League of Legends Game Result Predictor Based on Champions Selected

Yage Jin

San Diego State University

San Diego, California, United States

Email: yjin7099@sdsu.edu

Abstract—**In this study, I experimented creating multiple neural network models with different parameters trying to predict the match result of ranked games in League of Legends, given selected champions and their statistics. The data used were extracted from Riot Games API including champions selected with their corresponding positions, gold and experience points earned for each player at 15 minutes, and the outcome of the match. These were converted into feature vectors and statistics as pre-game knowledge, and the feature vectors were used to train and evaluate the neural network. I find that with limited data, it is extremely hard to accurately predict the outcome of a match, and I was able to improve the accuracy of the model slightly by tweaking the parameters of the models.**

1 Introduction

League of Legends is a video game created by Riot Games in 2009, and it has become one of the largest esports to this date in the multiplayer online battle arena (MOBA) genre. The game mode, Summoner's Rift, requires 10 players to play, 5 on each team with the goal of destroy opposing team's Nexus. Before a ranked game starts, every player is given a role for the game, and is required to choose a unique champion from a pool with over a hundred different champions, each with their own abilities. The player is also given options to choose a set of runes that can further boost some aspect of the selected champion, and two summoner spells as universal abilities.

Players found out that a champion's performance varies when picked against another champion. Some match ups are heavily one-sided where we call it one champion counters another champion. A champion's performance may also be affected by its teammate. When a champion is able to enhance another champion's performance significantly, we call it champion synergy. There are many other factors that could determine the result of a ranked match, such as player skill level (including but not limited to how familiar the player is with the champion, player's mechanical skill, player's game knowledge and decision making), the runes and summoner spells chosen, item purchased, luck, etc.

In this study, I created a multi-input feed forward neural network to predict the match result of a game in League of Legends, given selected champions and their past game statistics. All of the data were gathered using Riot Games API, and the statistics were derived and calculated from the raw data. In my experiment, I was able to predict the result of a match with 55.6% accuracy before the game starts using champions selected and champions related statistics based on previous games.

2 Related Work

There are existing applications using data analysis such as websites like u.gg and blitz.gg. These websites give players the statistics of game related data, such as if one champion counters or has synergy with another champion, and information about their selected champion, such as the runes and item builds that has higher win rate using probabilities. These information are very helpful for the player, but they also have some limitations too. Since League of Legends is a 10 players game, there could be very complicated relationship between the 10 selected champions, and statistics of these information could not be found on these websites. For example, we could not see the statistics of a champion when this champion counters a champion on the enemy team, but is also counted by another champion on that team, while having synergy with an ally champion.

There is also a research done using gradient boosted trees with logistic regression to predict the match results of players within Gold ranks with an accuracy of 93.6% using pre-game knowledge, such as champion selected, as well as in-game knowledge, such as damage dealt, gold earned, kills to deaths ratio to predict the outcome of the game [1]. There is another research done using various deep learning models to predict the outcome of a match given the player-champion data such as how familiar a player is with the selected champion [2]. Unlike these predictors, my model only uses pre-game knowledge of the selected champions, without considering the effects from the player, because enemy players stays unknown during champion select, which makes it impossible to find their player data.

3 Approach

In my approach, I created and trained six different neural network models to compare and predict the match result of a game in League of Legends, given selected champions and their statistics. All of the data were gathered using Riot Games API, and the statistics were derived and calculated from the raw data.

3.1 Dataset

Using the public Riot Game API, I gathered 9805 unique ranked solo queue matches in Master rank and above (top 0.3% players) during patch 11.20. Master rank and above was picked in order to best eliminate the impact of difference in player skill and focus the study on champion selection. This dataset contains informations such as champions selected for a match, result of a match, gold and experience point earned at 15 minutes. The dataset is split into statistic set of 6805 matches, training set of 2000 matches, validation set of 200 matches, and testing set of 800 matches. Since my study is focusing on prediction with only pre-game knowledge, the champion statistics were calculated using only the statistic set, and it was treated as game histories. Champion statistics calculated are:

- **Champion Positional Win Rate:** A percentage $\in [0, 1]$ indicating the champion’s win rate given a position. If no matches are found given champion and its position in the statistic set, 50% is assumed, indicating that the probability of winning and losing is the same for the champion in the given position.

