Deep Learning-Based Sequential Model for Predicting Victories in Video Games

Ali Mohammadi Esfahani, Azhar Syed, Françoise Blanc, and Sandeep Reddy Venna

Abstract—The "IEEE BigData 2021 Cup - Predicting Victories in Video Games" is a data mining competition in which participants submitted solutions to predict victories in the video game *Tactical Troops: Anthracite Shift*. As none of the best-performing teams have considered the influence of previous game states in predicting the winner, this paper proposes an approach that uses a sequential dataset generated from game logs to feed into deep neural networks to predict victories. Among the six different deep neural networks implemented, a simplified Fully Convolutional Network model achieved the best performance with an Area Under the Curve score of 0.8447. Although our model did not surpass the performance of the best challenge models, our proposed approach has two significant advantages: the limited need for feature engineering and greater adaptability to other video games.

***Index Terms*—deep learning, deep neural networks, sequential data, winner prediction**

I. Introduction

The "IEEE BigData 2021 Cup - Predicting Victories in Video Games" [1] is a data mining competition in which participants submitted machine learning solutions to predict victories in the 2D turn-based video game *Tactical Troops: Anthracite Shift*. The predictions were made at a given time in a game using gameplay data provided to the participants in various formats, each offering a different representation of the game. Most of the best-scored teams [2], [3], [4], [5] employed exhaustive feature engineering and feature selection techniques on these various data formats prior to train and assess their proposed machine learning models. They all implemented gradient boosting decision trees (GBDT) variants, such as LightGBM and XGBoost. Their work resulted in better prediction by over 1% than the baseline model created by the challenge's organizers [6].

None of the best-performing teams have considered the influence of previous game states in predicting the winner at a given point in time. Some of the gameplay data formats allow to extract key features, like player behaviour, over a game to create sequential data. This sequential dataset can then be fed into Deep Neural Networks (DNN), such as Recurrent Neural Networks (RNN), to predict the winner of a game. DNNs are characterized by hidden layers capable of extracting hidden features within sequential data.

Therefore, to explore the potential of sequential data in making predictions, we propose to create a sequential dataset generated from game logs to train and evaluate six different DNNs to predict victories. The main contributions of this work can be summarized as follows:

- A review of the use of DNNs with sequential data for predicting victories;
- A demonstration of a new approach using sequential data to predict victories in the video game *Tactical Troops: Anthracite Shift* that none of the best-performing teams have considered;
- Development of a more general machine learning solution that can be easily adaptable to other video games or sports-related datasets;
- Training and evaluation of six different DNNs to predict victories; and
- Recommendations to improve the proposed solution.

The rest of this paper is organized as follows: In Section II, we first review related work in predicting victories, including a summary of the best-performing solutions to this challenge. Section III offers background information regarding the six chosen DNN algorithms to predict victories. Section IV details the methods and procedures to implement and assess the proposed DNN algorithms. Section V shows the model results, and then Section VI provides an analysis of our project outcomes. Section VII concludes our report with the key findings and possible future work.

II. Related Work

A. *Overview of the Top-Performing Solutions*

This section provides an overview of the supervised machine learning solutions implemented by the four most successful teams in the competition. The winning team [2] relied on the use of an exhaustive feature engineering process and the design of a group-based recursive feature elimination method to obtain an optimal subset of features. The authors used the LightGBM algorithm as the base model and proposed an ensemble approach to improve the model generalization. The second team [3] also used extensive feature engineering resulting in more than 2000 features. The best features were selected by incorporating the greedy forward search, greedy backward search and SHApley Additive exPlanations (SHAP) analysis. This team is the only leading team considering a hybrid model using a general model and two specific models for each game mode. The general model utilizes all the finalized best features, and the two other models use features specific to each game mode. After experimenting with logistic regression, XGBoost, LightGBM, Catboost, and neural networks, LightGBM was selected for its better performance and lower training time. The third team [4] also relied on an extensive feature engineering process but used the XGBoost algorithm instead of the LightGBM algorithm used by the two best teams. The fourth team [5] focused their efforts on using a semi-automated feature engineering process and ensemble learning. Their best solution used different ensemble averages of boosting models (i.e., XGBoost, LightGBM, and CatBoost) and a mix of levels of data representation.

