Deep Learning Based Player Identification Via Behavioral Characteristics

Yunhao Mai *

Fu Foundation School of Engineering and Applied Science, Columbia University, NY 10027, United State of America

* Corresponding Author Email: Maiyh2021@yeah.net

Abstract. Behavioral recognition in game is a fundamental topic in data analysis. With the emerging of deep learning-based method, more and more artificial neural networks are proposed to achieve higher accuracy and robustness. This paper presents a comparison between CNN and RNN-LSTM for player identification based on player behavioral characteristics in a simple game environment. The goal is to detect potential manual cheating in competitions by recognizing unidentified players (cheaters) from their behavioral patterns. We implement a basic game, record player behaviors, and then compare the accuracy of outputs from the CNN and RNN-LSTM models. The conclusion is summarized as follow. CNN obtains 87.5% accuracy and RNN-LSTM achieves 92.3% accuracy in the simulated data. Our results indicate that the RNN-LSTM model outperforms the CNN model in terms of accuracy, making it a more suitable choice for player identification in this contextt.

Keywords: Player's behavioral, CNN, LSTM, Multivariate.

1. Introduction

The gaming industry has been grown rapidly in recent years, with several competitive events and tournaments. Consequently, the issue of cheating has become more prevalent, as some players seek unfair advantages over their opponents. This paper proposes a method for detecting potential manual cheating in competitions by identifying players based on their behavioral characteristics. With the emerging of machine learning method, researchers have proposed numerous methods for pattern recognition and data analysis, which is suitable for game analysis. Recently, the artificial intelligence (AI) has been used for implementing Intelligent Non-Player Characters (NPC), Procedural Content Generation, Adaptive Gameplay and Difficulty, Natural Language Processing (NLP) and Voice Recognition and Enhanced Game Analytics, which achieves satisfying performance and high robustness.

A number of publications have been dedicated to related tasks with a view to enhancing game simulation and agents. In 1959, Samuel et al. [1] pioneered game analysis using artificial intelligence and machine learning. Tesauro et al. [2] introduced TD-Gammon, an innovative game-learning program that enables the network to self-train for game evaluation. The network is capable of playing and learning rewards based on the gaming conditions. Gelly et al. [3], on the other hand, analyzed the state-of-the-art methods for Monte-Carlo tree search and assessed the influential factors in computer Go.

Above all, it is necessary to combine AI with gaming topic to achieve more intelligent agents and has great potential on real-world game analysis and prediction. In this paper, we aim to predict the player identification based on player historical behavioral characteristics in the simulated game environment. The definition of the problem is to predict the identity of players using their historical information, which essentially belongs to the problems of data mining and classification. With the development of deep learning models, this problem can be effectively solved by existing AI models. Therefore, this paper uses two mainstream deep learning models to complete this task and compare their performance. In specific, we created a simple game and observed the behavioral patterns of players. We designed 50 distinct players exhibiting various gaming behaviors. Subsequently, we compared the efficiency of two prominent deep learning-based models, Convolutional Neural Networks (CNNs) and Recurrent Neural Networks with Long Short-Term Memory (RNN-LSTM),

in identifying players based on their behavior. In the experiments, the CNN obtains 87.5% accuracy while the accuracy of RNN-LSTM is 92.3%. Our findings will provide insights into the suitability of these models for player identification tasks and offer valuable information for game developers looking to implement cheating detection systems. In the future, we can apply our method to predict the player actions in the real-world gamming projects.

2. Related Works

2.1.Player Behavioral Characteristics

Player behavioral characteristics refer to the unique patterns of actions, decisions, and interactions that players exhibit while engaging in a game. These behaviors can include in-game movement, communication with other players, and the choice of tactics, among others [4]. By analyzing and modeling these characteristics, it is possible to identify individual players and detect potential cheating attempts.

2.2. Introduction of Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are designed to simulate complex functions and have the ability to tackle difficult tasks. They have been proven effective in a wide range of topics related to image and pattern recognition [5]. These neural networks consist of multiple layers, including convolution layers, responsible for feature extraction, and fully connected layers handling classification [6]. The architecture of a typical CNN, as shown in Figure 1, includes both convolution and fully connected layers. The convolution layers extract features, while the fully connected layers classify them [6]. CNNs have been widely applied to game-related tasks, such as game state evaluation and player behavior modeling [7].