- **Champion Positional Counter Win Rate:** A percentage ∈ [0, 1] indicating the champion's win rate given an opponent champion and the position. If no matches are found given a champion, an opponent champion and the position in the statistic set, 50% is assumed, indicating that the probability of winning and losing is the same when the two champions are playing against each other in the same position.
- **Champion Synergy Win Rate:** A percentage ∈ [0, 1] indicating the champion's win rate given an ally champion. If no matches are found given a champion and an ally champion in the statistic set, 50% is assumed, indicating that the probability of winning and losing is the same when the two champions are on the same team.
- **Champion Counter Gold Difference Average:** An integer ∈ (-∞, ∞) indicating the champion's total gold income relative to its opponent with the same position at the 15 minutes mark in the training set. A positive/negative number X means the champion earned X more/less gold comparing to its opponent in the same position. If the game ended before 15 minutes, the total gold earned when the game ended would be used. If no matches are found given a champion, an opponent champion, and the position, 0 is assumed, indicating both champion earn the same amount of gold.
- **Champion Counter Experience Point Difference Average:** An integer ∈ (-∞, ∞) indicating the champion's total experience points gained relative to its opponent with the same position at the 15 minutes mark in the training set. A positive/negative number X means the champion gained X more/less experience points comparing to its opponent in the same position. If the game ended before 15 minutes, the total experience points gained when the game ended would be used. If no matches are found given a champion, an opponent champion, and the position, 0 is assumed, indicating both champion gain the same amount of experience points.

3.2 Feature Selection

In my approach, I split the data used for the neural network into categorical and statistical. Categorical data consist the 10 champions selected and the match result, and statistical data consist all the statistics I calculated previously using the statistical set.

One-hot encoding is used to transform the 10 champions selected into a vector of shape 10 x 157, in the order of top, jungle, middle, bottom and support for blue side and red side respectively. The vector of size 157 represents the index of all 157 champions available during patch 11.20, sorted by their name. I also used the same process to convert the match result into a vector of shape 2, where (1, 0) represents blue team won and (0, 1) represents red team won.

All 10 champions' statistical data of a match are transformed into a vector of shape 10 x 8, based on the champion's position and team information. The vector of size 8 represents the statistics of the champion, in the following order: champion counter gold difference, champion counter experience point difference, champion positional counter win rate, champion positional win rate, and four champion synergy win rate with each of the teammate in the order of top, jungle, middle, bottom and support.

3.3 Models

I chose multi-input feed forward neural network as the machine learning model, and I used Tensorflow to create my model. Regularizer, batch normalization, dropout, and early stopping were used to prevent the model from overfitting. All Dense layers have X nodes and use l2 with penalty Y as their regularizer. X and Y are parameters that were configured differently for each model. The model is defined as follow:

- Input_1: (None, 10, 157) => (None, 10, X)
- Dense_2: Input_1 => (None, 10, X)
- Dense_3: Dense_2 => (None, 10, X)
- Input_4: (None, 10, 8) => (None, 10, X)
- Dense_5: Input_4 => (None, 10, X)
- Dense_6: Dense_5 => (None, 10, X)
- Concatenate_7: [Input_1, Input_2] => (None, 10, X)
- Dense_8: Concatenate_7 => (None, 10, X)
- Flatten_9: Dense_8 => (None, 10 * X)
- Output_10: Flatten_9 => (None, 2)

- Each Dense layer uses ReLu as activation function and l2(Y) as regularizer.
- The Output layer uses softmax as activation function.
- The model uses Adam as optimizer, and categorical cross entropy as loss function.
- Batch normalization layers are added between each Dense layer.
- Two Dropout layer with 20% drop rate are added between Dense_2 and Dense_3 and between Dense_5 and Dense_6.
- Model is trained with 40 epochs.
- Early stopping is used when validation loss does not decrease after three epochs.

3.4 Results

For this experiment, six neural network models were created using number of nodes and l2 penalty as parameters for all hidden layers. Each model was trained with 2000 matches, validated with 200 matches and tested with 800 matches. Table 1 shows the testing accuracy for each model and the epochs passed.

Table 1 Test epochs, accuracy and loss of the Models

Model	Epochs	Test Acc	Test Loss
Nodes = 50, l2 = 0.005	23	54.12%	0.8768
Nodes = 50, l2 = 0.01	25	53.75%	0.8012
Nodes = 50, l2 = 0.02	16	53.13%	0.8516
Nodes = 200, l2 = 0.005	22	53.13%	0.8116
Nodes = 200, l2 = 0.01	18	54.34%	0.7865
Nodes = 200, l2 = 0.02	19	55.62%	0.7500