The Area Under the Curve (AUC) scores of these four most successful teams in predicting victories in the game *Tactical Troops: Anthracite Shift* are 0.8997, 0.8980, 0.8978, and 0.8963. These results are better by over 1% than the baseline model created by the challenge's organizers [6]. Since these top solutions and the baseline solution all used variants of GBDT, the proposed solutions mainly differentiate from one another through their implemented feature engineering and feature selection method. Only the second ranked team [3] provided insights about other tested algorithms, including logistic regression, variants of GBDT, and neural networks. Future work to improve the AUC could explore other commonly known algorithms in winning prediction, such as random forest and DNNs. Additionally, most teams used the truncated data format by aggregating the features present in them over time to develop new aggregated features that were later used in prediction. Instead of using aggregated features over the game time until the prediction moment, the contestants could have used the features from the truncated logs as multivariate sequential input data in predicting the winner. Such sequential data can be processed with DNN like RNN. RNNs are suitable models for classifying sequential data as they are designed to capture dependencies between time steps.

B. Time Series Classification

Predicting victories using sequential data is a task that falls under time series classification (TSC), which is a type of supervised machine learning problem. TSC methods are used in various domains, including health records, human activity recognition, acoustic scene classification, and cyber-security [7]. Numerous algorithms have been developed to classify time series. These algorithms often look for related features in the time series based on distance, interval, frequency, and shapes. DNNs are also used for TSC problems since they are characterized by hidden layers capable of extracting hidden features within time series. In a recent review on DNNs for TSC [7], the authors compared several DNNs against univariate and multivariate time series datasets. Their work showed that Fully Convolutional Networks (FCN) and deep Residual Networks (ResNet) performed better than other tested DNNs algorithms. Therefore, FCN and ResNet represent two models of interest in predicting victories in video games using sequential game data. Note that the authors did not consider RNNs in their comparison due to their limitations, such as computational complexity and learning long-term dependencies.

C. DNNs for Predicting Victories in Video Games

To the best of our knowledge, only one study has considered in-game time series to predict victories. Silva et al. [8] compared three variants of RNN (i.e., simple RNN, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU)) to predict the winner of the multiplayer online battle arena video game League of Legends. The authors generated data representing the state of the game for every minute of a match. Using data from the start of the game until five minutes into the game, Silva et al. achieved an accuracy of 63.91% using a simple RNN. When using the data from 20 to 25 minutes into the game, the authors achieved an accuracy of 83.54%. Since the data sequence fed into the RNN models is relatively short, it was expected that LSTM and GRU, which have been developed to limit the vanishing gradients and capture long-term dependencies, would not provide better performance over a simple RNN. The authors could have compared the results of the simple RNN with other machine learning algorithms that apply to TSC.

Instead of considering in-game data, Qi et al. [9] used post-match data as sequential data to predict the winner of the video game Defense of the Ancients 2. The authors considered a two-stream match-RNN approach. The input features consist of a sequence of features related to previous games played by both teams. Using the data from the last 14 matches played by both teams to feed into LSTM, they obtained an accuracy of 73.68%. Both tested RNN variants (i.e., LSTM and GRU) obtained better accuracy than Support Vector Machines, Convolutional Neural Networks (CNN), and Stacked Autoencoder.

D. DNNs for Sports Prediction

Predicting victories or other outcomes using machine learning is also a common research topic in sports prediction. Two recent papers [10], [11] provide a literature review on machine learning algorithms to predict sports results. Based on these reviews, we observe that the use of deep learning-based sequential models for sports prediction is not a common approach and warrants more investigations. Among the small number of papers that have considered sequential data in sports prediction, the work of Watson et al. [12] provides a good example of how TSC using DNN can be used in sports prediction. Watson et al. considered the sequential spatial-temporal nature of rugby to predict the outcomes of sequences of play. Their work compared the performance of various DNNs, including CNN, three RNN variants (i.e., GRU, LSTM, and bi-directional recurrent units), and a combined CNN-RNN model. In general, CNN-RNN and LSTM outperformed the other models in predicting the outcomes of sequences of play.

Zhang et al. [13] developed an attention-based LSTM model to predict the outcome of upcoming sports matches. The learning model was trained using the football team's historical match data consisting of seven features such as the number of goals, the rate of ball control, and home-court advantage. Their performance evaluation showed that their model effectively predicted the final result (i.e., win, lose, or draw) of the next five matches for a given football team.