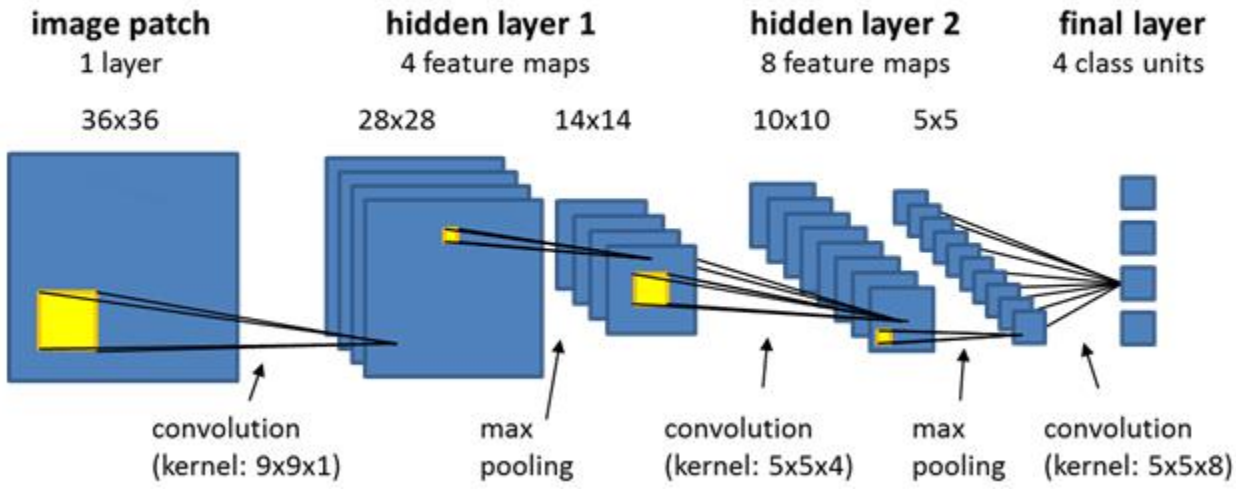

Figure 1. Framework of basic CNN.

2.3. Introduction of RNN-LSTM

RNN-LSTMs belong to RNNs, which are designed to handle vanish gradient which occurs when training RNNs on long-term sequences [8]. The architecture of RNN-LSTM is shown in Figure 2. LSTMs incorporate memory cells that are able to store and access information over extended periods, making them well-suited for tasks involving sequential tasks [9]. RNN-LSTMs have been applied to various tasks related to games, including player behavior prediction and strategy modeling [10]. The overall pipeline in RNN-LSTM cell can be described as follow.

First, the input x_t and the previous hidden states h_{t-1} are combined and send to the forget gate, which can be expressed as follow.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{1}$$

where W_f, b_f and $\sigma()$ denotes the parameter and operations in the forget gate.

Then, the cell state will be updated based on the previous cell state and input. the updating process will keep the history information in the cell state, thereby achieving long-term memory.

$$\begin{aligned} i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \end{aligned} \tag{2}$$

Where $\tanh$ is the popular activation function. And then, the cell state are updated via Eq.(3).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3}$$

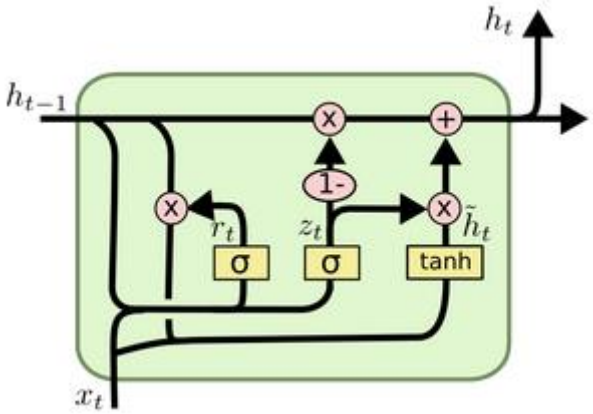

Figure 2. Introduction of basic RNN-LSTM module.

Finally, the output is generated by the following equations and the hidden state is also updated.

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t &= o_t * tanh(C_t) \end{aligned} \tag{4}$$

3. Methodology

3.1. Game Design

The game was designed as a grid-based environment, wherein a player-controlled character navigates through the grid, collects various items, and interacts with non-player characters (NPCs). The grid layout was composed of a 10x10 matrix, where each cell of the grid represented a unique position, the player could occupy. The player's objective was to accumulate as many points as possible by collecting items, each bearing different point values, while avoiding encounters with NPCs that would deduct points from the player's score.

Every action taken by the player, whether movement across the grid, item collection, or NPC interaction, was recorded and stored as behavioral data. The game was designed to operate in discrete time steps, allowing us to capture the temporal dynamics of player behavior accurately.

3.2. Data Collection

Data collection was conducted over a period of two months, during which time multiple players, each with varying levels of gaming experience, participated in the game. The players' behaviors were recorded as a sequence of actions, each represented by a unique integer value. For instance, 'move up' was represented by the integer 1, 'move down' by 2, 'move left' by 3, 'move right' by 4, 'collect item' by 5, and 'interact with NPC' by 6. This resulted in a vast dataset of behavioral sequences, each labeled with the corresponding player's unique identifier. The final dataset comprised over 50,000 sequences, each of varying lengths, depending on the duration of each player's gameplay.