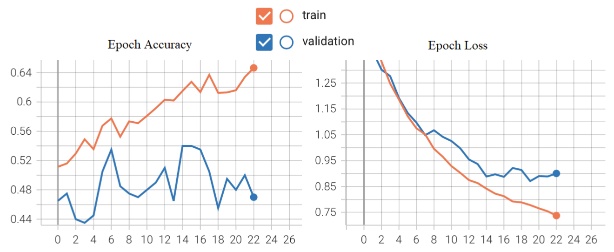

Fig. 1 Train and validation loss and accuracy of the Model 1

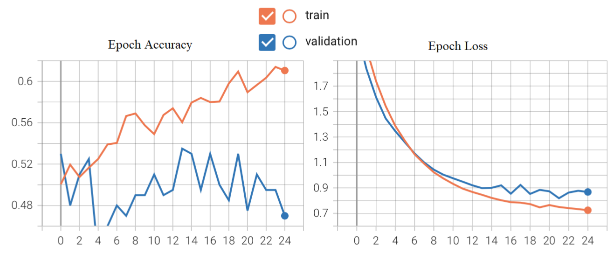

Fig. 2 Train and validation loss and accuracy of the Model 2

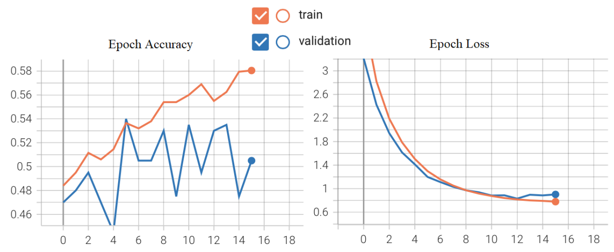

Fig. 3 Train and validation loss and accuracy of the Model 3

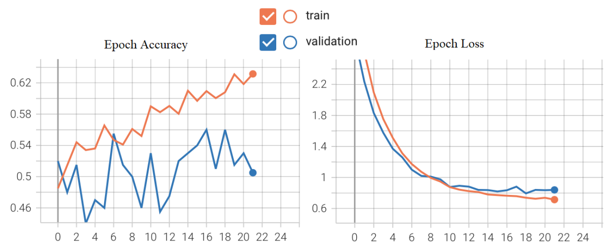

Fig. 4 Train and validation loss and accuracy of the Model 4

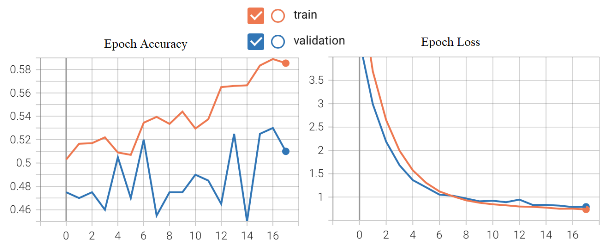

Fig. 5 Train and validation loss and accuracy of the Model 5

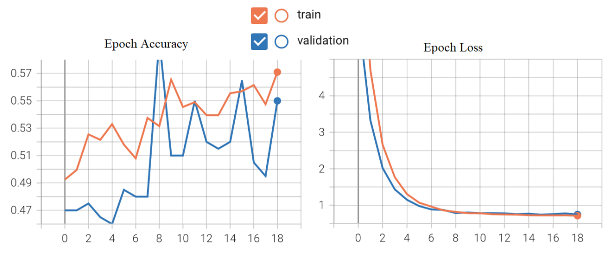

Fig. 6 Train and validation loss and accuracy of the Model 6

4 Evaluation

Based on the experiment result shown in Table 1, model 6 has the highest test accuracy at 55.62%. Comparing models with 50 nodes in the hidden layers to models with 200, the models with more nodes in the hidden layers have lower test loss when l2 penalty are the same, and two out of three models have slightly higher accuracy. Interestingly, models with 50 nodes in the hidden layers has an inverse relationship between l2 penalty and test accuracy, while models with 200 modes in the hidden layers have their test accuracy increased when l2 penalty is increased. There is also an inverse relationship between test accuracy and test loss for the models with 200 nodes in the hidden layers. None of the six models finished training for 40 epochs, they all stopped early between 16 to 25 epochs. From Fig. 1-6, we can see that all six models had a sizable gap between training accuracy and validation accuracy. This means that our models are overfitting the training data.

5 Future Work

There are still a lot more things that I could do to try further improve the accuracy of the models. Collecting more data to calculate more accurate champion statistics will for sure improve the models. Currently, there are a lot of missing statistics in my dataset that are filled with default values. This means that I was not able to use accurate statistics during training and testing. Also, I could have taken more variables into account, such as champion average kill death assist ratio, summoner spell and runes picked, champion average objective taken, and some other statistics.

6 Conclusion

In my experiment, I configured 6 multi-input feed forward network to predict the match result of solo queue ranked games from Master and above during patch 11.20. The best performing model has an accuracy of 55.62%, while the least performing model has an accuracy of 53.13%. Data such as the 10 champions selected for the match and statistics based on past games with these champions were used to train, validate, and test the model. I found that only using the data I acquired was not enough to accurately predict the outcome of a match, since there are many other factors that could affect the result of a match. And not having enough data to calculate statistics to help determine the match result made it even harder. Despite this, my experiment shows that it is possible to predict the match result before the game starts. With more data, this prediction should be more accurate.

References:

[1]. Lin, Lucas. “League of Legends Match Outcome Prediction.” (2016).
[2]. Do, Tiffany D. et al. “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* (2021): n. pag.
[3]. League of Graphs. https://leagueofgraphs.com/rankings/rank-distribution