A Comparison of ML Approaches to Playing in the Google Research Football Environment

Juncheng Tang
Yale University
New Haven, US
juncheng.tang@yale.edu

Greg Schwartz
Yale University
New Haven, US
greg.schwartz@yale.edu

Figure 1: Google Football Project Visual Display

ABSTRACT

In this paper we explore various approaches to playing in the Google Research Football Environment, using performance in Google's Kaggle competition as our primary evaluation. We evaluate reinforcement learning, rule-based, and imitation learning strategies; ultimately using a hybrid approach that implements rules to supplement imitation learning on collected expert data from the Kaggle competition. We use the random forest and neural network classification models to classify common behaviors such as moving directions and implement rule-based algorithm to handle behaviors in special circumstances such as sliding or shooting. Using this hybrid approach, we achieved a Kaggle score of over 1100 and a top 20 percent leader board position in the Kaggle competition. We show how our approach avoids well established weaknesses of imitation learning and discuss how these methods can be generalized to other problems.

KEYWORDS

game artificial intelligence, football game, reinforcement learning, imitation learning, random forest, neural network

1 INTRODUCTION

International football is the most popular sport on the planet. The football video game industry alone is worth billions of dollars worldwide. Being a good player in football requires a natural balance between ball control, learned concepts such as passing, and high level strategy to handle different states. To explore the AI agent's ability to play in complex settings such as a football video game, Google researchers have created an open source RL environment [5]. Through the Manchester City F.C. and Google Research sponsored AI football Kaggle competition, we aim to further explore this space.

We use the standard 11 vs 11 competition mode of the game environment, where the AI controls only one player at one time. The match length is 3000 frames. Our goal for this project is to evaluate

and explore several promising approaches, eventually devising one very good machine learning agent that can defeat most opponents' agents with limited computational resources and time. We will analyze our model's performance in the Kaggle leader board, which contains over 1100 teams and 10000 submissions to the competition. The Kaggle score with which teams are ranked is similar to the elo rating system in chess: teams play similarly ranked teams, gain points for winning, and lose points for losing. A high ranked team beating a low ranked team results in a small score change, whereas an upset results in a large change. The median score is 600 and the highest scores on the Kaggle leader board are around 1600.

In our effort to create a high performing model, we reproduce several open source rule-based, reinforcement learning (RL), and imitation learning (IL) implementations. While there has been some previous research success found in reinforcement learning approaches [2], we conclude that it is not possible to produce a good pure reinforcement learning model with our limited computational power and time. On the other hand, both the rule-based model and the imitation learning model alone are able to achieve fairly reasonable scores at around 1000 in the Kaggle competition. However, the poor generalizability of rule-based models, and the imbalance data problem from the imitation learning expert data, makes them hard to have further improvements. So finally, we choose a hybrid model of both rule-based model and the imitation learning achieve further improvements on AI agent. For reproducibility, the code for this final model has been made available in open-source*.

2 RELATED WORK

As noted in the introduction, there had been several effective open source models already implemented for this google football research environment. The Google Football Research team initiate three reinforcement learning agents using PPO, IMPALA, and Ape-X DQN as the learning model [5]. The authors found that fairly good results were achieved with the provided medium benchmark difficulty AI being beaten by DQN and IMPALA after 500 million frames of training.

Another google research team improves the IMPALA/V-trace learning models with the new Google RL Seed agent [2]. The SEED adopts two state of the art distributed algorithms, the IMPALA/V-trace for policy gradients [3] and R2D2 [7] for Q-learning. This algorithm is a great improvement, requiring substantially less cost and computational power while achieving a faster training speed.

In the Kaggle competition open-source notebooks, there are several high score approaches to the game beyond using reinforcement learning models. Some of these approaches use rule-based models, with varying degrees of success. One of the highest performing models public at the time of the competition was rule-based and scored over 900 points [9]. This relatively high performance was largely due to a complicated set of passing and shooting rules that considered the positions opposing players as well as teammates.

Another Kaggle notebook of interest uses imitation learning with expert data on offensive states and a random forest classifier [6] to classify different actions for given offense states [8]. This model achieves a score of over 1000 in the Kaggle competition.

