CS7642 Project 3

Google Research Football

Wei En Chen (commit: `f8ca4fa6b53124c611fcabca3ff1d3a06707150b`)

***Abstract* – This paper demonstrates the use of centralized critic and custom reward shaping functions to improve upon baseline performance in Google Research Football RL challenge. It also includes key metrics showing actors learned teamwork as opposed to solo behaviours in baseline algorithms.**

I. INTRODUCTION

Google introduced an football (soccer) playing game environment [1] a few years ago for RL research. The full blown version can include 11 players vs 11 players in variety of game modes such as normal, goal kick, penalty, etc. The observation space includes ball, both team players, and match state related information in continuous space. Each player's action space includes movement, ball passing/shooting related values in discrete space. For this project, the number of players is reduced to 3 on each team and the goalkeeper is controlled by the game AI (built-in algorithm). The observation space is reduced to a 43-dimensional vector while the action space is kept the same, a 19-dimensional vector for each player. The type of game mode is reduced to normal, and the objective is to maximize win rate (i.e. kick the ball into opponent's net). The reward is sparse that when a team wins, each player gets +1.0 reward while losing team get -1.0, and the player in possession of the ball gets reward in increment as they advance to opponent's side up to +1.0 reward.

In this project, I intend to analyze the performance of the 3 baselines RL algorithms (PPO) [2] provided, and then implement other recent MARL (multi agent RL) algorithms such as QMIX [3] and CCPPO (centralized critic PPO), as well as looking at some different reward shaping methodology to increase team effort and enhance certain behaviours.

II. BASELINES

The three baselines provided [4] are all trained with PPO [2] algorithm, which utilizes clipped surrogate objective to stabilize learning. The idea behind it is that taking too big of a step during gradient descent can be undesirable so clipping can theoretically make the training more stable. It also uses an advantage function to reduce variance of the estimation which also helps with training stability.

The differences in the three baselines are hyper-parameters and model architecture related. I ran each baseline against the hardcoded AI with one thousand games, with alternating ball possession. The win rates are shown in Figure 1. The win rates were averaged by running 100 games for each baseline model. One can see that baseline 1 is on average the worst, and 3 can be the best. From Figure 2, the mean rewards of the training show that baseline 3 has the highest rewards. More runs are required to narrow down the error bars to see baseline 3 is likely the best among the three algorithms.

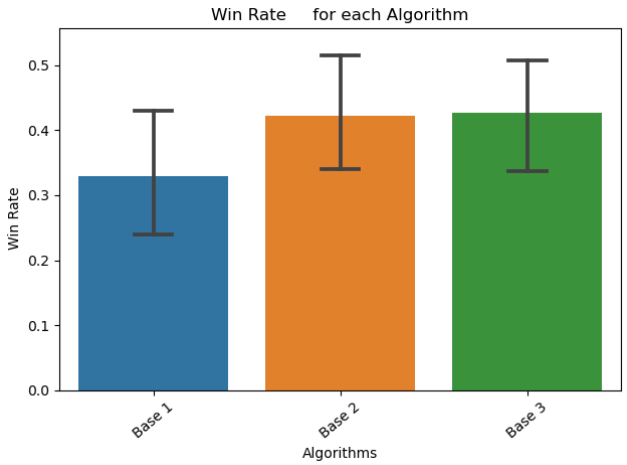

Figure 1 - Win rates for 3 baselines. 100 episodes per baseline were run to average.

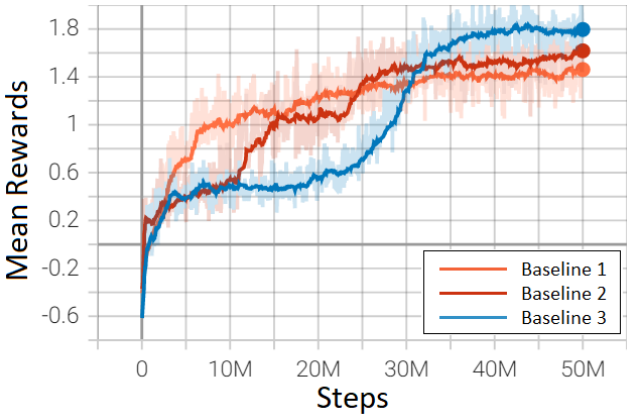

Figure 2 - Baseline training curves showing mean rewards of the team (i.e. average of sum of rewards of two players)