E. Summary

Previous studies investigating DNNs using sequential data to predict victories in video games or sports outcomes show that this approach can lead to accurate predictions. The performance of the DNNs highly depends on the task at hand and the datasets. However, it seems that the use of LSTM generally tends to better capture patterns in sequential game data and therefore make better predictions.

III. Background

Based on the literature review and initial experiments, we chose to explore six DNN models for predicting victories using the sequential data created from the game logs. Since Silva et al. [8] obtained good prediction accuracy with a simple RNN, LSTM, and GRU for the video game Leagues of Legends, these three models were implemented. Additionally, we also considered the FCN and ResNet model assessed by Fawaz et al. [7]. Their work showed that these two models developed by Wang et al. [14] generally provide better accuracy for TSC than other DNNs. During the first experiments, we observed that the FCN and ResNet models may be too complex for the task at hand; therefore, we decided to also implement a simplified version of the FCN model. The following paragraphs describe these six DNN models.

A. Simple RNN

A simple RNN was implemented using the *SimpleRNN* layer provided in the Python library Keras. This built-in layer is an RNN where the output from the previous time step is fed into the next time step. This architecture aims to capture the dependencies between time steps. The structure of a simple RNN cell is shown in Fig. 1. Each simple RNN cell has two inputs: x_t, the sequential data at time step *t*, and h_{t-1} which is the state of the previous time step *t-1*. Then, a hyperbolic tangent activation function is applied to the calculation to obtain the current state at time *t* (h_t). This output is then fed into the next RNN cell.

B. LSTM

Since simple RNN models suffer from the vanishing gradient problem which makes model training harder, the LSTM model was developed to overcome this shortcoming. As shown in Fig. 2, the structure of an LSTM cell is more complex than a simple RNN cell. The LSTM cell consists of a cell state (*c*), a hidden state (*h*), an input gate, an output gate, and a forget gate. In addition to using the current input and past hidden state, the LSTM cell includes a cell state which passes information from one cell to another. This information is controlled by the three gates.

C. GRU

The complex structure of the LSTM cell results in a longer training time and may be too complex for the machine learning task at hand. GRU is a simpler version of the LSTM model. The cell structure of a GRU, as shown in Fig. 3, is similar to the internal structure of the LSTM. One main difference is that a GRU cell uses two gates to control the information instead of three: a reset gate and an update gate.

D. FCN

RNN variants, such as simple RNN, LSTM, and GRU, are not the only type of DNNs able to model sequential data. CNNs are commonly known as powerful deep learning models for image classification, but can also handle TSC. A typical CNN architecture consists of several layers, including convolutional, pooling and fully connected layers. Our fourth model is an FCN model, which is a CNN without fully connected layers [15]. Without the fully connected layers, the FCN can be applied to inputs of any size. The FCN architecture used for our project is the one proposed by Wang et al. [14]. As shown in Fig. 4, the FCN model consists of three convolutional blocks. Each block consists of a convolutional layer, a batch normalization (BN) layer and a Rectified Linear Unit (ReLU) activation function. The output of these three blocks is then averaged over the whole time dimension (i.e., global average pooling layer). The three convolutional layers have respectively a filter size of 128, 256, and 128 and a kernel size of 8, 5, and 3 without striding and zero padding.

E. Simplified FCN

The simplified FCN model consists of a single block of the original FCN model. The filter and kernel size of the convolutional layer will be determined through optimization. Additionally, the global average pooling layer of the original FCN model has been replaced by a global maximum pooling to extract the most prominent features.

F. ResNet

The last model considered to predict victories is a ResNet. ResNets were developed to limit the vanishing/exploding gradient problem encountered in very deep neural networks. The particularity of the ResNet architecture is added connections between the input and output of the convolutional layers. These connections reduce the vanishing gradient effect by allowing the gradient to be directly propagated to earlier layers. Fig. 5 provides the ResNet architecture used to predict victories. This model was proposed by Wang et al. [14] and assessed by Fawaz et al. [7] for TSC. This model consists of nine convolutional layers and a global average pooling layer. Note that the last layer of the FCN and ResNet shown in Fig. 4 and Fig. 5 was replaced by a sigmoid function instead of a softmax function to support binary classification.

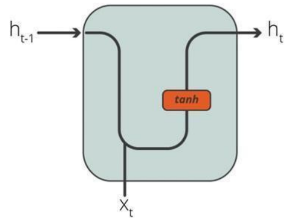

Fig. 1. Simple RNN unit structure [16].