After the end of the competition, additional agents were shared. One such highly specific rule-based model is noteworthy for reaching a score over 1200, placing it in the top 2% of the competition [1]. One noteworthy aspect of this model was its emphasis on object-detection, checking to see whether opposing players were in the controlled player's movement direction or potential kicking paths.

Another agent released at the end of the competition uses an imitation learning model to achieve a score of over 1400 in the Kaggle competition [13]. This model uses the CatBoost classifier[10] from one top agent's competition data while tuning the input data variables. In addition to its choice of classifier, this model preprocesses the data to add additional information, such as the distance to goal and distance to the nearest teammate. Such changes likely helped to prevent certain false positive classifications, like shooting or passing from too far of a distance.

While there is little available work on applying neural networks to the Google Football Research Environment, such strategies have found success on similar problems. In constructing our neural network, we take particular interest in the use of Dropout [11] as a way to avoid overfitting. We also explore the use findings regarding the coordinate transformations in neural networks to improve performance by reducing the abstraction required of our model [4].

3 METHOD

Observation States and Actions

The Google Football Research Project provides a number of observation states and action states for the agent. The observation states include time, scores, all twenty two player positions, ball positions, player directions, active players, tiredness factor, yellow/red card conditions etc. The actions states are chosen from the given nineteen actions in Table 1, including the directions of moving and actions of handling the ball. Some states such as passing and shooting can only happen when the player is actively controlling the ball while actions such as sliding is only available to perform when the active player does not control the ball. The Google Football Research Project also provides us with the reward function for the reinforcement learning model, which for every score goal counts for +1 and for every conceded goal counts for -1.

Table 1: Action set for agents

Top	Bottom	Left	Right
Top-Left	Top-Right	Bottom-Left	Bottom-Right
Short Pass	High Pass	Long Pass	Shot
Do-Nothing	Sliding	Dribble	Stop-Dribble
Sprint	Stop-Moving	Stop-Sprint	−−−

Rule-Based Model

We start with example agents from Kaggle notebooks and research papers. The first agent created was a pure rule-based bot. It was heavily based on Google's example notebook [12]. This agent only performs basic actions such as moving the ball towards the opponent's goal. There is no consideration of whether to avoid opponents or to chase toward the ball position if possession is lost.

*https://github.com/gregSchwartz18/googleFoootball

Reinforcement Learning Model

The next series of agents were reinforcement learning agents based on Google's open source code for RL SEED and DQN [2][7]. While Google trained their DQN and RL SEED models for 500 million frames with distributed GPU, due to computational constraints we are only able to train our SEED agent on one four-core GPU machine. We faced memory errors when trying to add more actors to distribute computation resource in order to improve computational speed. Due to these issues, we have only trained two agents with reinforcement learning methods. Our Deep-Q Learning model's training is done without distributing computing power, with 1 million training frames, and with 4 actors for about 6 hours. We have also trained our RL Seed model for 100,000 frames.

Imitation Learning Model with Limited Expert Data

Following the issues with computational constraints we explore the possibility of an imitation learning approach. We start with an imitation learning example from one Kaggle notebook, which uses the expert data and the random forest model[6] from SK-Learn library. Its expert data set contains over 80000 frames of active ball possession state data, with most of the data included being the direction controls [8].

Besides the random forest model to classify the actions from the observations, we have also trained a neural network model and several other models from the SK-Learn library. We found that the random forest model was the most effective, and used it when predicting offensive actions in all future models.

In seeking to improve this model, we look to create a more complete set of expert data. As the original data only includes active ball possession states, with the agent required to handle the states without ball control only by rule-based ways. Given the importance of defensive play, this we determined to be a substantial shortcoming. This original data set also does not include many behaviors such as passing and shooting. These behaviors are critical for the ball possession states, increasing the motivation to create a more complete expert data sets.

Imitation Learning Model with Full Expert Data

Data Collection. The Kaggle competition provides full competition match data for all submitted agents online. Accordingly, we create a script to fetch and parse the publicly available json recordings of all games played by the top performing models in the competition. We collected over 100 the top leader board team match data with over 600000 frames of state action pairs from all the movements of expert agents.

