OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

INFORMATIK UND
MATHEMATIK

Comparative Analysis of Predictive Performance: Neural Networks vs. GRU in League of Legends Match Outcome Prediction

Bachelorarbeit
von

Moritz Palm
Matrikelnummer: 1234567

Fakultät Informatik und Mathematik
Ostbayerische Technische Hochschule Regensburg
(OTH Regensburg)

Gutachter: Prof. Dr. Brijnesh Jain
Zweitgutachter: Prof. Dr. Timo Baumann

Abgabedatum: November 14, 2023

Herr
Moritz Palm
Konrad-Adenauer-Allee 55
93051 Regensburg

Studiengang: Künstliche Intelligenz & Data Science

1. Mir ist bekannt, dass dieses Exemplar der Bachelorarbeit als Prüfungsleistung in das Eigentum des Freistaates Bayern übergeht.

2. Ich erkläre hiermit, dass ich diese Bachelorarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtlich und sinngemäße Zitate als solche gekennzeichnet habe.

Regensburg, den November 14, 2023

Moritz Palm

Contents

1 Introduction

esports is highly relevant due to it being a huge and strongly growing market. Esports and mobas in particular are hard to understand and follow. A live game prediction view can help fans understand the action and decisions made better and help immerse the audience by detecting upsets and swings in win probability. many games are hard to understand, due to lots of information being displayed with very little explanation a win predicition graph can help viewers understand the action and the significance of certain plays better, thus increasing engagement and enjoyment. riot games has already implemented their own proprietary win prediction a win prediction model can also help players make more informed decisions about what the optimal path of actions is

the model should be able to answer the question, if team a is far enough ahead to win or if team b with their hyperscaling heroes can come back and win

2 Background

2.1 League of Legends

League of Legends is a Multiplayer Online Battle Arena (MOBA) game developed by Riot Games. MOBA games are a subgenre of real-time strategy games in which two teams, typically consisting of five players ('summoners') each, compete against each other with each player controlling a single character [1], called 'champion'. It is one of the most played video game genres CITATION NEEDED and attracts millions of players and fans watching the professional scene. Most MOBAs differ only slightly in terms of basic gameplay or map layout, but vary in details such as champions, abilities, graphics etc. As League of Legends is the most played game in the MOBA genre, we will focus on it. At the start of the game, each player picks a champion from a pool of currently 165 champions, each with distinct abilities and characteristics. The map consists of two bases which are connected by three lanes. Each base contains a large structure, the so called 'nexus', which is protected by two turrets. The goal of the game is to destroy the enemy nexus. The bases are connected by three lanes, separated by a jungle. Non-player characters (NPCs), so called 'minions', spawn in regular intervals and advance down the lanes. Killing minions grants gold and experience points (xp), which are used to improve different attributes by buying items or increasing ones level. It is conventional that the players split up at the start of the game, with one player going to the top lane, one to the mid lane and two players to the bottom lane. The last player gets his gold and xp from killing neutral monsters in the area between each lane, commonly referred to as 'the jungle'. It also contains two large neutral monsters, Baron Nashor and a dragon, which require multiple team member to be killed and grant improvements to the whole team. Larger fights are usually centered around either destroying a turret or killing a large neutral monster. Every year, the game enters a new 'season', where it undergoes major challenges. In between seasons, the developers release smaller patches every two weeks, which are usually aimed at changing the strength of different champions. These patches have the potential to cause major disruption to the best way the game is played (the so called 'meta'), for example by introducing a new champion.

The players need to pick a champion that fits best into the team, taking into account the team strategy, the damage composition, the role the player has been assigned etc., while also selecting a champion the player has played before and can play well. This is a very critical, yet highly difficult choice, as in theory there are $\binom{165}{10} = \frac{165!}{10! \cdot (165-10)!}$ different combinations of champions.

a modified elo system is used to determine the skill of an individual player, which is the most important factor in which players get matched against each other