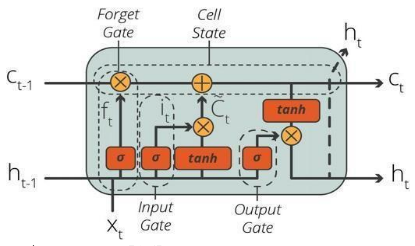

Fig. 2. LSTM unit structure [16].

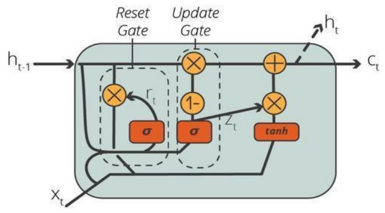

Fig. 3. GRU unit structure [16].

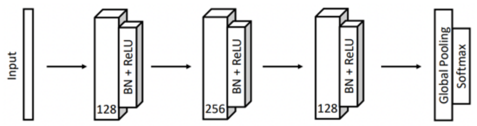

Fig. 4. FCN model architecture [14].

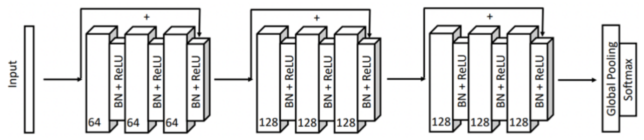

Fig. 5. ResNet model architecture [14].

IV. Methods and Procedures

A. *Configuration of the Environment*

The six models were implemented using Python version 3.7.12. The Python codes for the data preprocessing and models 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 building the models, 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.

B. *Video Game Description*

A brief description of the video game is warranted to understand better the following sections explaining the game data and the data preprocessing techniques. *Tactical Troops: Anthracite Shift* is a 2D turn-based video game in which a player commands a team of four soldiers, also referred to as units. The game can be played as a single-player or against another player. This competition focused on the latter player mode to predict the winner of the game between two players. The game consists of a selection of destruction tools, including 30 weapons and seven gadget types. During a turn, a player can take several actions based on its pool of action points (AP). Types of action, which require a different number of AP, include, for example, shooting a weapon, reloading a weapon, and using a gadget. A player wins if all opponent's soldiers are killed. There are other ways of winning based on the game mode. The challenger's organizers provide more details about the game modes and other aspects of the game in their paper summarizing the competition results [6].

C. *Data Preprocessing*

The raw data 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. 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 using AP) 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 time step 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 time steps in each game varies from 2 to 521. A depiction of game logs modelled as sequential data is provided in Fig. 6.

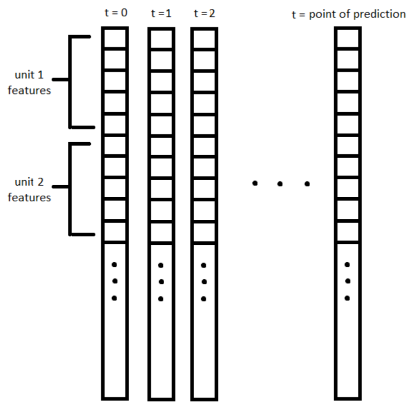

Fig. 6. Time-series data format.

As for feature selection, our goal was to select the most informative features that can be modelled as time-series data and that are common among video games. We also considered the information-gain statistics provided in the work of the first- and second-best teams of the challenge [2], [3] to determine the most informative features for classification. Since both players have four units, there is a total of eight units in each game. For each unit, we selected

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

the following features: AP, health points (HP), weapons (each unit carries two weapons), and the unit type (i.e., assault, heavy, support, scout, and captain). AP correspond to the maximum number of actions a unit can take, whereas HP denote the maximum damage a unit can take.

HP and AP features were normalized using the ratio between their current and maximum values over each game to ensure the values would be in the range of 0 to 1. Data normalization generally leads to more effective learning (i.e., faster convergence) when using DNNs. The remaining features (i.e., unit type and weapons), which are both categorical variables, were initially replaced using label encoding. We later modelled these features using binary encoding to improve the models' performance. Lastly, we randomly shuffled and split the dataset into training and validation sets using an 8:2 split ratio.

