Predicting Match Outcome in League of Legends Based on Team Composition

Ratu, Valdi
valdi00@student.ubc.ca

Chen, Bob
bbyung@student.ubc.ca

Project Github

Front-end Gihub

Abstract

League of Legends (LoL) is one of the largest e-sports titles in the world. With each of the 2 teams having 5 players, there is a wide array of possible team compositions in the game. We used a number of models to predict the outcome of LoL matches only based off of the champions and roles of the champions. We were able to obtain a performance of 53% using an embedding neural network on the evaluation set, but an accuracy of up to 56% on the test set using logstic regression. Though the evaluation performance was modest, the test performance signifies the potential of a better performance. This is a crucial first step to develop models that helps with drafting, one of the most important aspects of League of Legends.

1 Introduction

In Riot Games' popular MOBA title, League of Legends (LoL), 2 teams of 5 players battle with only 1 victorious team in the end. Every player in the match controls a "champion", a character in the game with unique abilities. There are 5 roles in each team which each player fills, and certain champions are better versed in certain roles.

Many factors determine the winner, especially in-game metrics such as team gold count, objectives destroyed, and the number of kills a team has. However, the champions that the players pilot could be considered the biggest factor in determining the outcome of the match. In each LoL patch, there are consensus strong champions, but synergies between champions as well as counters to enemy champions must also be considered. Playing a champion in different roles also changes the dynamic of the champion. Thus, the phase of "champion-select", where players choose their champions is often touted as one of the most important yet controversial parts of the game.

The goal of this project is to create a machine learning model that predicts the probability of a given team composition in LoL to win against another team composition.

Such a tool could be useful given League of Legends' large competitive scene, often cited as being the biggest e-sport in the world today. The prize pool for the 2022 World Championship tournament was $\$2,225,000$ USD[1] and the peak concurrent viewers that the tournament drew was 5.15 million people[2].

The model may be used to enhance the spectating experience as it could provide insight into which team is heading into the game with a better draft. It can also analyze the strength and weaknesses of a team composition, allowing teams to create better drafts. Nevertheless, it may help find new trends and team compositions that are not currently being used, granting teams an advantage over opponents.

[1] https://liquipedia.net/leagueoflegends/World_Championship/2022#Prize_Pool
[2] https://escharts.com/tournaments/lol/2022-world-championship

2 Related work

Prior works have attempted to predict League of Legends match outcomes. For example Lin (2016) used gradient boosted trees (55$\%$ accuracy), Kim and Lee (2020) used a bidirectional LSTM embedding model (58.07$\%$), Chen et al. (2017) used logistic regression (60.24$\%$). and Do et al. (2021) used a deep neural network (75.1$\%$). These are accurate predictions, however they utilized player-specific features such as a player's champion mastery and experience, player role mastery, and player match history.

The models some prior works created may also not generalize to the entirety of all League of Legends matches. For example, Lin (2016) only used matches for a specific gold-ranked league and only used 1000 matches to train their model. The temporal information of the matches was also not specified, meaning that the dataset could contain matches between different versions of the game. None of the aforementioned papers discussed the game version when describing their data.

There are also other studies that obtained impressive accuracy such as Ani et al. (2019) where they used ensemble methods to obtain a prediction rate of above 90$\%$. However, in addition to pre-match features as mentioned before, the model also accounts for in-match features such as the number of kills, experience, and gold gain. The data used was also only 1,500 matches, and the matches were professional e-sports matches and the temporal information was also not explicitly stated.

3 Data

The goal is to create a model to predict the outcome of a match only from the team compositions of both teams. The model is desired to be player agnostic. Thus, only the highest tiers of players from multiple different regions were looked at as skill and champion experience (whether playing against or playing as a champion) is a lesser factor compared to matches of lower skill levels. As champion strengths also depend on the version of the game, only matches within a specific date range are captured.

A website[3] that tracks player ranks was used to scrape the names of the top 500 players from each of the regions North America, Europe West, and Europe-Nordic-&-East, giving us 1500 players in total to look for. Using Riot Games' API[4], up to 100 games per player were fetched between the dates of March 8, 2023 and March 22, 2023 which corresponds to patch version 13.5 of the game.

