Midterm Project Report: Deep Learning-Based Sequential Model for Predicting Victories in Video Games

Ali Mohammadi Esfahani – 101245278
Azhar Syed – 101150942
Françoise Blanc – 101209287
Sandeep Reddy Venna – 101214885

Introduction

This midterm project report consists of an update on the main activities completed so far. As a first step, our team conducted an extensive literature review on predicting victories using sequential data. Then, our team pursued an analysis of the game log formats to determine which features could be used as sequential data. After the extraction of the sequential data from the game logs, our team performed data preprocessing and data engineering. Then, a Long Short-Term Memory (LSTM) model, which is a variant of a Recurrent Neural Network (RNN), was implemented to determine how the use of sequential data can predict victories. One change from our initial proposal is that our work will not limit itself to the use of RNNs since our literature review showed that other deep neural network (DNN) algorithms might provide better performance. Therefore, our main goal is to investigate the performance of DNNs, including RNNs, in predicting victories in the video game *Tactical Troops: Anthracite Shift* using sequential in-game data.

Main Activities

Literature Review Overview

A literature review was first completed to evaluate what has been done and what still needs to be done regarding the use of sequential data as input to DNNs to predict victories in video games. The first section of the literature review provides an overview of the top-performing solutions of the "IEEE BigData 2021 Cup - Predicting Victories in Video Games". This section includes the main key points of the leading solutions. It also includes future work that could improve the prediction accuracy, which leads to our proposal: the use of game logs to model sequential data for predicting victories. This section of the literature review was developed by summarizing Assignment 2.

Since predicting victories using sequential data is a task that falls under time series classification (TSC), the second section of the literature review offers an overview of how DNNs can be used for TSC. DNNs are characterized by hidden layers capable of extracting hidden features within time series. In a recent review on DNNs for TSC [1], the authors compared several DNNs against univariate and multivariate time series datasets. Their work showed that Fully Convolution Neural Networks (FCN) and deep Residual Networks (ResNet) performed better than other tested DNN algorithms. The authors did not consider RNNs due to their limitations, such as computational complexity and difficulty in learning long-term dependencies. Since the sequential data provided in the data logs of the challenge are relatively short, the latter limitation might not affect our project. RNNs, and RNN variants like LSTM, should still be considered as valuable algorithms for our project. The literature review then summarizes two papers [2], [3] in which the authors explored sequential data and DNNs, mainly variants of RNN, to predict victories in video games. Predicting victories or other outcomes using machine learning is also a common research topic in sports prediction. Therefore, the literature review also outlines the key points of two papers [4], [5] in which the authors considered DNNs using sequential data as input to predict sports outcomes.

Data Description and Preprocessing

The raw datasets provided by the challenge's organizers primarily consist of two types of data: image-based data (i.e., screenshots of the game) and game logs. As we want to extract key features, like player behaviour, by considering the previous game states, we have decided to disregard image-based data in our model for now. The image-based data does not contain the previous states of the game but only the screenshot of the video game at the point of prediction. The challenge's organizers provided three kinds of game logs: truncated logs, flattened logs, and tabular data. It is not a straightforward process to model the log files as sequential data, as none are sequential out of the box. The changes in game states (e.g., a unit losing a health point (HP)) are found in the truncated logs. We used these state changes to create sequential data. We start with the initial state of the features at time step 0. Subsequently, for every state change recorded in the log, we add a new timestep to our dataset for the game. This way, we created a new sequential dataset with the number of time steps equal to the number of state changes until the point of prediction. Throughout the training dataset (i.e., 38658 games), the number of timesteps in each game varies from 2 to 119. As the number of timesteps is not equal for all games, we had to introduce the mechanism of padding and masking in our model. A more detailed description of the padding and masking mechanism is provided in the next section. Only RNN variants, such as LSTM and Gated Recurrent Unit (GRU), can use the masking functionality. Therefore, for other DNN algorithms, which also require the same sequence lengths, we will create sequences of the same length through interpolation or other techniques.

Each player in the game has four units. Since the number of players is two, there are eight units in each game. For our initial experiment, we only considered the sequential data of HP of each unit until the point of prediction as at least two of the best-performing teams [6], [7] of the challenge have shown that HP is one of the most informative features in predicting the winner. We normalized the HP by considering the ratio of current HP and maximum HP of the unit to ensure that the values are in the range of 0 to 1. Data normalization generally leads to more effective learning (i.e., faster convergence) when using DNNs. In some of the games, the HP of all the units remain unchanged. As we are only considering the HP of units to make a prediction, the prediction will be random in such cases. We intend to handle this limitation by accounting for more general features such as unit type and game mode in our final model. We may also consider creating two input channels (i.e., one for each player) for the DNNs.