D. Models Implementation

The models were built using the sequential API-based Keras model, which allows us to create a model layer-by-layer. Processed data were fed into the models in batches using a generator functionality to reduce memory allocation issues. The six DNN models were trained and optimized on Google Colab to determine the maximum training and validation AUC while minimizing the binary cross-entropy loss as loss function. Google Colab provides Graphical Processing Units and Tensor Processing Units runtime with more memory options leading to faster training time than standard laptops.

Since DNN models require input sequences of the same length and our sequential data for each game consists of a different number of time steps, we had to introduce different techniques to obtain sequences of the same length. For the simple RNN, LSTM and GRU models, we used Keras's padding and masking functionalities. For these models, the first layer is a masking layer which tells the model to ignore the time steps that have been previously padded with a user-defined value. The last layer of these three models is a dense layer with a single neuron (i.e., an output layer with a sigmoid activation function). The FCN and ResNet models were implemented in the same manner as proposed in the work of Fawaz et al. [7].

E. Models Evaluation

The performance measure selected by the challenge's organizers to evaluate the participants' submission is the AUC of the Receiver Operating Characteristics (ROC). This performance measure is used to assess our models and compare them with the baseline and top-performing solutions to the challenge. The AUC is a common performance measure for machine learning tasks involving binary classification. Compared to accuracy, AUC uses probabilities of class prediction instead of the number of correctly predicted samples. Therefore, AUC allows better comparison between models. The ROC is a curve of probability in which the True Positive Rate is plotted against the False Positive Rate at different classification thresholds [17]. An AUC can take values between 0 and 1, where 1 indicates a perfect classifier and 0 suggests that the model is predicting the opposite classes. The latter condition is unlikely to happen and probably signifies improper target labelling.

F. Hyperparameter Optimization

Hyperparameters and model parameters are the two types of parameters that make up machine learning models. The parameters that the user can adjust freely before starting training are referred to as hyperparameters (e.g., the number of neurons in a neural network), whereas model parameters are learned during model training (e.g., weights in a neural network). A low number of neurons in a neural network can lead to underfitting, whereas a high number of neurons can lead to overfitting. The model is unsatisfactory in both circumstances; therefore, hyperparameter optimization is an essential component of every machine learning pipeline.

Neural networks consist of various hyperparameters, such as number of neurons, number of layers, learning rate, momentum, initializers, activation functions, number of epochs, batch size, and optimizers. We need to find the best combination of values of the hyperparameters searching in this multi-dimensional space. This makes hyperparameter tuning a very complex and time-consuming task. There are several ways to perform hyperparameter tuning. We used the Optuna library, which is an open-source hyperparameter optimization framework to automate hyperparameter search [18]. Due to limited computational resources, we only performed hyperparameter optimization on the LSTM and simplified FCN model. The optimized values obtained for the LSTM model were applied to the simple RNN and GRU models. Hyperparameter optimization was not performed for the FCN and ResNet models since both models do not generalize well for our task. The hyperparameters for each model are shown in Table I.

TABLE I
HYPERPARAMETER OPTIMIZATION

Model	Hyperparameter	Optimized value
Simple RNN GRU LSTM	Number of neurons	43
	Dropout	0.1107
	Activation function	Sigmoid
	Weight constraints	1
	Initializer	Glorot uniform
	Optimizer	Adam
	Learning rate	0.0008
	Batch size	188
Simplified FCN	Number of filters	36
	Kernel size	12
	Initializer	He normal
	Optimizer	SGD
	Learning rate	0.0003
	Batch size	32

V. RESULTS

Fig. 7 presents the final AUC score for each DNN model implemented to predict victories. This figure shows that the simplified FCN model achieved the best performance with an AUC score of 0.8447. Appendix A provides more insights into each tested model's training and validation learning curves.

During our initial experiments in which we were investigating the potential benefit of our approach in predicting victories, we first considered only HP in the sequential data. Using this single feature, the simplified FCN model achieved an AUC score of 0.8020. The addition of AP, weapons, and unit type as features and the use of label encoding increased the AUC score by 3%. Finally, replacing label encoding with binary encoding resulted in a 2% increase.