In total 26,569 unique matches that matched the criteria were found. The matches were divided up, reserving 3,000 matches for evaluation and 3,000 matches for testing and optimization. The remainder 20,569 matches were reserved for training.

4 Models

4.1 Logistic Regression

As this is a binary classification problem, logistic regression was utilized as a simple baseline model. The implementation was done through PyTorch as an equivalent 1-layer neural network using a sigmoid activation function. Stochastic gradient descent (SGD) was used to train with a batch-size of 64, a learning rate and weight decay of $lr = 0.00075, wd = 0.0015$. Binary cross entropy (BCE) loss was used as the criterion.

For the input, one-hot encoding to encode champion-role pairs as well as the team the pairing was in was utilized. As there are 163 champions in patch 13.5, a total of

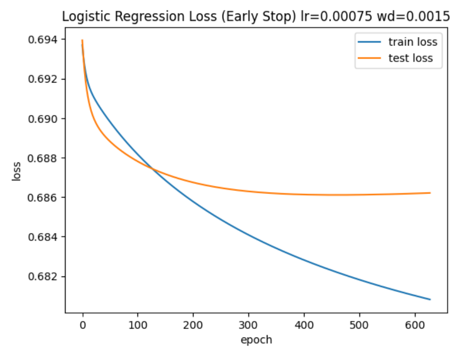
$$163 \text{ (champions)} \cdot 5 \text{ (roles)} \cdot 2 \text{ (teams)} = 1{,}630$$

features were used as input.

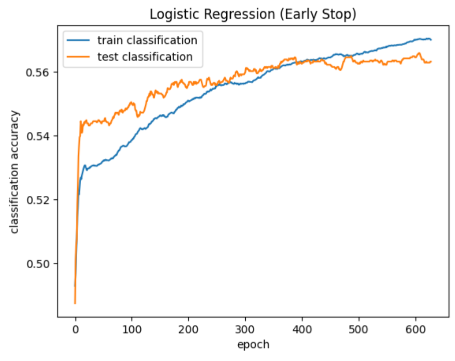
Figure 9 in the appendix demonstrates how the logistic regression model quickly over-fit to the data, so we employed early stopping to obtain the best test-accuracy and test-loss as shown by Figure 1. Table 1 shows the results, with the early stop having a test accuracy of 56.31$\%$, an almost 3$\%$ increase from the no-early stop model.

[3] `https://u.gg/leaderboards/ranking?region=na1`

[4] `https://developer.riotgames.com/apis`



(a) Loss



(b) Classification accuracy

Figure 1: (a) Loss and (b) classification accuracy of logistic regression model with early stopping.

Model	Train loss	Test loss	Train accuracy	Test accuracy
log reg	0.6751	0.6883	0.5795	0.5365
log reg (early stop)	0.6808	0.6862	0.5701	0.5631

Table 1: Results for logistic regression model

4.2 Random Forest

Another baseline model ran is the ensemble method random forest. SKLearn's implementation of RF was used with the following settings: *criterion* = *gini, bootstrap* = *False, max_depth* = *5*. The same one-hot encoding used in the logistic regression model was also used here. Figure 2 and Table 2 shows the results with 200 trees giving us the best test accuracy of 53.83%

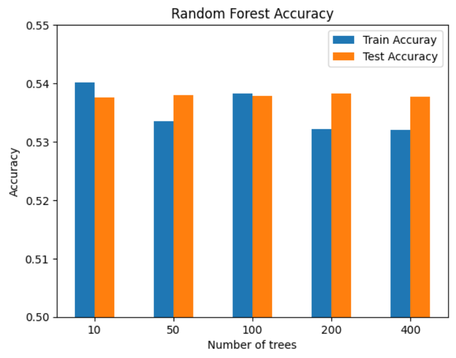


Figure 2: Bar graph of random forest results (avg. 3 runs)

Num. trees	Train accuracy	Test accuracy
10	0.5402	0.5376
50	0.5335	0.5380
100	0.5328	0.5378
200	0.5322	0.5383
400	0.5320	0.5377

Table 2: Random forest results (avg. 3 runs)

4.3 Naïve Bayes

The third and final baseline model is Naïve Bayes. SKLearn's implementation of Naïve Bayes was used. The same one-hot encoding feature set mentioned previously was also used here. Figure 3 and Table 3 shows the results with an α (Laplace smoothing parameter) of 30, resulting in the best performance of 55.58% test accuracy.