2.2 Neural Networks

Artificial Neural Networks (ANNs) have been originally designed to simulate the way the human brain processes information. The fundamental unit in an ANN is called a neuron, first proposed by McCulloch and Pitts [2]. A neuron is a function which takes a number of values $\mathbf{x} = (x_1, x_2, ..., x_m), m \in \mathbb{N}$ and outputs its activation a. The weighted input z is calculated as the dot product of the input vector $\mathbf{x}$ and the weight vector $\mathbf{w} = (w_1, w_2, ..., w_m)$ with the bias b added:

$$z = w_1 x_1 + ... + w_m x_m + b = \mathbf{w}^T \mathbf{x} + b. \tag{1}$$

z is then passed through an activation function ϕ, which provides non-linearity: $a = \phi(z) = \phi(\mathbf{w}^T \mathbf{x})$. Common activation functions include sigmoid functions, Rectified Linear Unit (ReLU) and more. The Multi-Layer Perceptron (MLP), first introduced by Rosenblatt [3] organizes neurons in layers. Input data is fed into the input layer, and computations propagate forward through the hidden layers to produce an output. In the simplest form of an ANN, every neuron in a layer is connected to every neuron in the subsequent layer, forming a fully connected architecture. There are no other connections, making it a feed-forward network, and it constitutes a directed acyclic graph. A fully connected feed-forward network network can be described as a series of matrix-vector multiplications for each layer l:

$$\mathbf{a}^{(l)} = \phi(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}) \tag{2}$$

In equation 2, $\mathbf{a}^{(l)}$ denotes the activation of layer l, $\mathbf{W}^{(l)}$ denotes the weight matrix of layer l and $\mathbf{b}^{(l)}$ is the corresponding bias vector with d being the number of neurons per layer. If l is the output layer L, the vector $\mathbf{a}^{(L)}$ represents the predicted output $\hat{y}$ of the network.

To allow the network to approximate any measurable function, at least one hidden layer is required [4]. During the training process, the weights and biases are iteratively adjusted using the backpropagation algorithm to minimize the loss function E. For regression tasks, a commonly used loss function is the Mean Squared Error (MSE) Loss

$$E_N = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3}$$

where N is the number of samples [5]. For binary classification, the Cross-Entropy Loss (CEL)

$$E_N = \frac{1}{N} \sum_{k=1}^{N} y_k \ln \hat{y}_k + (1 - y_k) \ln (1 - \hat{y}_k) \tag{4}$$

is widely used [6].

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) build upon the foundation of feed-forward neural networks. They are designed to handle sequential data by introducing the concept of

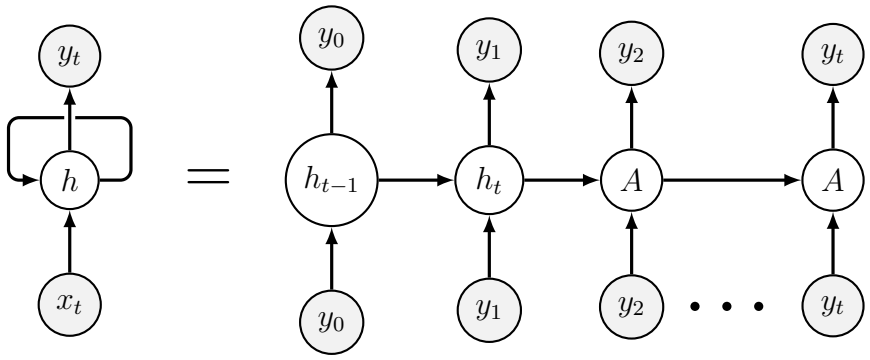

Figure 1: Unrolling of a RNN over time

recurrence. In an RNN, the output at each time step depends not only on the current input but also on the network's previous internal state or hidden state. This allows RNNs to capture dependencies over time, making them suitable for tasks involving sequential input such as speech and language [7]. The first fully connected RNN was proposed by Elman [8]. RNNs maintain a 'state vector' in their hidden units (see figure 2), implicitly containing information extracted from all the past elements of the sequence [7]. The output $\hat{y}$ of a neural net can be calculated by

$$\hat{y}_t = \tanh(Vh_t + b_o) \tag{5}$$

where V and b_o are the weight matrix and the bias vector of the cell output. In addition to calculating the output, the forward propagation involves updating the hidden state h_t at every timestep t:

$$h_t = \begin{cases} 0, & \text{if } t = 0 \\ \sigma(Wx_t + Uh_{t-1} + b_h), & \text{otherwise} \end{cases} \tag{6}$$

where σ is the sigmoid function, U and b_h are the weight matrix and the bias of the hidden state and W is the weight matrix of the input. For an RNN with just one recurrent layer, one hidden weight matrix W_h is sufficient, as the weights are shared across timesteps.