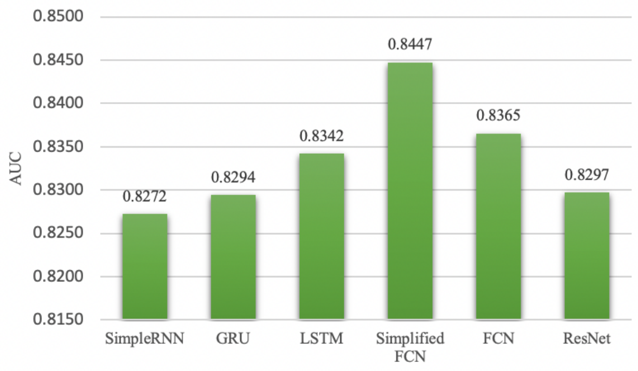

Fig. 7. Model performance.

VI. Discussion

Finding the best DNN models for a specific problem, such as predicting victories using sequential data from game logs, is a time-consuming task that requires extensive knowledge and hands-on experience with deep learning algorithms. In this project, we decided to consider six DNN models with increasing complexity, as shown in Fig. 8, to find the model that would generalize well for our dataset. More complex models can extract more complex patterns in the data. As shown from our results in Appendix A, the FCN and ResNet models, which have the highest complexity among the tested models, were significantly overfitting, indicating that these models were too complex for our dataset. Since the first few epochs of the FCN model provided higher AUC scores than any other models, reducing this model into a simpler architecture (i.e., simplified FCN model) resulted in a good fit model for predicting victories.

Fig. 9 compares our solution with the top-performing solutions of the challenge. Although our model did not surpass the performance of the best challenge models, our proposed approach has significant advantages. First, since we are using DNNs designed to capture hidden features within time series, limited feature engineering and feature selection were required for our project. All top-performing solutions performed very extensive and time-consuming feature engineering and feature selection. For example, the best team's [2] feature engineering and feature selection phase resulted in over 2000 features, while our approach only used 40 features. Additionally, our approach is significantly better in predicting victories than the other winning solutions when only considering HP as an input feature. Our model achieved an AUC score of 0.8020, while a GBDT model obtained an AUC score of 0.72 [5]. This superior performance clearly indicates that modelling the data sequentially is a powerful technique in predicting victories in video games. Another advantage of our approach is that our solution is adaptable to other video games. We used general features, such as HP, unit types, and weapons, which are common in most video games. Therefore, it would be easy to implement our approach to other game logs of other video games using these features.

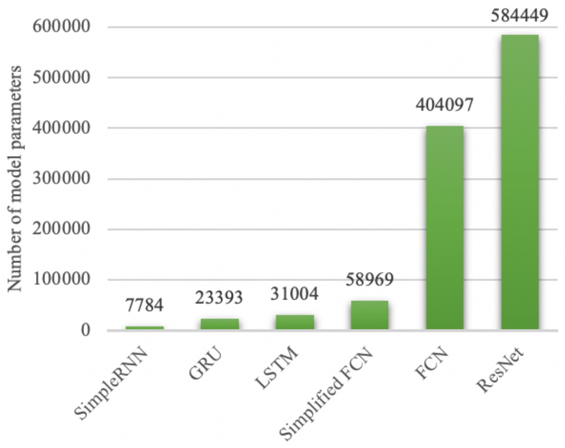

Fig. 8. Model complexity.

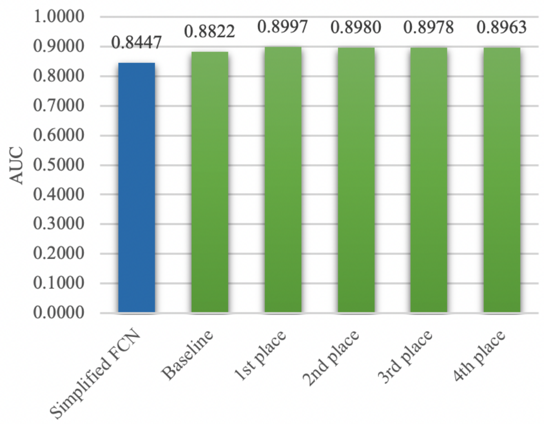

Fig. 9. Model comparison with solutions from the challenge.

VII. Conclusion and Future Work

In this work, we developed an approach to predict victories in the video game *Tactical Troops: Anthracite Shift* using sequential data from game logs. We trained and assessed six DNNs and found that a simplified FCN model achieved the best AUC score among the tested models. The limited need for feature engineering and greater adaptability to other video games are two serious advantages of our approach over the top-performing solutions of the challenge.