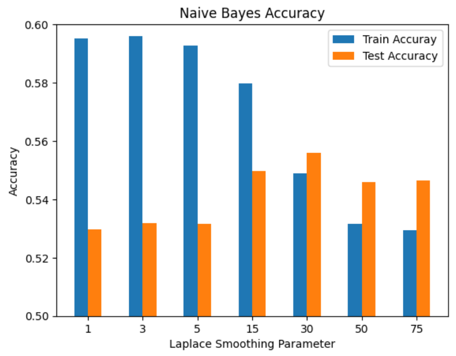


Figure 3: Bar Graph of Naïve Bayes Results

α	Train accuracy	Test accuracy
1	0.5952	0.5298
3	0.5960	0.5318
5	0.5928	0.5315
15	0.5797	0.5498
30	0.5490	0.5558
50	0.5317	0.5458
75	0.5294	0.5465

Table 3: Naïve Bayes Results

4.4 Neural Network Embeddings

To capture more information and interactions between champions (both synergies and counters) and roles, adding interaction terms to the one-hot encoding feature set would result in an excess amount of features to be appropriate. Therefore, embedding layers to process the champion and roles were used. Figure 7 in the appendix illustrates the architecture.

After testing, a champion embedding size of 50 with a role embedding size of 3 gives a feature space of:

$$10(\text{players}) \cdot [50 + 3] = 530.$$

Having 2 hidden layers with sizes [100, 10] performed the best. The model was implemented in PyTorch. Training was accomplished using SGD with a batch size of 64, a learning rate and weight decay of $lr = 0.001, wd = 0.0019$, and using BCE loss. As seen in Figures 10 and 11 in the appendix, using a ReLU or Tanh activation function would result in over-fitting. Therefore, the sigma activation function as well as early stopping were used to ensure no over-fitting. Table 4 shows the result with the sigmoid activation function, producing the best result with a 55.58% test accuracy, an almost 4% improvement compared to the other activation functions.

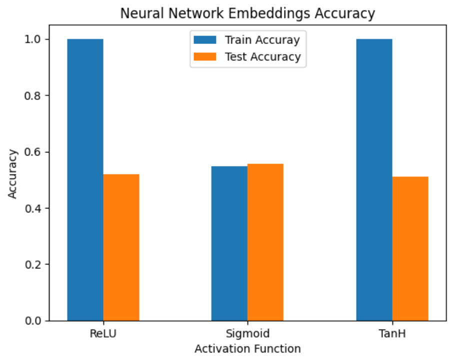


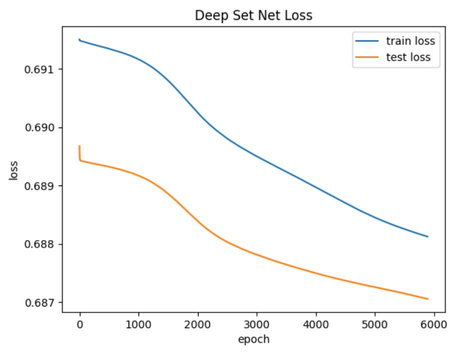
Figure 4: Bar graph of results for embedding neural network).

Activation Function	Train loss	Test loss	Train accuracy	Test accuracy
Sigmoid	0.6859	0.6866	0.5485	0.5558
Tanh	0.0758	1.5325	0.9995	0.5102
ReLU	0.0601	1.3211	1.0000	0.5198

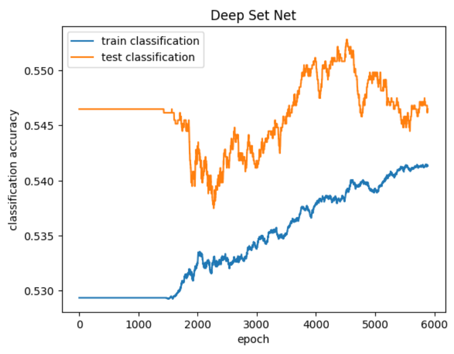
Table 4: Results for embedding neural network

4.5 Deep Set Network

For the final model, the deep set architecture outlined by Zaheer et al. (2018) was used. The architecture is appropriate as within each team, the model should be invariant to the order the champion and roles are introduced. Not only that, but with the combining of set elements after processing them through the Φ network, it should capture champion synergies within a team. Each team is considered a set and within each set, there are champion-role tuples as the elements within the set. Figure 8 in the appendix showcases the architecture we used.