Bengio et al. [9] showed, however, that RNNs have challenges with exploding or vanishing gradients, especially in long sequences, which can affect their ability to capture long-range dependencies [10].

2.4 GRU

In order to overcome the exploding/vanishing gradient problem of vanilla RNNs, gated networks like the Long Short-Term Memory (LSTM) [11] and Gated Recurrent Unit (GRU) [12] have been developed [13]. As they introduce an increased number of parameters compared to traditional RNNs, gated networks like the LSTM and GRU demand greater computational power [14]. Compared to the LSTM network, GRU reduces the number of gate networks to two, thus being simpler to implement and

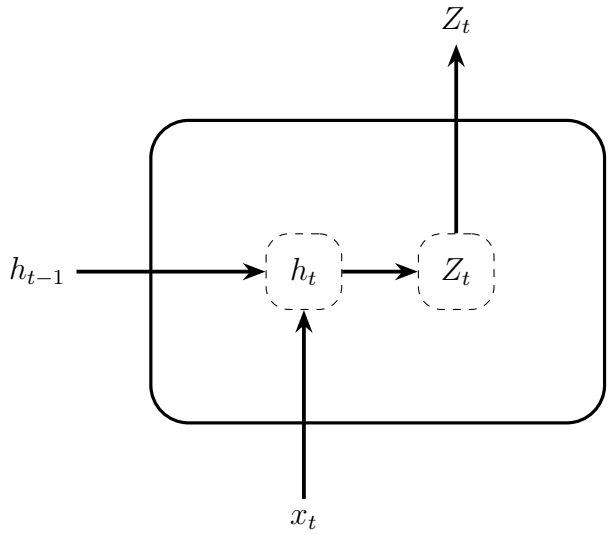

Figure 2: Elman Recurrent Unit

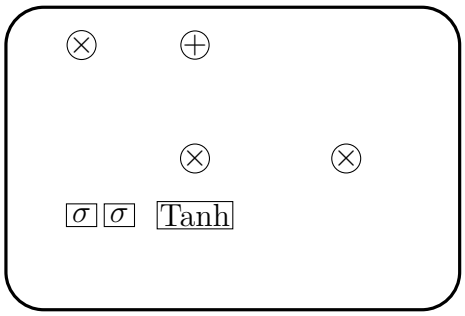

Figure 3: Gated Recurrent Unit

compute [12]. Chung et al. even found that GRU is at least comparable to LSTM most of the time [15]. The gates control the activation of each hidden unit. The reset gate is calculated by

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{7}$$

and the update gate z_j by

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{8}$$

[14]. The hidden state update is a linear interpolation between the previous activation h_{t-1} and the candidate activation $\tilde{h}_t$, where the update gate z_t influences how much the hidden state is changed [15]:

$$h_t = (1 - z_t) \odot h_t + z_t \odot \tilde{h}_t \tag{9}$$

with

$$\tilde{h}_t = g(W_h x_t + U_h(r_t \odot h_{t-1} + b_h) \tag{10}$$

. In equations 9 and 10 $\odot$ denotes the elementwise (Hadamard) multiplication and the function g in equation 10 is the activation function.

2.5 Related work

Utilizing machine learning methods to extract information from data generated by e-sport games is an area of ongoing research. A lot of scientific research focuses on the similar MOBA DotA 2, which has easier and more fine-grained data collection methods (see section 3.1). Due to the high similarity between these two games, it is to be expected that any findings for one game can be replicated and used for the other game with minimal adaptations. Nevertheless, to ensure a fair comparison, both games are presented separately below.