To visualize any behaviour difference among 3 baselines, I plotted the ball X-position before the team player attempts to shoot to score, shown in Figure 3. An interesting behaviour difference is that baseline 3 players take the ball a lot closer to the net before kicking the ball to score. It seems like the agents learned it's more effective to score if the ball is closer to the net before the kick. More

metrics will be compared between baselines and my implementation in later section.

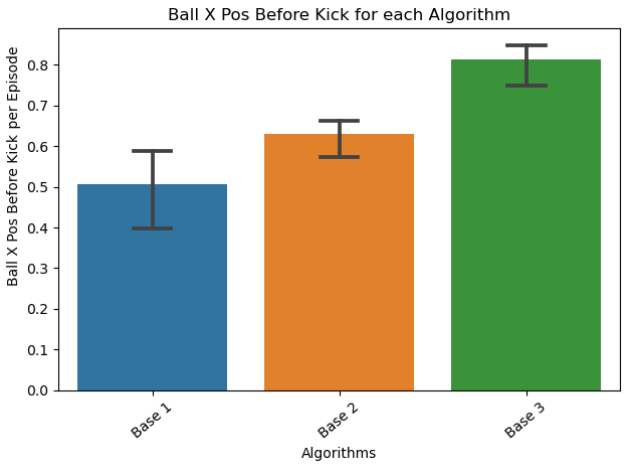

Figure 3 - Ball final X-position before a scoring shot is attempted by team player.

III. QMIX

The algorithm QMIX [3] aims to strike a learning methodology between independent Q-learning where each agent learns an individual action-value function independently and a fully centralized value function which acts as a overall critic and it optimizes the policies in actor-critic methodology. The former does not address how each agent can cooperate since they are learning separately, and the latter may require on-policy learning which is sample inefficient and increasing in number of agents can dramatically increase the complexity of centralized learning.

QMIX instead takes the approach to represent the overall Q as the sum of individual agent Q, and individual agents act based their own Q values. QMIX consists of individual Qs for each agent, and a mixing network that utilizes non-linear method (i.e. a feed-forward neural net takes agent network output as input and use non-negative weights to mix them) to combine individual Q into the overall Q. The mixing network is constrained to have only positive weights to represent centralized action value functions with factored representation [3].

I believe this concept may fit well with the google football challenge because there are two players on the same team and some interaction between teammates can be beneficial to maximize the reward. The algorithm is also off-policy which is more sample efficient. The overall architecture is shown in Figure 3 [3]. The illustration assumes partial observation so each agent's states are replaced with o_t^i.

The authors ensured the weights of the mixing network to be non-negative by training a hyper network to output the weights of the mixing network. The end of the hyper network is a ReLU activation so the output is always non-negative.

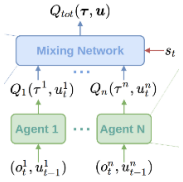

Figure 4 - Overall QMIX model architecture [3]

To conduct the google football training with QMIX, the QMIX algorithm from Ray is utilized [7]. To find a set of suitable hyperparameters, I used the tuning capability from Ray as well to include the hyperparameters: lr, gamma, mixing_embed_dim, optim_alpha, optim_eps, etc., a total of 13 hyperparameters. When hyperparameter search is running, RAM was quickly drained and eventually ran out. It was due to running algorithms in parallel which store lots of replay buffers. The maximum amount of buffer had to be reduced to avoid out-of-memory error. With limited time, the maximum number of steps were reduced to 2 millions with 100 runs for hyperparameter search. Once a best set of hyperparameters were found, I used it for final training of 20 million steps. The mean episode reward is shown in Figure 5, did not look very promising.