(a) Loss



(b) Classification accuracy

Figure 5: (a) Loss and (b) classification accuracy of deep set neural network with early stopping.

After rounds of testing, we found that a champion embedding size of 32, role embedding size of 3, a Φ network output size of 10, and sigmoid activation granted the best results. The model was implemented in PyTorcch and was trained using a learning rate and weight decay of: $lr = 0.001, wd = 0.0003$, and BCE loss. Early stopping was used to prevent overfitting. Due to this model being very sensitive to initialization, the best resulting model was taken in terms of test loss. The result can be found in Table 5 with a test accuracy of 54.65%. The volatility of the model can be seen in Figures 13 to 15 in the appendix.

	Train loss	Test loss	Train accuracy	Test accuracy
Deep Set Network	0.6881	0.6871	0.5413	0.5465

Table 5: Results for Deep Set Network

5 Results and Evaluation

With all of our models fitted and optimized, we then test them against the 3,000 matches we had reserved for evaluation purposes. The results can be found in Figure 6 and Table 6

We had also set up a front-end (Figure 16 in the appendix) that would allow us to do a human benchmark test to allow us to see how well our models perform compared to humans. Unfortunately due to time constraints and availability, the benchmark was only done by 4 players, guessing 50 matches each. All matches were part of the evaluation set and each person who did the experiment have played LoL for 6+ years.

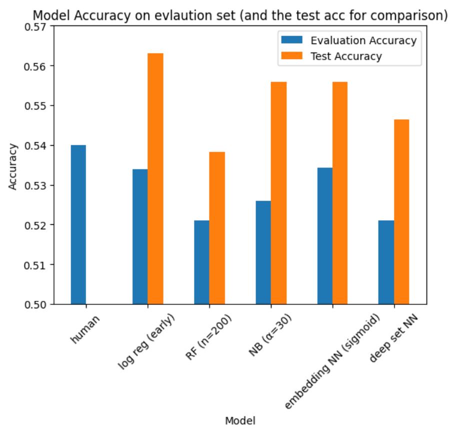


Figure 6: Bar graph of model performance on evaluation set.

Model	Num correct	Num total	Evaluation accuracy	Test accuracy
Human benchmark (avg.)	27	50	0.54	N/A
log reg (early stop)	1,602	3,000	0.5340	0.5631
RF (n=200)	1,563	3,000	0.5210	0.5383
NB (α=30)	1,578	3,000	0.5260	0.5558
Embedding NN (sigmoid)	1,603	3,000	0.5343	0.5558
Deep set NN	1,563	3,000	0.5210	0.5465

Table 6: Model performance on evaluation matches

6 Discussion

From the evaluation results, surprisingly the simple baseline model of logistic regression with early stopping performed very well (53.40%), only being narrowly beaten by the larger and more complex embedding neural network (53.43%). The test error of the logistic regression model was also surprising, considering it is only $\sim 4\%$ lower than Chen et al. (2017)'s logistic regression model despite using far less information. The other baseline models of random forests and Naïve Bayes also performed decent, only $\sim 1\%$ worse than the best model.

Unfortunately, the deep set neural network performed much worse than we hoped for, tying for last place in terms of evaluation accuracy. Other than the arduous training, we do not see a reason for this model to perform as poorly as it did. We still believe a deep set neural network could work very well for this situation, so more exploration and optimization into this architecture is required for future works.

Comparing the results to human performance, none of the models were able to match the human benchmark of 54%. However this should be taken with a grain of salt due to the low sample size and low number of matches looked at for the human experiment. A larger sample size is required to make a more definitive conclusion on how these models compare to a human's ability to predict match outcomes based on champion and roles.

The models we used were not as strong of a predictor of match outcomes as we had hoped, however this result is not at all surprising. League is a very complex game and champions are always constantly balanced in an attempt to have win-rates settle around 50%. As such a player-agnostic model will likely never obatin the same match prediction accuracy of other models which utilize other pre-match information. However, we still believe that it is possible to increase the performance to be around $55 - 58\%$, though that will require more work possibly with more data, better optimization, or entirely different architectures.