In DotA 2, a wide variety of algorithms have been used. Yu et al. [16] trained a RNN on 71,355 matches and achieved an accuracy of 0.7083 at the half-way point of a match, which according to their analysis is on average at 20 minutes. Wang [17] compared Logistic Regression with a Feedforward Neural Network (FNN) trained on up to 911,468 matches, with Logistic Regression (LR) achieving a slightly better accuracy (0.6104) than the FNN (0.588).

Silva et al. have used RNNs to predicting the winner using data of different time intervals. They achieved an accuracy of 75% when using data from between the 10 and 15 minute mark. An evaluation of LSTM resulted in lower accuracy, most likely due to the large amount of data required [18].

3 Data

As two very different experiments are compared against each other, two different datasets need to be constructed: one dataset containing all relevant information prior to the start of the game and one dataset containing only the temporal information from the beginning of the game.

3.1 Data Collection

High-Rank Matches The focus of data acquisition was directed towards high-rank matches, in which a mix of excellent amateur and professional players play. Lower rank matches are not considered due to their higher unpredictability as less skilled players make huge, game-changing mistakes way more often. Pro matches, defined as professional players playing with their respective teams in an esport tournament or league, were not included as there are way less matches and they are not available through the official Riot Games API. High rank matches in this context are defined as having at least one player holding the rank of Master, Grandmaster or Challenger. Riot Games themselves considers any rank above Diamond 3 as 'Elite' [19], but we raise this bar just slightly to only include any rank above Diamond 1. Due to the fact that for a match to be included in the dataset, only one out of ten players needs to hold one of the aforementioned highest ranks, some slightly lower ranked players are also present in the dataset. These ranks combined account for the top 0.2% of all players [20].

Riot Games API The primary source of data stemmed from the Riot Games API, a comprehensive repository of information pertaining to League of Legends gameplay. The Riot Games API provided access to a plethora of essential data points, including champion statistics, general match information, timeline details, and player-specific information. These variables collectively form a comprehensive and multifaceted dataset crucial for the development of an effective predictive model.

Other Data Sources However, not all pertinent data were available directly from the Riot Games API. These include the winning chance of each champion and statistics on how each player performs on each relevant champion. To address this limitation, a web-scraping approach was employed to gather additional relevant information. The amalgamation of Riot Games API data and web-scraped data was meticulously organized and stored in a Database. This central repository served as the foundational storehouse from which distinct datasets were constructed, ensuring an organized and structured approach to data management.

Regions Multiple regions were included in the data collection process, including Europe West (EUW), Europe Nordic & East (EUN), Korea (KR), and North America (NA). This regional diversity contributes to the model's generalizability across different player bases and playing styles.

Period of Time All matches included in the dataset were played in season 13 and on patch 20. It is important that all matches are played on the same patch, as a patch may cause major shifts in the balance of the game, thus making certain strategies and champions way better than others.

In summary, the data collection process for this study involved a dual-pronged approach, leveraging the extensive resources provided by the Riot Games API alongside a targeted web-scraping strategy. The resulting raw dataset containing $20,513$ and $3,972$ matches in the pre-game and in-game datasets respectively, stored in a PostgreSQL Database, reflects a comprehensive compilation of high-rank amateur League of Legends matches.

3.2 Data Processing

3.2.1 Pre-Game Dataset

The raw pre-game dataset contains 368 columns which can be categorized into four distinct groups: General Match Information, Player Information, Champion Information and Player-Champion Information. General match details, such as the patch number, are exclusively utilized for validation purposes and are excluded from the final dataset.

Player Information Player Information feature x_p is a two-dimensional vector containing information about the player. This includes the account level, serving as an indicator of the player's accumulated gaming experience, and the player's rank, functioning as a metric for assessing the player's skill level.

Champion Information The Champion Information feature $\mathbf{x}_c$ is composed of

However, it is noteworthy that a limitation inherent in these metrics lies in their aggregation across all player ranks, reducing their specificity to the ranks under analysis.

Player-Champion Information This feature vector contains information about the player on a specific champion. It encompasses metrics such as the average amount of gold earned by the player across all matches played on the champion during Season 13. Costa et al. [21] found that the most pivotal feature within this category is the player's win rate while piloting this champion. Unfortunately, this information is not readily available to be extracted by our web scraper, necessitating its omission from the dataset.