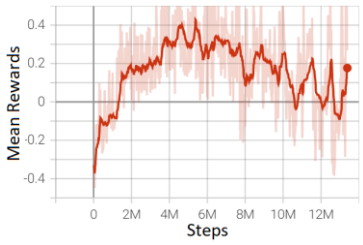

Figure 5 - QMIX training curve showing mean rewards of the team, the training was stopped early because it was not promising

The model was definitely learning up to half way through but then became volatile and overall trend declining in the mean reward plot. I suspect the hyperparameters tuning would require longer time steps than 2 millions to find a more stable training. The training speed was slow due to the inclusion of LSTM, and rerunning the tuning would take even longer. QMIX does have potential to be a good algorithm for google football, but due to the reasons listed above, I decided to give up QMIX.

IV. CCPPO AND REWARDS SHAPING

The next concept is to improve existing PPO implementation with multi-agent components. The key concept in PPO is clipped surrogate (trust region) objective, which is shown below equation.

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

The clipping avoids large unwanted gradients which stabilizes the learning. $\hat{A}$ is the generalized advantage estimate (GAE). Advantage estimate is essentially the expected value of a state subtracted by the expected value of a state with an action. The generalized version is to tackle bias-variance trade-offs since typically the advantage estimates are – estimates, which have built-in bias. To reduce bias, TD estimate can be utilized, but the extended terms will increase variance. GAE tweaks λ to balance bias-variance trade-offs [5, 6].

The implementation of the PPO is from Ray project and default project 3 documentation. The multi-agent component is modified from Ray example centralized_critic.py [7]. The centralized critic is developed with a neural net of some fully connected layers that takes multiple actors' observations and actions as input and outputs a scalar value. This value, instead of the default of a simple fully connection layer with a scalar output, is then used to calculate surrogate objective function. Since the input to the neural net is not simply individual player's observation and action, the hypothesis is for the neural net to learn some kind of overall strategy for individual members. Figure 6 shows single actor PPO schematic taken from [8]. To make it work for multi actor, $V_\mu(S_t)$ is estimated with observations and actions from all agents. There is one $L^V(\mu)$ for all actors and the value dictates how well the actors are doing as a team. For the football observations, every actor has total observation including opponents' movement so collecting everyone's observation is likely not going to be provide much advantage. However, having other agents' actions and having one critic instead of actor-critic pairs can help in a sense to maximize the team's reward instead of maximize individual actor's reward.

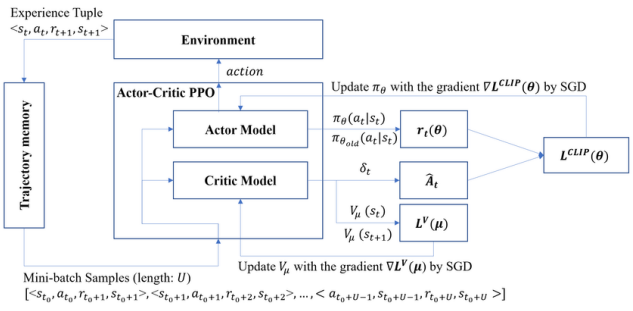

Figure 6 - PPO optimization algorithm schematic taken from [8]. This represent a single actor-critic pair PPO. For multi agent, $V_\mu(S_t)$ takes observation from all agents to estimate the value function.

In addition to CCPPO, I tested some different reward shaping. From observing several games, I found the baselines are amazing at scoring when the team has initial ball possessions, and very strong at undefeated rate. However, the baselines are not able to score a goal most of the time when the opponents have initial ball possessions. It is possible that the algorithms have not figured out an effective way to force a turnover, either from reward shaping, or team effort. To encourage CCPPO to behave in certain way, I proposed two different reward shaping methods on top of the default checkpoints and scoring:

1. Negative reward of -0.6 when losing ball possession to opponents, and positive reward of +0.4 when gaining possession from opponents. Reward only applies to the actor who loses and gains ball possession, not the whole team. The number of turnovers is limited to 300. The idea here is to direct the actors to keep ball possession, and also try to force a turnover when opponents have initial possession.
2. One time positive reward of +0.5 when the actor gains possession from opponents when they have the initial ball possession. The reward function here is more sparse than the first one because usually deep RL models learn better when rewards are sparse likely due to the way they learn. Having more reward rules may sound reasonable to humans, but the model may be limited to fewer ways to learn to win the game.