References

Ani, R., Vishnu Harikumar, Arjun K. Devan, and O.S. Deepa (2019). "Victory prediction in League of Legends using Feature Selection and Ensemble methods." *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pp. 74–77.

Chen, Zhengxing, Yizhou Sun, Magy Seif El-nasr, and Truong-Huy D. Nguyen (2017). *Player Skill Decomposition in Multiplayer Online Battle Arenas*. arXiv: `1702.06253 [cs.SI]`.

Do, Tiffany D., Seong Ioi Wang, Dylan S. Yu, Matthew G. McMillian, and Ryan P. McMahan (2021). "Using Machine Learning to Predict Game Outcomes Based on Player-Champion Experience in League of Legends." *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*. ACM. DOI: `10.1145/3472538.3472579`.

Kim, Cheolgi and Soowon Lee (2020). "Predicting Win-Loss of League of Legends using Bidirectional LSTM Embedding." *KIPS Transactions on Software and Data Engineering*. Vol. 9. 2, pp. 61–68.

Lin, Lucas (2016). "League of Legends Match Outcome Prediction."

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola (2018). *Deep Sets*. arXiv: `1703.06114 [cs.LG]`.

A Appendix

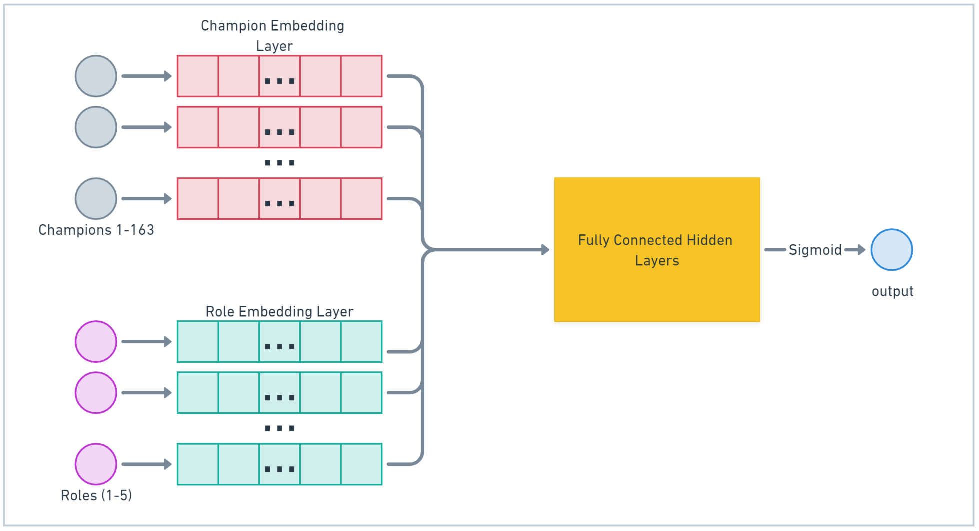


Figure 7: Neural Network with Embeddings Architecture.