3.2.2 In-Game Dataset

The raw in-game dataset contains 381 features describing the current state of the game after every minute. It only consists of statistics of each player like the damage done by him to other champions, the level of his champion or his total gold earned thus far.

4 Experiments

5 Results

6 Discussion

7 Conclusion

Bibliography

[1] M. Mora-Cantallops and M.-Á. Sicilia. MOBA Games: A Literature Review. *Entertainment Computing* 26, 128–138, May 2018.

[2] W. S. McCulloch and W. Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The bulletin of mathematical biophysics* 5(4), 115–133, Dec. 1943.

[3] F. Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* 65(6), 386–408, 1958.

[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* 2(5), 359–366, Jan. 1989.

[5] K. Liano. Robust Error Measure for Supervised Neural Network Learning with Outliers. *IEEE Transactions on Neural Networks* 7(1), 246–250, Jan. 1996.

[6] Z. Zhang and M. Sabuncu. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In: *Advances in Neural Information Processing Systems*, 2018.

[7] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature* 521(7553), 436–444, May 2015.

[8] J. L. Elman. Finding Structure in Time. *Cognitive Science* 14(2), 179–211, 1990.

[9] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent Is Difficult. *IEEE Transactions on Neural Networks* 5(2), 157–166, Mar. 1994.

[10] I. Sutskever. "Training Recurrent Neural Networks". PhD thesis. Toronto, ON, Canada: University of Toronto, 2013. URL: `https://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf` (visited on 11/01/2023).

[11] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation* 9(8), 1735–1780, Nov. 1997.

[12] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. *Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation*. Sept. 2014. arXiv: `1406.1078 [cs, stat]`. URL: `http://arxiv.org/abs/1406.1078` (visited on 11/03/2023).

[13] G. Van Houdt, C. Mosquera, and G. Nápoles. A Review on the Long Short-Term Memory Model. *Artificial Intelligence Review* 53(8), 5929–5955, Dec. 2020.

[14] R. Dey and F. M. Salem. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. In: *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1597–1600, Aug. 2017.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. Dec. 2014. arXiv: 1412.3555 [cs]. URL: http://arxiv.org/abs/1412.3555 (visited on 09/26/2023).

[16] L. Yu, D. Zhang, X. Chen, and X. Xie. *MOBA-Slice: A Time Slice Based Evaluation Framework of Relative Advantage between Teams in MOBA Games*, July 2018.

[17] W. Wang. "Predicting Multiplayer Online Battle Arena (MOBA) Game Outcome Based on Hero Draft Data". MA thesis. Dublin, National College of Ireland, Dec. 2016. URL: https://norma.ncirl.ie/2523/ (visited on 10/05/2023).

[18] A. L. C. Silva, G. L. Pappa, and L. Chaimowicz. Continuous Outcome Prediction of League of Legends Competitive Matches Using Recurrent Neural Networks. *SBC-proceedings of SBCGames*, 2179–2259, 2018.

[19] R. Games. */Dev: Balance Framework Update - League of Legends*. June 2020. URL: https://www.leagueoflegends.com/news/dev/dev-balance-framework-update/ (visited on 11/09/2023).

[20] R. Games. *Ranked Tiers, Divisions, and Queues*. Aug. 2023. URL: https://support-leagueoflegends.riotgames.com/hc/en-us/articles/4406004330643-Ranked-Tiers-Divisions-and-Queues (visited on 11/09/2023).

[21] L. M. Costa, R. G. Mantovani, F. C. Monteiro Souza, and G. Xexéo. Feature Analysis to League of Legends Victory Prediction on the Picks and Bans Phase. In: *2021 IEEE Conference on Games (CoG)*, 01–05, Aug. 2021.

List of Figures

List of Tables

List of Abbreviations

ANN Artificial Neural Network

CEL Cross-Entropy Loss

FNN Feedforward Neural Network

GRU Gated Recurrent Unit

LR Logistic Regression

LSTM Long Short-Term Memory

MLP Multi-Layer Perceptron

MOBA Multiplayer Online Battle Arena

MSE Mean Squared Error

NN Neural Network

NPC non-player character

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

xp experience points