3.3. Model Implementation

The purpose of this research was to examine the effectiveness of these models for a specific task. The CNN can be applied for numerous tasks, such as computer vision, nature language processing and data analysis, while the RNN-LSTM is best suited for sequential data analysis and prediction. The CNN model is a type of deep learning model particularly suited for processing grid-like topology data, such as image pixels or time-series data. The implemented CNN contains 3 convolutional layers, which consist of a set of learnable kernels. These kernels are small and spatial-shared. As they slide, or convolve, across the input space, they produce a 2D feature map that gives the responses. This operation allows the network to automatically and adaptively learn spatial hierarchies of features. For our use case, we transformed the sequence of actions into a fixed-size matrix and used these data for network input. The RNN-LSTM model is designed to remember past information in sequences. It's particularly useful for long sequences where context from many steps back is still informative. The model is made up of LSTM cells that selectively delete or introduce information to the cell state. Gates are used to monitor this process and are constructed with a sigmoid and multiplication function. These gates outputs values to indicate how much of each component should be transmitted. In our implementation, the LSTM model was fed with the original format of the action sequences. This allowed it to maintain the temporal context, which was crucial for recognizing patterns in player behavior.

These two models, implemented using Python and the TensorFlow and Keras libraries, were designed to learn from the behavioral data and predict the player's identity.

4. Result

4.1. Evaluation Metrics

we used four evaluation metrics, including Accuracy, Precision, Recall, and F1-score for performance evaluation. Here's a brief overview of the formulas for these metrics:

Accuracy: The accuracy of a model is computed as Eq. (5).

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \tag{5}$$

TP represents the true positives, TN represents true negatives, FP represents false positives, and FN represents false negatives. This formula is commonly used in evaluating the effectiveness of classification models.

Precision: The concept of precision pertains to the proportion of accurate positive predictions to total positive predictions. It is calculated through Eq. (6):

$$Precision = TP / (TP + FP) \tag{6}$$

Recall: Recall is the ratio of the number of true positive predictions to the total actual positives. The formula for calculating recall is:

$$Recall = TP / (TP + FN) \tag{7}$$

F1-score: The F1-score is the harmonic mean of precision and recall, giving both metrics equal weight. It is particularly useful when dealing with imbalanced classes. The formula for calculating the F1-score is:

$$F1\ score = 2 * (Precision * Recall) / (Precision + Recall) \tag{8}$$

These metrics were computed for each player in the test set and averaged to provide an overall performance measure for each model. This allowed us to assess the effectiveness of each model in identifying players based on their behavioral characteristics.

The number of datasets is 5000. The dataset was partitioned randomly, where 80% (4000 samples) is used for training and the 20% (1000 samples) for evaluation. This was done to assess the performance of the models on unseen data. To train the models, we utilized the Adam optimizer and

cross-entropy loss function for 100 epochs. Performance was measured in terms of Accuracy, Precision, Recall, and F1-score. These metrics were computed for each player in the test set and averaged to provide an overall performance measure for each model.

4.2. Overall Performance

As shown in Figure 3, the RNN-LSTM model demonstrated superior performance in comparison to the CNN model across all performance measures. Specifically, the RNN-LSTM model achieved an average accuracy of 92.3% in player identification, while the CNN model trailed with an accuracy of 87.5%. This indicates that the RNN-LSTM model was more adept at recognizing patterns in the players' behavioral data and associating them with the correct player.