Initial Model Implementation

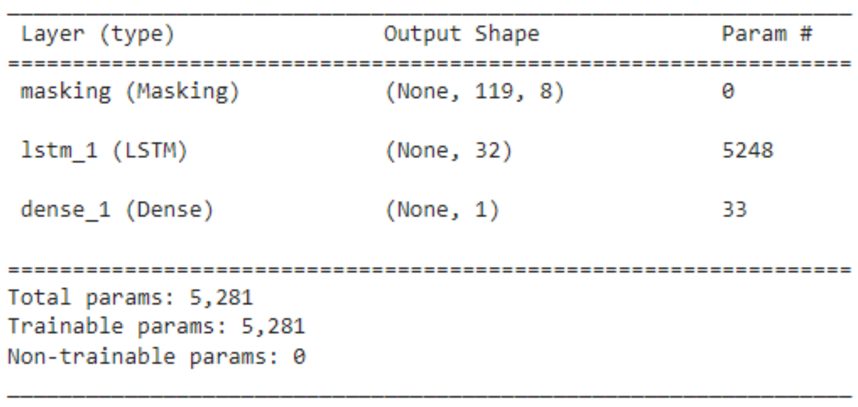
For our first attempt at predicting victories using sequential data, we implemented an LSTM model, which is a variant of RNN. LSTM was selected over a simple RNN since it better captures long-term dependencies. Additionally, LSTM performed well in various studies [2], [3], [5] that have considered DNNs for predicting victories in games or sports.

Our initial model was built using Python version 3.7.12. The Python codes for the data preprocessing and our model are available in a GitHub repository[1]. Python was chosen for its simplicity and access to great libraries and frameworks for artificial intelligence and machine learning. For designing the LSTM, we employed Keras, which is built on top of Theano and TensorFlow. Keras is an open-source Python library for developing and evaluating deep learning models. As mentioned in the previous section on data preprocessing, most DNN algorithms, including LSTM, require that sequence input data have the same length. Therefore, our sequential data were first preprocessed using the padding and masking techniques provided by the Kera preprocessing functionalities. The padding technique was first used to ensure all individual sequences have the same length as the longest sequence found in the training dataset (i.e., 119 time steps). The sequences were "padded" with the value "99.0". This value was selected over the default value of "0" since HP can take a value of "0".

For building our model, we used the sequential API-based Keras model, allowing us to create a model layer-by-layer. The layers of our model are shown in Figure 1. The first layer is a masking layer which tells the model to ignore the time steps that have been padded with the value "99.0". As a next layer, we used an LSTM layer of 32 neurons with a hyperbolic tangent activation function. The last layer is a dense layer with

[1] Python codes are available on https://github.com/azhartalha/PredictingVictories

a single neuron (i.e., output layer with a sigmoid activation function). This crafted neural network model is trained with Adam optimization and cross-entropy loss for binary classification. Adam optimization was chosen as it is considered, in most cases, the best among the adaptive optimizers [8]. The model was trained on the input data with a batch size of 1000 (i.e., number of samples processed before the model is updated) for 150 epochs (i.e., the number of complete passes through the training dataset). Using an 80:20 split ratio for the training and validation data set, the designed model achieved about 70% accuracy.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 masking (Masking)           (None, 119, 8)            0