Both reward shaping functions are aiming to encourage the actors to get the ball from opponents through central critic, possibly some team work. These additional rewards are implemented by using Gym Wrapper class [9] to essentially add multiple layers of rewards if needed. In total there are three algorithms in the study: default CCPPO, CCPPO with reward #1, and CCPPO with reward #2. All of them are setup with hyperparameters from baseline 3, except for the central value network size.

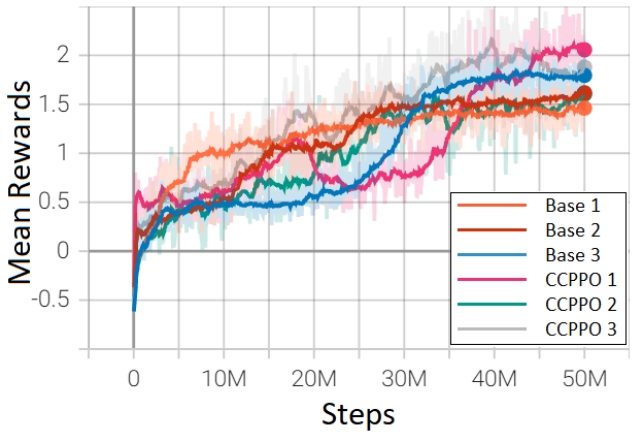

Figure 7 - Training curves of mean rewards, for all baselines and all CCPPO variations. Curves are smoothened for visualization.

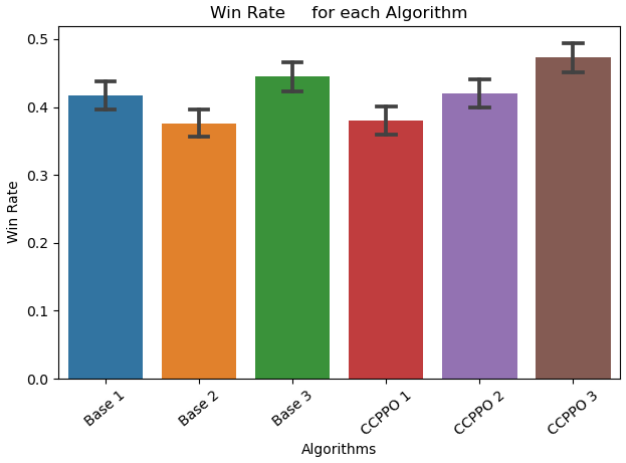

Figure 8 - Win rates among all algorithms. Win rate of 1.0 = 100%.

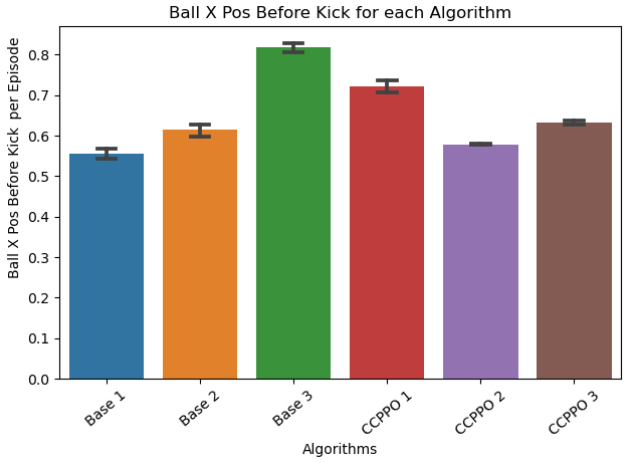

Figure 11 - Ball X-pos before kick to attempt score.