We choose to first divide our data set into the offense states and defense states. This was done because the learned behavior for the two data sets were substantially different that learning for one would not inform the other. Due to the complications of the observation states from the data set, we only extract the most relevant input variables from the data set to predict for our actions. The variables are chosen based on some rule-based equations in choosing directions and actions.

For the offense states, we need to consider how to run or pass the ball into safe spaces to avoid our opponents. Also, determining

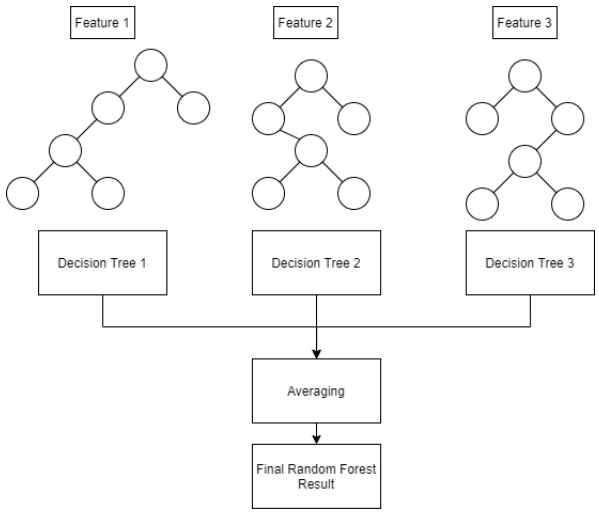

Figure 2: Diagram for Random Forest

a safe space requires a good understanding of the surrounding environment of the active player. So we will consider the active player position, the ball positions, the passing score to teammates, the distances and the heading directions of all other twenty two players towards the active player and the current sticky action of the player.

$$off\_run\_dir = \frac{safe\_space(surrounding) - pos_{player}}{speed(sticky\_action)}$$

$$surrounding = f(dist(oppos), dist(mates), pos_{player}, head_{player})$$

For the defense states, we need to accurately predict the future ball position based on the current ball position and its moving directions. There are also a few variables which will influence the speed and moving direction of the ball such as the opponent player's ball possession states and ball distance toward our current player. So we extract the current player role, the active player position, the ball position, current active ball control team and direction and the current sticky action of the player as our variable for defense states.

$$def\_run\_dir = \frac{pos_{ball} + \delta * dir_{ball} - pos_{player}}{speed(sticky\_action)}$$

$$\delta = f(state, dist(pos_{ball}, pos_{player}), angle(dir_{ball}, pos_{player}))$$

We attempted training the model and agent with the full expert data set on both the offense and defense states. We used both random forest and the neural network to perform the classification. Although we achieve a very high accurate action prediction rate (over 90%) from the data set, the agent plays poorly against our previous model with the limited expert data set.

Hybrid Model with Full Expert Data

Imbalance Data. Although expert data from the Kaggle competition provided us with abundant state action pairs to perform imitation learning, we discovered issues when we use the data to create the

classification model for our agent. One of the main issues is the imbalance data problem. For example, in the defense state, the slide action is a critical action to tackle opponent players attempting to dribbling through our current defense player. However,the sliding actions consists only fewer than 2% of the total actions in our data set. (As shown in Figure 3, Action 16) This will largely be ignored by our training model. On the contrast, the sprint, release sprint, and release dribble actions which are the most common behaviors consist over 30% of the total action data. (As shown in Figure 3, Action 13 and 15 and 17)

The over-represented actions can be easily determined using rule-based methods, as the conditions governing their optimal usage are relatively simple. We incorporate this into our imitation learning approach by using the IL model if these conditions are not satisfied and filtering the data appropriately.