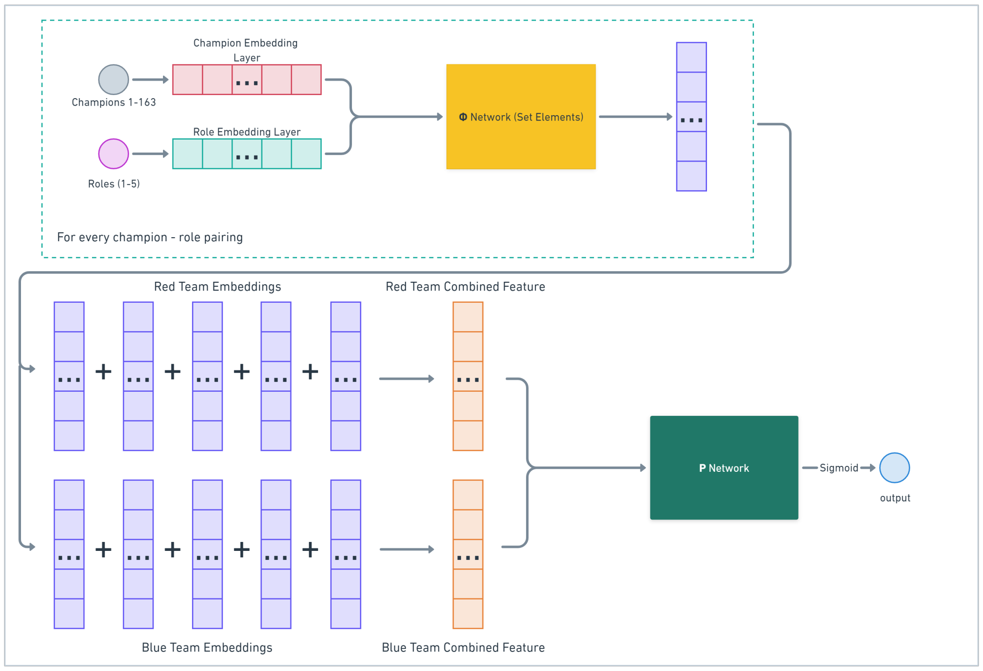
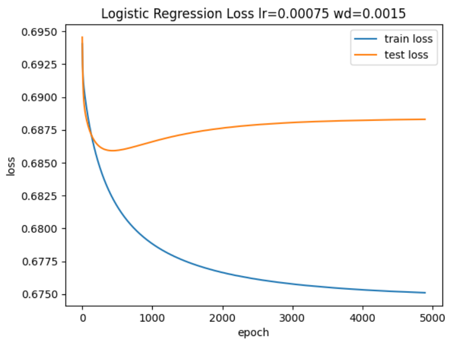
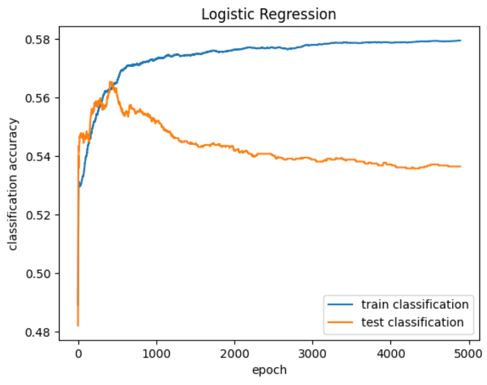


Figure 8: Deep Set Network Architecture.

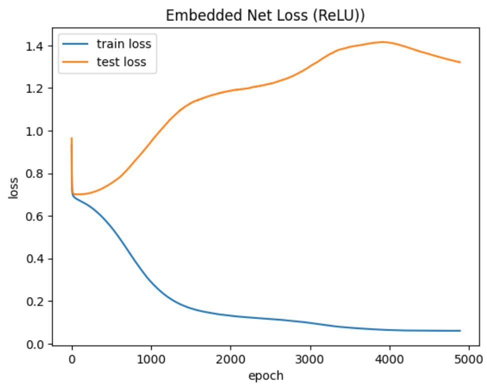


(a) Loss

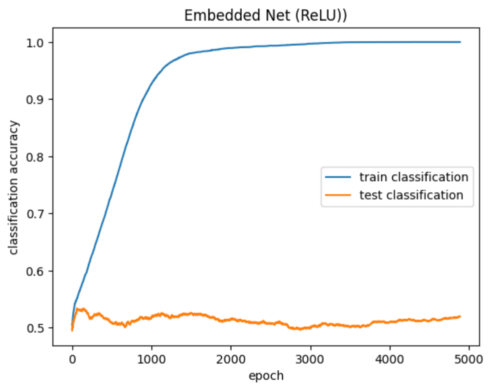


(b) Classification accuracy

Figure 9: (a) Loss and (b) classification accuracy of logistic regression model up to 4890 epochs