With increased computational resources, future work could explore ways to improve our approach. Possible solutions to investigate are as follows:

- Combination of sequential data and fixed data;
- Improved hyperparameter optimization;
- Train other DNNs (e.g., CNN-RNN);
- Consider ensemble models (e.g., GBDT and simplified FCN); and
- Test our generalized approach on other video games.

Appendix A
Training and Validation Learning Curves

This appendix provides the training and validation learning curves for each tested model.

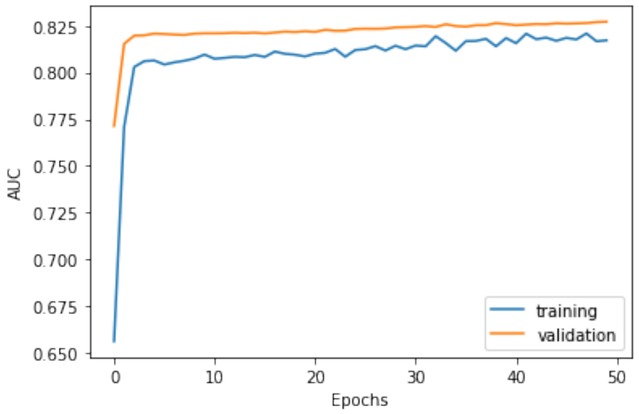

Fig. A1. Simple RNN model training and validation learning curves.

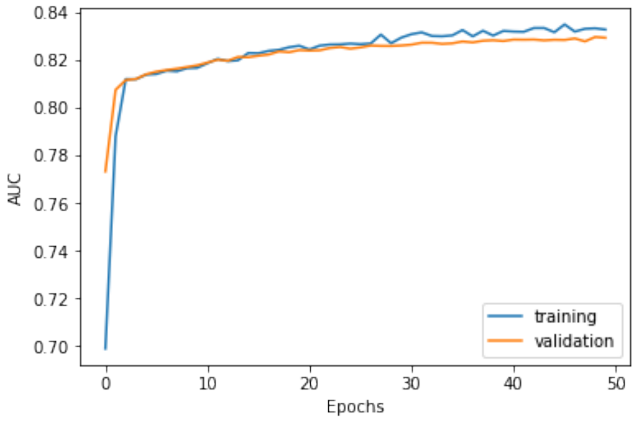

Fig. A2. GRU model training and validation learning curves.

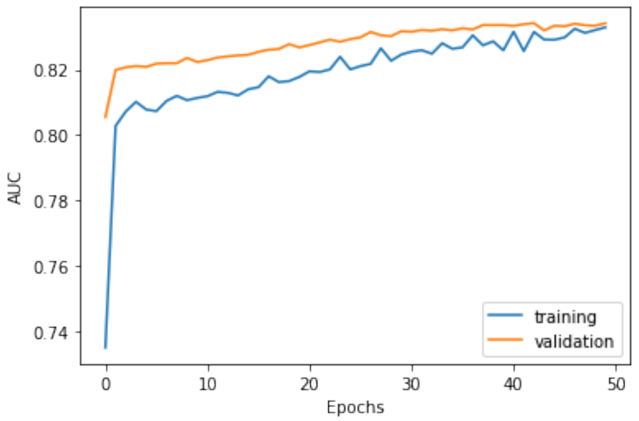

Fig. A3. LSTM model training and validation learning curves.

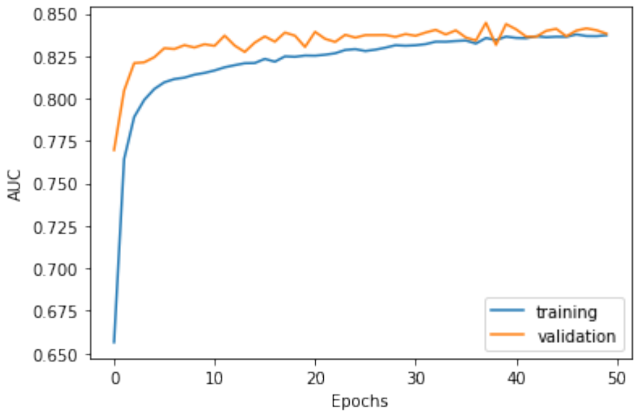

Fig. A4. Simplified FCN model training and validation learning curves.