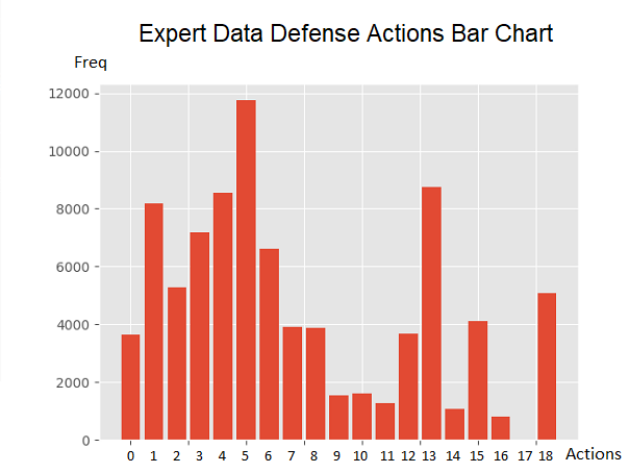

Figure 3: Bar chart of actions in the defense expert data

Filter Data. In order to make our data set more balanced, we further classify our data into multiple parts. For the offense state, we will categorize our actions to be direction action and kick action, which the kick actions only include passing and shooting actions. For the defense states, we categorize our actions to be direction actions and tackling actions. For the sprint and release sprint actions, we filter them out from the data set and handle them independently using rule-based methods. Figure 4 shows the category on how our model filters the data. After the new category applied to our data set, the data becomes more balanced and we can use the data to train for our model.

We first reused the random forest model for the defensive data, although the resulting sav file proved too large for submission. Due to this, we handle defensive movement with the selected critical data using a neural network.

Neural Network Classification. The our first approach with the neural network to classify defensive movement data was based off of the model previously used for the offensive data. This approach originally suffered from over-fitting, even with relatively simple models. However, we were able to prevent this overfitting and even increase our model size using Dropout layers [11].

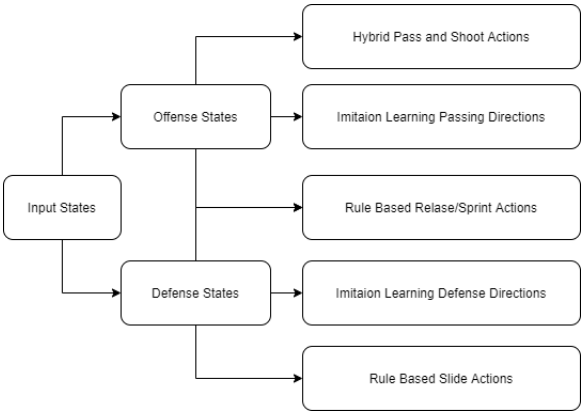

Figure 4: Diagram for Hybrid Model

Including dropout, the architecture uses 12 layers for training. The first layer is the input with a size of 46 and the last is the output with a size of 19. The middle 10 layers alternate between a dense layer of 512 ReLu-activated neurons and a 0.1 dropout layer. After training, a softmax layer is added for making probabilistic predictions. This approach worked approximately as well for the defensive movement data as it did for the offensive movement data, still worse than the random forest classifier.

Neural Network Regression. To improve the neural network's performance, we change the model from performing classification to predicting two values, which combined form a coordinate pair. This is done by preprocessing the data such that the original actions are translated into two indices as shown in Figure 5. These actions are ordered such that the indices form coordinate pairs in the same direction as their corresponding action, assuming that the agent's current position is in the center.

	0	1	2
0	Top Left	Top	Top Right
1	Left	Idle	Right
2	Bottom Left	Bottom	Bottom Right

Figure 5: Directions Grid

This change was made as the relationship between the ball movement, players' movement, and optimal movement direction seems to be understood more easily and with less abstraction with the target is also in terms of coordinates. We note that our neural network

functions by learning the systems of equations that govern the coordinate transformations representing the data [4]. As such, we expected to see better performance if the output was kept in terms of coordinates. Additionally, the translation to coordinates makes the relationship between the actions more clear. If the ideal movement direction was between 'Top Left' and 'Top', the regression model would likely be better able to handle the uncertainty.

This resulted in a similar architecture for most of the layers. We kept with 14 layers for training and an input layer of size 46. The output layer was changed to a size of 2. The over size of the laters was shrunk as well, with the middle 12 layers alternating between a dense layer of 256 ReLu-activated neurons and a 0.1 dropout layer.