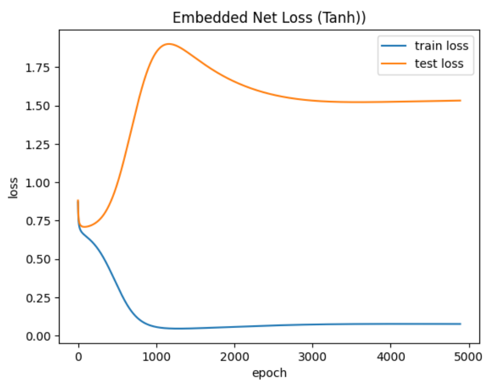


(a) Loss

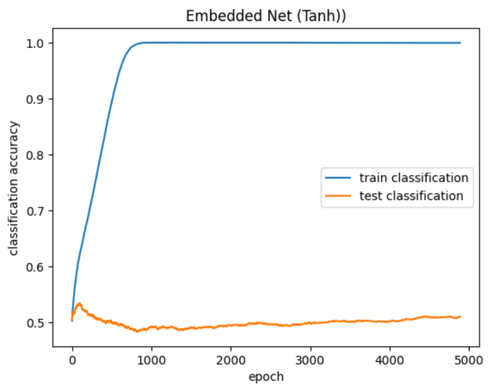


(b) Classification accuracy

Figure 10: (a) Loss and (b) classification accuracy of neural network with embeddings (ReLU activation)

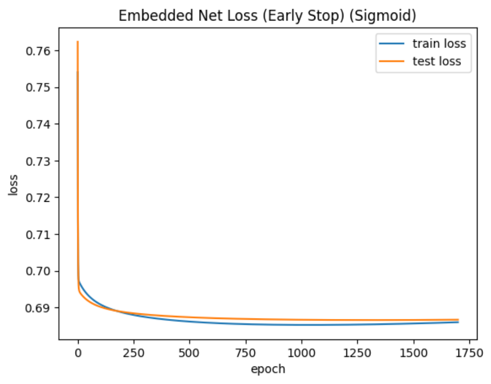


(a) Loss

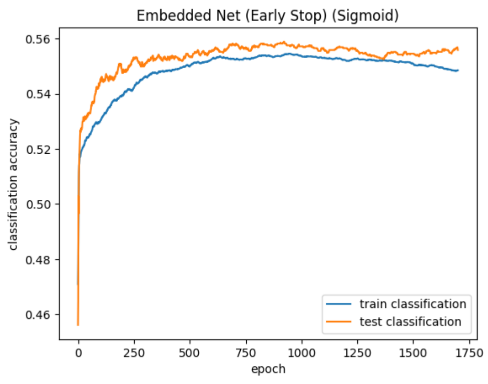


(b) Classification accuracy

Figure 11: (a) Loss and (b) classification accuracy of neural network with embeddings (TanH activation)

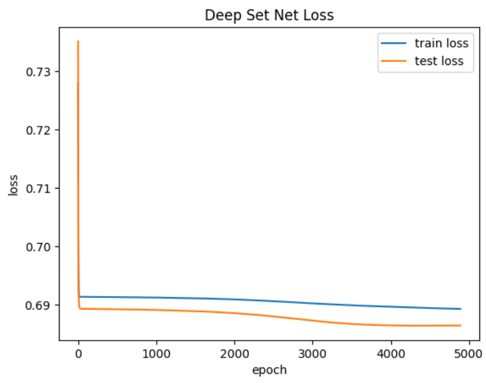


(a) Loss

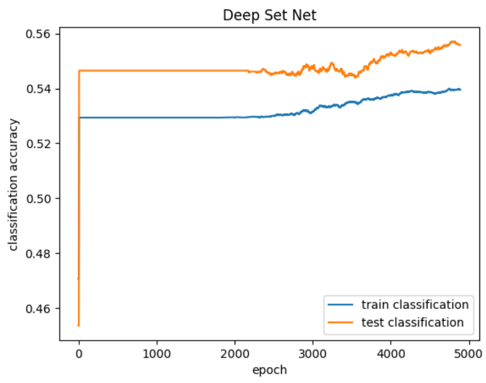


(b) Classification accuracy

Figure 12: (a) Loss and (b) classification accuracy of neural network with embeddings (Sigmoid activation)