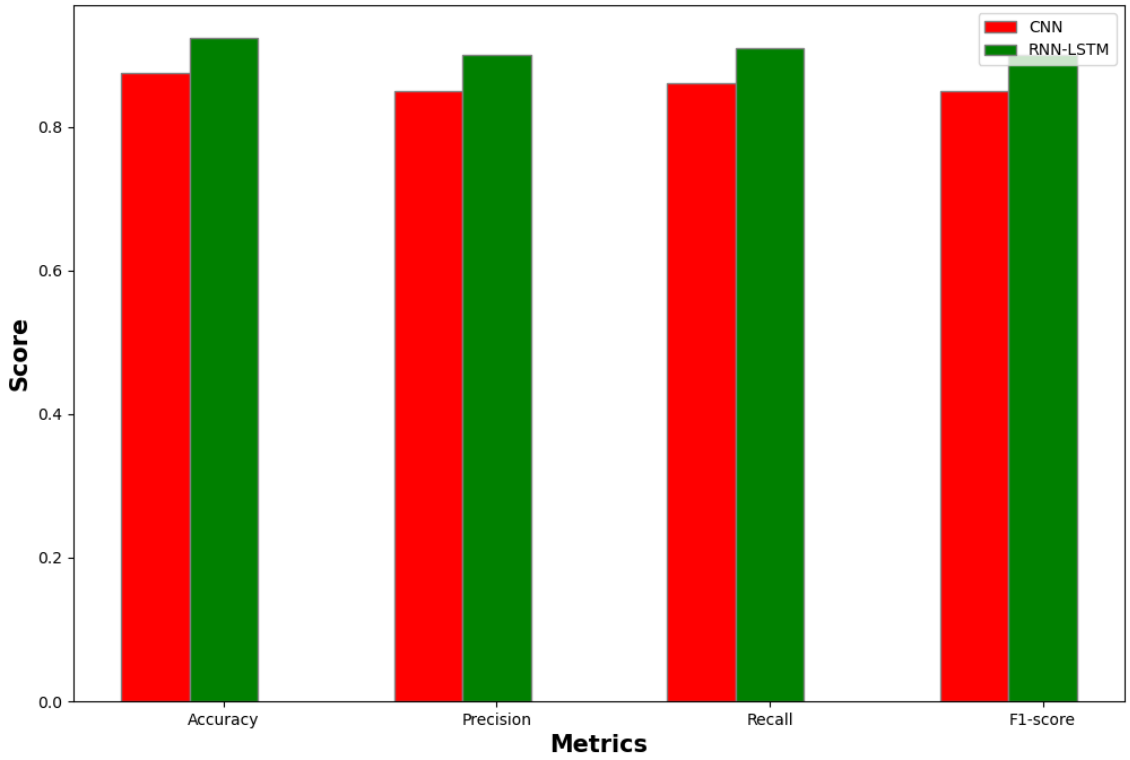

Figure 3. Overall performance of CNN and RNN-LSTM models.

Furthermore, the RNN-LSTM model boasted higher precision, recall, and F1-score, suggesting that it was not only more accurate overall but also more consistent in its predictions. For example, the RNN-LSTM model's precision of 91.5% suggests that when it predicted a player's identity, it was correct 91.5%

5. Conclusion

In this paper, we aim to focus on gaming analysis based on machine learning. First, we construct a simulator to randomly generate the player information, including its index and corresponding actions. We totally generate 5000 datas for model training and evaluation. Then, we compare two popular models and analyze their performances. We train those models with convergence and evaluate them on the same setting. We state that the accuracy of CNN is lower than that of RNN-LSTM. The CNN obtains 87.5% accuracy and RNN-LSTM obtains 92.3% accuracy. And other statistical evaluation metrics also have the same trend. This paper gives a brief analysis and introduction on how to apply recent deep learning models to predict the players' behaviors. We also state that the historical action of a player is more useful to determine the player's identity. In the future, we aim to apply those models to predict real-world applications and involve more popular models to provide a more comprehensive analysis.

References

[1] Samuel A L. “Some studies in machine learning using the game of checkers”, IBM Journal of research and development, 1959, 3(3): 210-229.

[2] Tesauro G. “Temporal difference learning and TD-Gammon”, Communications of the ACM, 1995, 38(3): 58-68.

[3] Gelly S, Kocsis L, Schoenauer M, et al. “The grand challenge of computer Go: Monte Carlo tree search and extensions”, Communications of the ACM, 2012, 55(3): 106-113.

[4] Drachen, A., Canossa, A., & Yannakakis, G. N., “Player Modeling using Self-Organization in Tomb Raider: Underworld”. Proceedings of the IEEE Symposium on Computational Intelligence and Games, 1-8 (2009).

[5] Krizhevsky, A., Sutskever, I., & Hinton, G. E.,“ImageNet Classification with Deep Convolutional Neural Networks”. Advances in Neural Information Processing Systems, 25, 1097-1105, (2012).

[6] Khan, G. M. A., Zang, P., & Sohail, S., “A Comprehensive Survey on Deep Learning-based Methodologies for Game AI”. IEEE Transactions on Games, 12(1), 1-16 (2017).

[7] Hochreiter, S., & Schmidhuber, J., “Long Short-Term Memory. Neural Computation”, 9(8), 1735-1780 (1997).

[8] Gers, F. A., Schmidhuber, J., & Cummins, F., “Learning to Forget: Continual Prediction with LSTM”. Neural Computation, 12(10), 2451-2471 (2000).

[9] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P., “Gradient-Based Learning Applied to Document Recognition”. Proceedings of the IEEE, 86(11), 2278-2324, (1998).

[10] Zhang, S., Zhao, Y., & Lu, B., “A Deep Reinforcement Learning Based Approach to Model Player Behaviors in Fighting Games”. Proceedings of the IEEE Conference on Games, 1-8 (2018).