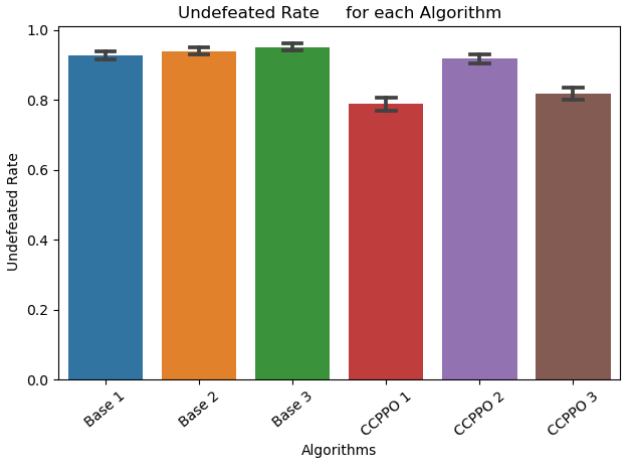

Figure 9 - Undefeated rate for all algorithms. Rate of 1.0=100%.

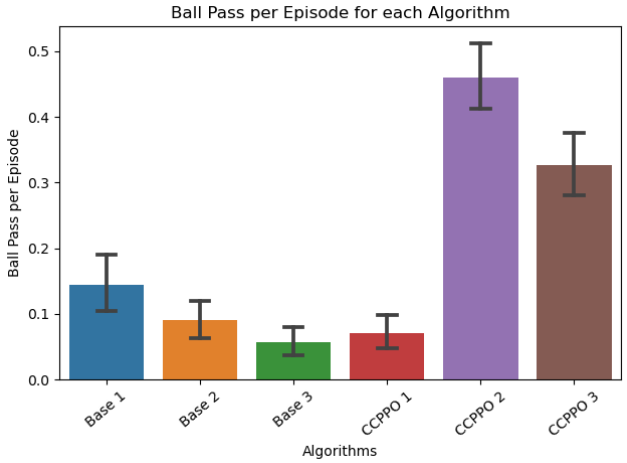

Figure 12 – Number of ball passes per episode.

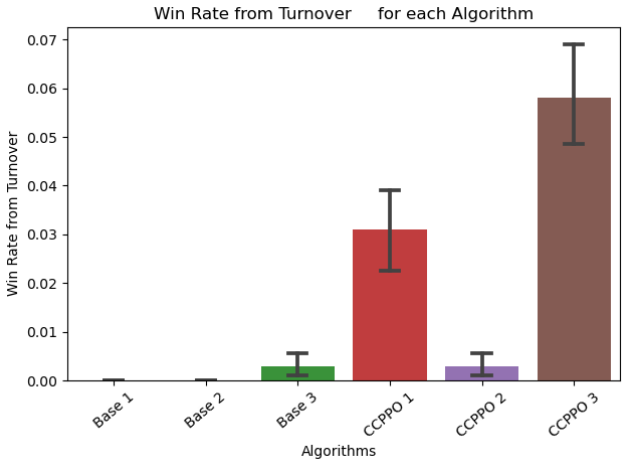

Figure 10 - Win Rate from forcing turnover. Rate of 1.0=100%.

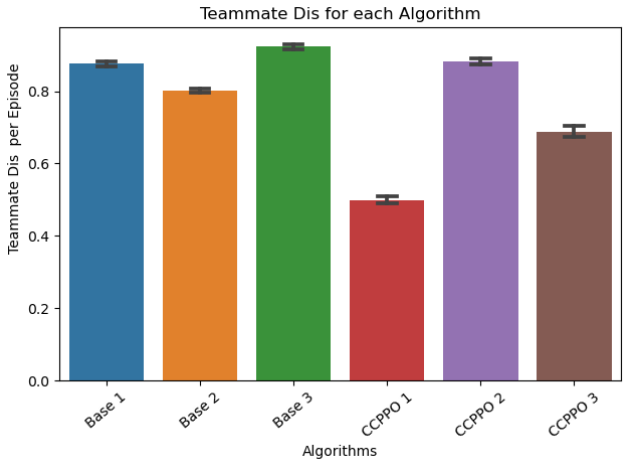

Figure 13 - Average teammate distance per episode.

The central value function network has two hidden layers of 256 and 128 sizes, with a ReLU activation in between. All of them ran for 50 million steps, to make comparison with baselines on even ground.

