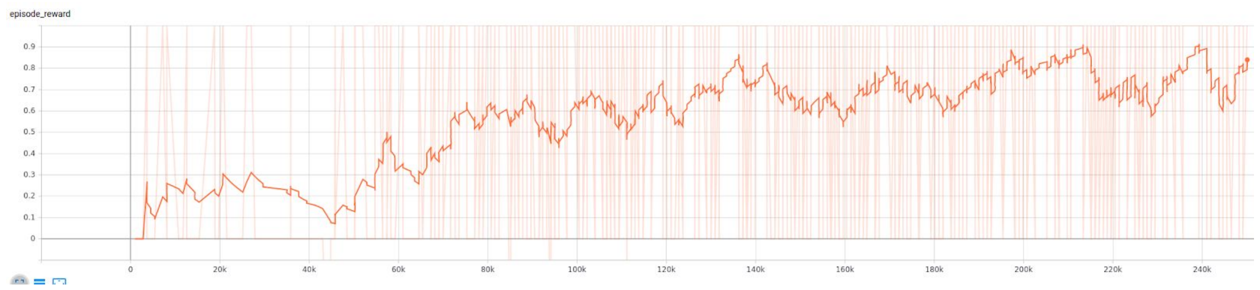
Figure 10.2 Individual Rewards for Left, Right, Middle players



Eval time: 22 min 35 seconds, 250K iterations, 185 timesteps/second

Figure 11

3v1 Single Agent controlling 3 players



Eval time: 17 min 5 seconds 250k Iterations, 245 timesteps/second

References:

[1] Kersten Heineke, Philipp Kampshoff, Armen Mkrtchyan and Emily Shao. When will the robots hit the road?, 2017.

[2] Karol Kurach, Anton Raichuk, Piotr Stan´czyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google research football: A novel reinforcement learning environment, 2019.

[3] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning, 2017. In *Adaptive Learning Agents Workshop*.

[4] Niranjan Balachandar, Justin Dieter, Govardana Sachithanandam Ramachandran. Collaboration of AI agents via cooperative multi-agent deep reinforcement learning, 2019.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.