This resulted in effective movement when the defenders were near the ball. Advanced behaviors were observed such as running alongside attackers, forcing them into poor shooting angles[†]. When the defenders were far from the ball, unreasonable actions – such as running away – were much more common[‡]. This was likely due to a lack of expert data in states where the ball is far away from the defender.

Hybrid Defensive Movement. We address the poor accuracy rate of the neural network in such states by switching to to a rule-based approach when the ball is a certain distance away. This rule uses the ball's position and direction to determine where the ball will be, and moves the agent in that direction.

The addition of hybridization of our defensive movement resulted in our final model, as displayed in Algorithm 1.

Algorithm 1: Main Agent Model Algorithm

```
Result: Return the action based on the observation
Load Prediction models;
Get the Observation States;
if not sprinting then
    return sprint action;
end
if close to keeper and sprinting then
    return release sprint action;
end
if Offensive state then
    RF predict kind of kick;
    if not a kick then
        RF predict kind of movement;
        return movement;
    else
        return kind of kick;
    end
else
    if tackle condition then
        return tackle;
    else
        if near ball then
            NN predict kind of movement;
            return movement
        else
            return movement towards ball
        end
    end
end
```

4 EXPERIMENTS

When playing football one's performance is heavily dependent on the opponent being played against. As such we examine a variety of metrics. For testing full agents we examine their performance – against each other and against the easy, medium, and hard benchmark agents provided by Google. We also use their score generated by the Kaggle competition, which is determined by the elo-esque system described earlier. For component testing on the imitation learning models, we compare the prediction accuracy rate on testing data. For the classification models we compare cross-entropy loss and for the regression models we compare mean squared error.

As another benchmark, the pure rule-based agent achieved a public score of roughly 550, or just below the top 50%.

Reinforcement Learning Model

Our two reinforcement learning agents proved to be significantly hampered by their shortened training. While Google's pure RL agents were able to beat the medium benchmark AI, the shorter training imposed by our computational restrictions resulted in SEED agents that are not strong enough to defeat the medium or even easy difficulty benchmark AI.

Our Deep-Q approach proved more effective, but was still substantially worse than the rule-based model. For purposes of comparison, we got a score of only around 270 in the Kaggle competition with our pure RL model and 350 with our Q-Learning model, neither of which are good results. And as this under performance is largely due to the computational constraints, it severely limits the potential of taking this approach further.

Table 2: Public Scores of Submissions

Rule-based	SEED RL	Deep-Q
550	270	350

Imitation Learning Models

With Limited Expert Data. Our imitation agents developed in response require significantly less processing power to be effective. The first used a random forest classifier and expert data from a Kaggle open source notebook. The random forest classifier using the data provided predicts the players action from the given expert data at an accuracy rate of 75% on the data reserved for testing. This proved sufficient to generate a good machine learning model. In the Kaggle competition it scored 1000, which was marked a massive performance increase from the previous models.

The neural network trained on the same data resulted in a lower accuracy rate of 65% and substantially worse performances overall, scoring 930 in the Kaggle competition. The other decision trees

[†]For a video of this behavior, visit https://youtu.be/XzfGDxOzoHo
[‡]For a video of this behavior, visit https://youtu.be/TlmDTzu6A0w

tested were all less accurate than the neural network and were not submitted to the competition.

Hybrid with Full Expert Data. The unbalanced data from our json parser – with all variables and actions included – could be predicted by the random forest with 90% of accuracy. This was higher than the accuracy rate on the public offensive data set, but did not lead to good performance in the real games because of the unbalanced data problem. The model struggled against the easy benchmark AI and was not submitted to the competition.

The balanced data with only 6 variables – active player x and y, ball x and y, and ball direction x and y – was only predicted at 50% of accuracy with random forest. However, this was without the large prevalence of certain actions present previously, and the model appeared qualitatively to predict running directions with a reasonable degree of accuracy. In the Kaggle competition, this model scored 1040.