Figure 7 shows the team mean episode rewards throughout training for all baselines provided by the school course, and all three CCPPO variations, where CCPPO 1 is the default, CCPPO 2 is with reward shaping #1, and CCPPO 3 is with reward shaping #2. It is only meaningful to compare among baselines and CCPPO 1 because CCPPO 2 and 3 have different reward shaping functions. CCPPO 1 shows higher mean episode rewards than all three baselines. All the rest of the plots are made after the training is done. 2000 episodes are executed per algorithm to lower variances of the results. From the win rate Figure 8, we can see CCPPO 1 does pretty well, and CCPPO 3 surpasses baseline 3 win rate, approaching high 40%. For undefeated rate Figure 9, all 3 baselines are excellent, and all CCPPOs are not as good as the baselines. If CCPPO 3 win rate surpasses baseline 3 but CCPPO 3 undefeated rate is lower, it can only mean some of the win rates come from forcing a turnover, as seen in Figure 10 that CCPPO 1 and 3 are able to learn to force a turnover to increase rewards. They are also correlated with lower undefeated rate. It is possible that to force a turnover it does increase the risk of losing the game. CCPPO 2 failed to do so could be due to extra reward shaping functions. From Figure 11 we can see all CCPPO actors learn to take the ball close enough to the net before shoot to score.

For Figure 12 and Figure 13, they show how many passes (long, high, short) are made per episode, and how close are the teammates throughout the game, respectively. In general, CCPPOs with custom reward shaping learned to pass balls more often, possibly to avoid negative rewards when opponents take ball possession from our team (CCPPO 2) . CCPPOs also seem to learn to keep teammates in closer distance, possibly to help force a turnover, which requires team work.

All results shown from Figure 8 to Figure 13 have some variation to them, including the baselines. From Figure 8, baseline 1 has higher win rate than baseline 2, but when I ran for another 2000 episodes, I got baseline 1 with the lowest win rate and baseline 3 at the highest. The training curves also show raw data is in fact quite noisy, especially CCPPOs. I suspect the weights in the neural network make them more sensitive to change which leads to the volatility. It may be possible to reduce the noise by tuning the hyperparameters. If there is more time, I hope to look into hyperparameters more, relook at QMIX, algorithms other than PPO coupled with centralized critic, and self-play.

V. CONCLUSION

I successfully improved PPO win rate with CCPPO + custom reward shaping function. Part of the win rate improvement comes from making turnover possible which is almost nonexistent in baselines. It comes with a price of lower undefeated rate because it takes riskier moves to make turnovers so when they fail, they can lead to losses. The improved algorithm also displays increased ball passing and closer teammate distance, an important sign of team work. Riskier moves and perhaps neural nets are sensitive to weight change, the training curves are more noisy, and the results can still fluctuate even at high episode counts. Nonetheless, centralized critic does deliver noticeable improvement stated in the hypothesis.

VI. REFERENCES

1. Github: "Google Research Football," https://github.com/google-research/football
2. John Schulman et al. "Proximal Policy Optimization Algorithms", In: arXiv:1707.06347v2, URL: https://arxiv.org/abs/1707.06347
3. Tabish Rashid et al. "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning", In: arXiv:1803.11485v2, URL: https://arxiv.org/abs/1803.11485
4. "Reinforcement Learning and Decision Making", URL: https://github.gatech.edu/rldm/p3_docker (accessed Nov. 28, 2022).
5. Tangri, R. "Generalized Advantage Estimate: Maths and Code," https://towardsdatascience.com/generalized-advantage-estimate-maths-and-code-b5d5bd3ce737
6. https://arxiv.org/abs/1506.02438 (accessed Nov. 28, 2022).
7. Ray GitHub, URL: https://github.com/ray-project/ray/tree/releases/1.6.0 (accessed Nov. 28, 2022)
8. Lim. H., et al. "Federated Reinforcement Learning for Training Control Policies on Multiple IoT Devices" URL: https://www.researchgate.net/publication/339651408_Federated_Reinforcement_Learning_for_Training_Control_Policies_on_Multiple_IoT_Devices (accessed Nov. 28, 2022).
9. Gym Documentation on gym.Wrapper, URL: https://www.gymlibrary.dev/api/core/#gym-wrapper