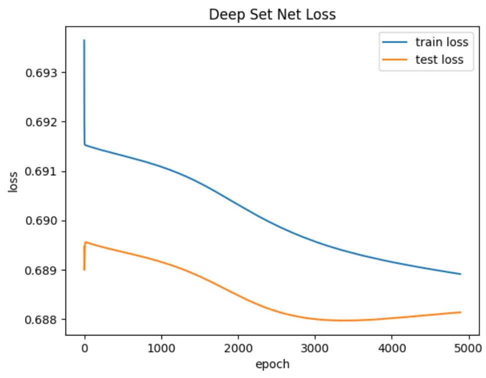


(a) loss

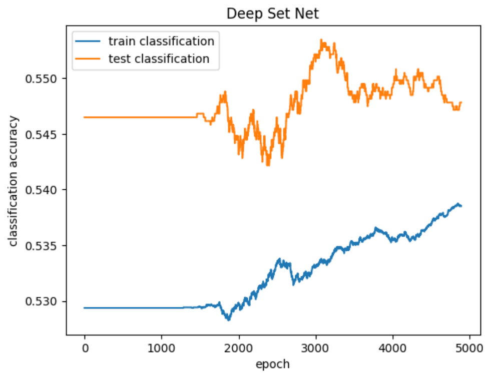


(b) classification accuracy

Figure 13: loss (a) and classification accuracy (b) of deep set neural network

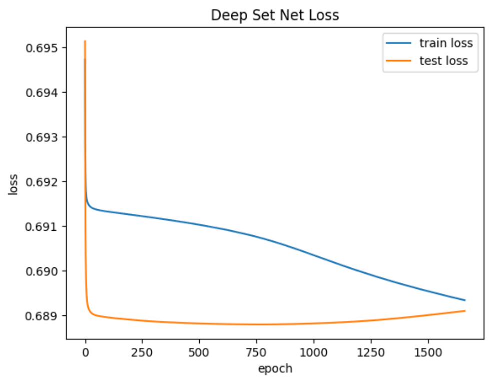


(a) loss

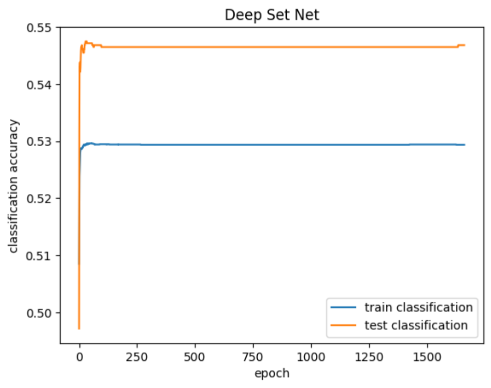


(b) classification accuracy

Figure 14: loss (a) and classification accuracy (b) of deep set neural network



(a) loss



(b) classification accuracy

Figure 15: loss (a) and classification accuracy (b) of deep set neural network

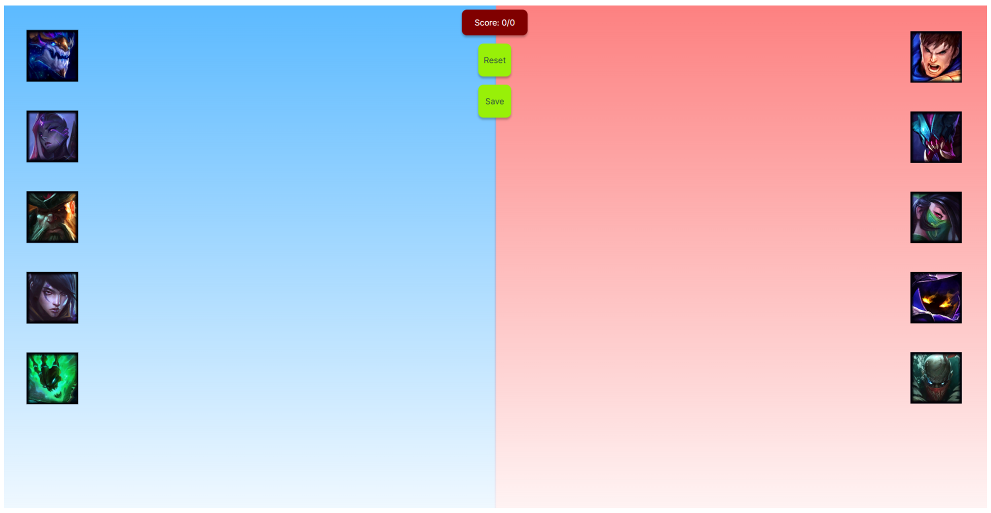


Figure 16: Frontend for human benchmark