Table 3: Public Scores of Submissions

Random Forest Limited	NN Limited	Full Balanced
1000	930	1040

Hybrid Models

Filtering the data as described above and adding data for sprinting, possession, and dribbling increased the accuracy to 65%; making it the best performing data set and used for training all future models. The improved data and additional rules for filtering, brought the model's score up to 1050.

Improving Defensive Movement. Using the random forest to predict defensive movement resulted in substantial performance improvements, able to typically beat the best of the previous hybrid models. However, the sav files on which the random forests were stored proved too big to submit.

The original neural network classifier was able to predict the expert data defensive movement with an accuracy rate of 65%, compared to the random forest's 75%. This decrease manifested in substantially worse performance and was unable to beat the best previous hybrid model.

The switch to using regression and a coordinate target for the neural network resulted in a mean squared error of roughly 0.15 on the validation data. After rounding the predicted coordinates to their nearest action, this meant 75% accuracy on the testing data. This improved defensive movement was submitted to the Kaggle competition and scored 1070.

The rules for defensive movement when far from the ball further improved performance. The described unreasonable behavior – such as running away from the ball – was no longer observed, and the model achieved our highest Kaggle score of 1100.

5 CONCLUSION AND FUTURE WORK

The google football research project is an interesting as well as challenging machine learning research project for our team to work on. The project includes challenges such as complicated environmental variables, large state observation spaces, and unbalanced action data sets. With over 1100 teams competing in this Kaggle competition, we tackle these challenges while competing against a diverse array of strategies. Our goal for this project is to devise one very good machine learning agent which can defeat most opponents' agents with limited computational resources. Towards this end, our hybrid model successfully achieves over 1100 points in the Kaggle competition with a top 20% position.

Table 4: Public Scores of Submissions

Hybrid v1	Hybrid NN Defense v1	Hybrid NN Defense v2
1050	1070	1110

During the period of developing our agents, we explore many elements of machine learning, including how to handle the imitation learning and how to choose the classifiers such as decision trees, random forest and neural networks. One crucial takeaway is the highly state-dependent performance of the different mechanisms tested, as highlighted by the success of our final hybrid submission. In many offensive states the random forest classifier outperformed all other approaches tested. In the more geometric defensive movement states, the neural network regression performed best. While the rule-based approach can be constrictive and scale poorly, on states under or over represented in the expert data its inclusion proved highly beneficial.

More specifically, rules were successfully used to supplement our implementation learning models in states that the experts visit infrequently, a well established problem with imitation learning. While our on/off binary usage of rules may not generalize to most imitation learning applications, using relatively unrestrictive rules as a bound on the imitator's behavior or to help set the imitator's confidence may help avoid clearly unreasonable behavior.

While the current Kaggle competition has ended, future work remains on our model as well. The sliding actions as well as the shooting actions are still not performing ideally in the competition based on our observations. Also the rule-based defensive movement used over long distances could be replaced with an imitation learning model training on mid-tier competitor data exclusively in those states. We would do this in the hope that the desired behavior is both simple enough to be similarly handed by high and mid tier models and that the combined data set will be large enough be enough for the model to properly converge. The rules themselves can also be significantly augmented, as demonstrated by high performing notebooks with much more nuanced rules [1].

Further, it could be worthwhile to revisit the model for predicting offensive movement. Given the improvement seen when using the coordinate-based neural network on the defensive movement, taking similar approach to offensive movement seems promising.

REFERENCES

[1] Ahmet Erdem. [n.d.]. Object-oriented rule-based agent for Google Research Football. https://github.com/aerdem4/google-football

[2] Lasse Espeholt, Raphaël Marinier, Piotr Michal Stanczyk, Ke Wang, and Marcin Michalski. 2020. SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference. In *International Conference on Learning Representations*. https://arxiv.org/abs/1910.06591

[3] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures *(Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, Stockholmsmässan, Stockholm Sweden, 1407–1416. http://proceedings.mlr.press/v80/espeholt18a.html

[4] Michael Hauser and Asok Ray. 2017. Principles of Riemannian Geometry in Neural Networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 2804–2813.