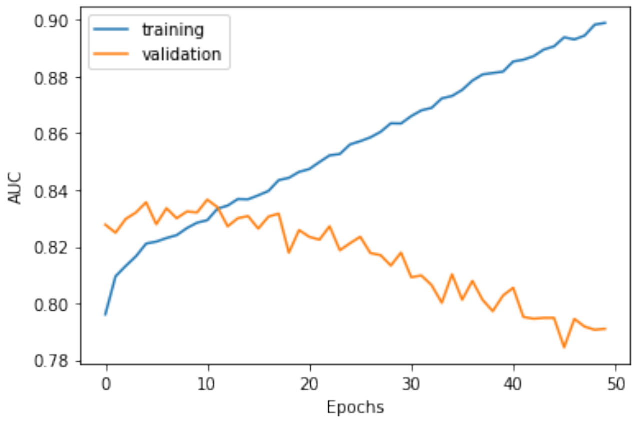

Fig. A5. FCN model training and validation learning curves.

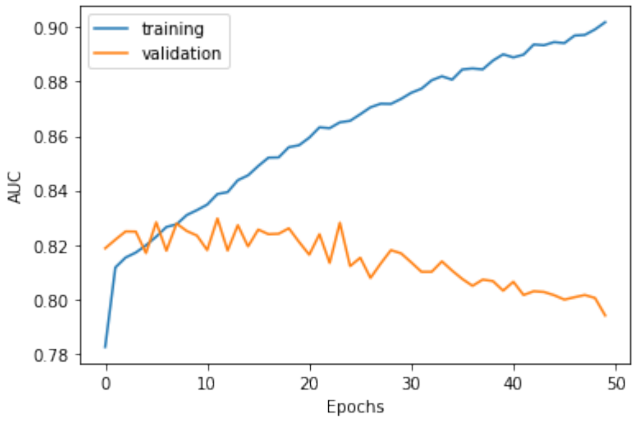

Fig. A6. ResNet model training and validation learning curves.

REFERENCES

[1] IEEE BigData 2021, "BigData Cup Challenges," [Online]. Available: http://bigdataieee.org/BigData2021/BigDataCupChallenges.html. [Accessed: 25-Jan- 2022].

[2] 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.

[3] Q. H. Vu, D. Ruta, L. Cen, and M. Liu, "A combination of general and specific models to predict victories in video games," *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 5683-5690, doi: 10.1109/BigData52589.2021.9671285.

[4] 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.

[5] D. Ruta, L. Cen, M. Liu, and Q. H. Vu, "Automated feature engineering for prediction of victories in online computer games," *2021 IEEE International Conference on Big Data (Big Data)*, 2021, pp. 5672-5678, doi: 10.1109/BigData52589.2021.9671345.

[6] M. Matraszek, A. Janusz, M. Świechowski, and D. Ślęzak, "Predicting victories in video games - IEEE BigData 2021 Cup report," 2021 IEEE International Conference on Big Data (Big Data), 2021, pp. 5664-5671, doi: 10.1109/BigData52589.2021.9671650.

[7] 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.

[8] 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.

[9] 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.

[10] T. Horvat, and J. Job, "The use of machine learning in sport outcome prediction: A review," *Wiley Interdisciplinary Reviews. Data Mining and Knowledge Discovery*, vol. 10, no. 5, Wiley Periodicals, Inc, 2020, doi: 10.1002/widm.1380.

[11] M. K. Langaroudi, and M. Yamaghani, "Sports result prediction based on machine learning and computational intelligence approaches: A survey," *Journal of Advances in Computer Engineering and Technology*, vol. 5, no. 1, Science and Research Branch, Islamic Azad University, 2019, pp. 27-36.

[12] 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.

[13] 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.

[14] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1578-1585, doi: 10.1109/IJCNN.2017.7966039.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[16] O. G. Yalçın, "Applied neural networks with TensorFlow 2 : API oriented deep learning with Python," 1st ed. 2021., Apress, 2021, doi: 10.1007/978-1-4842-6513-0.

[17] J. Kelleher, B. Mac Namee, and A. D'Arcy, *Fundamentals of Machine Learning for Predictive Data Analytics : Algorithms, Worked Examples, and Case Studies.* Second edition., The MIT Press, 2020.

[18] A. Takuya, S. Shotaro, Y. Toshihiko, O. Takeru, and K. Masanori, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference of knowledge discovery & data mining*, 2019 ACM, pp. 2623-2631, doi: 10.1145/3292500.3330701.