 lstm_1 (LSTM)               (None, 32)                5248

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 5,281
Trainable params: 5,281
Non-trainable params: 0
_________________________________________________________________
```

Figure 1. LSTM Model (Keras model summary).

Model Performance Evaluation

While designing and configuring deep learning models, many choices can be made for various hyperparameters. To get more confidence in the model, the performance of the created model must be validated. We used Keras functionalities to automatically pick the validation dataset (i.e., 20% of the training dataset), then we assessed the performance of the designed LSTM model on that validation dataset for every epoch. While fitting the model to the data, Keras can report on several metrics such as loss functions and accuracy. Figure 2 and Figure 3 provide the accuracy and loss over epoch. Figure 2 shows that training and validation accuracy increases similarly and reaches a plateau for 150 epochs with a batch size of 1000. This demonstrates that the model is a good fit with 150 epochs. Continued training of a good fit will likely lead to an overfit. A good fit is the goal of the learning algorithm and exists between an underfitted and an overfitted model.

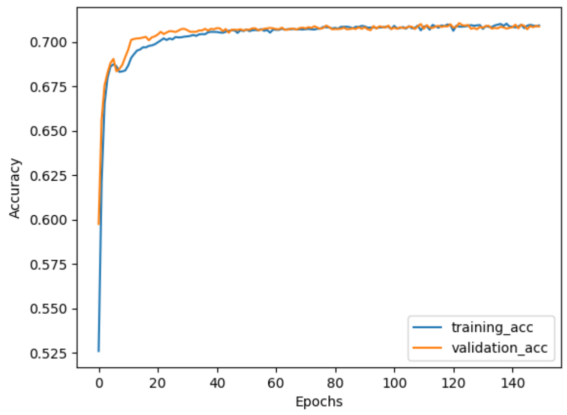


Figure 2. Training and validation accuracy over epoch.

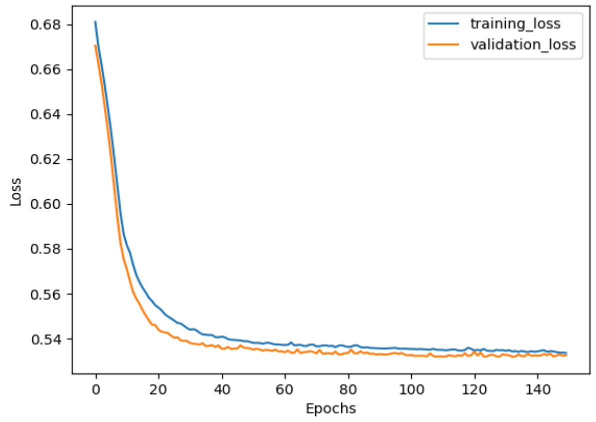


Figure 3. Training and validation loss over epoch.

Conclusion

Our preliminary data preprocessing and LSTM model implementation obtained an accuracy of about 70% in predicting victories using sequential data from the game logs. As we move forward, we expect significantly better prediction accuracy since the next activities of our project will mainly look at enhancing the data features and the learning models. The next activities of our project will include:

- More in-depth data analysis and feature engineering to improve the sequential data input;
- Improve the LSTM model;
- Investigate other types of DDNs-based sequential models based on the literature review;
- Decide on a hyperparameter optimization strategy;
- Evaluate the models using the Area Under Curve (AUC) metric to compare our results with the challenge results;
- Prepare presentation; and
- Redact final report.

References

[1] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, Springer US, 2019, pp. 917–63, doi: 10.1007/s10618-019-00619-1.

[2] A. L. C. Silva, G. L. Pappa, and L. Chaimowicz, "Continuous outcome prediction of League of Legends competitive matches using recurrent neural networks," *Proceedings of SBGames 2018*, 2018, pp. 639-642.

[3] Z. Qi, X. Shu, and J. Tang, "DotaNet: two-stream match-recurrent neural networks for predicting social game result," *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM)*, 2018, pp. 1-5, doi: 10.1109/BigMM.2018.8499076.

[4] N. Watson, S. Hendricks, T. Stewart, and I. Durbach, "Integrating machine learning and decision support in tactical decision-making in rugby union," *The Journal of the Operational Research Society*, vol. 72, no. 10, Taylor & Francis, 2021, pp. 2274–85, doi: 10.1080/01605682.2020.1779624.

[5] Q. Zhang, X. Zhang, H. Hu, C. Li, Y. Lin, and R. Ma, "Sports match prediction model for training and exercise using attention-based LSTM network," *Digital Communications and Networks*, 2021, doi: 10.1016/j.dcan.2021.08.008.

[6] H. Xiao, Y. Liu, D. Du, and Z. Lu, "WP-GBDT: an approach for winner prediction using gradient boosting decision tree," *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 5691-5698, doi: 10.1109/BigData52589.2021.9671688.

[7] K. Tseng, "Predicting victories in video games: using single XGBoost with feature engineering: IEEE BigData 2021 Cup - Predicting Victories in Video Games," *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 5679-5682, doi: 10.1109/BigData52589.2021.9671454.

[8] D. Soydaner, "A comparison of optimization algorithms for deep learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 34, no. 10, 2020, doi: 10.1142/S0218001420520138.