[5] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. 2020. Google Research Football: A Novel Reinforcement Learning Environment. arXiv:1907.11180 [cs.LG]

[6] Andy Liaw and Matthew Wiener. 2002. Classification and Regression by randomForest. *R News* 2, 3 (2002), 18–22. https://CRAN.R-project.org/doc/Rnews/

[7] Jieliang Luo and Hui Li. 2020. Recurrent Distributed Reinforcement Learning for Partially Observable Robotic Assembly. arXiv:2010.08052 [cs.RO]

[8] Ken Miller. 2020. 1149 ish bot RL approximation. *Kaggle* (2020). https://www.kaggle.com/mlconsult/1149-ish-bot-rl-approximation

[9] Ken Miller. 2020. best open rules bot score 1020.7. *Kaggle* (2020). https://www.kaggle.com/mlconsult/best-open-rules-bot-score-1020-7

[10] Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. CatBoost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 6639–6649. http://papers.nips.cc/paper/7898-catboost-unbiased-boosting-with-categorical-features

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958. http://jmlr.org/papers/v15/srivastava14a.html

[12] Piotr Stanczyk. 2020. GFootball Template Bot. *Kaggle* (2020). https://www.kaggle.com/piotrstanczyk/gfootball-template-bot

[13] S Toppo. 2020. S Toppo solution LB 1400. *Kaggle* (2020). https://www.kaggle.com/iiyamaiiyama/s-toppo-solution-lb-1400

A APPENDIX

A.1 Moving Kaggle Scores

The nature of the Kaggle elo-esque system for scoring means that the scores are not static. In our results we use the scores given to our models at the original end of the competition December 8, 2020; rounded to the tens place. However these scores are subject to change, especially the newer models that have played fewer matches. Additionally Google Research choose to extend the time spent playing the final games, reflected an expectation that some scores are still not as accurate as possible.In response, we include the past Kaggle scores of all of our models, taken at the end of each day in the week leading up to the original end of the competition.

The pure Rules-based, SEED, and Deep-Q models stay constant throughout the week. This is likely due to having already competed in nearly a month of matches previously.

The newer models show more variation. When evaluating the validity of the Kaggle scores, we are primarily concerned with relative performance between the models. This is in part because this paper aims to evaluate the effectiveness of different approaches relatively to each other and part because the nature of the scoring system also makes analyzing the magnitude of score differences challenging. It is unclear both the quantity and quality of Kaggle opponents within any given score gap between agents. With this comparative approach, the score history supports our analysis on the Neural Network Limited and Hybrid Neural Network Defense v2 models. Both are consistently ranked as the best and worse of the newer models, respectively.

The relative performance of the Full Balance and other hybrid models is less clear. The placement and trends in the later days are more meaningful than those in the earlier days, as the latter days consider more matches and are likely to reflect a more accurate placement of opposing models. Even so, these models' placements appear to still be in flux. This makes certain comparisons not discussed in our paper – such as whether the improved close-up defense in Hybrid Neural Network Defense v2 had more of an impact than it's worsened movement at a distance – unclear until more data can be collected.

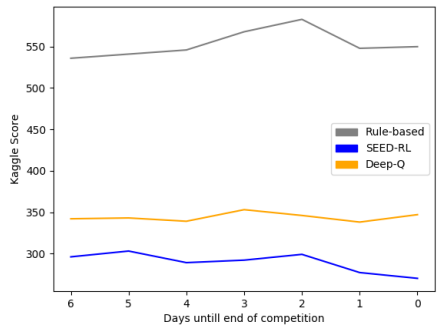

Figure 6: Kaggle score history for older models

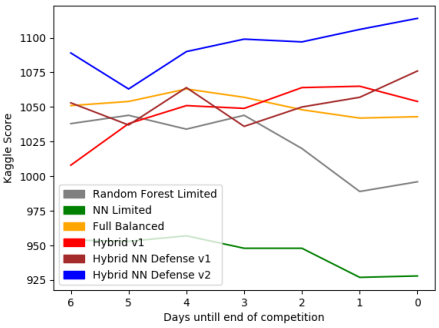

Figure 7: Kaggle score history for newer models