Proceedings of the 3rd International Conference on Computing Innovation and Applied Physics
DOI: 10.54254/2753-8818/30/20241075

New usage of telemetry for anti-cheating in FPS game

Zilu Wang

School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, The United Kingdom

zw21u22@soton.ac.uk

Abstract. First-person shooter games are experiencing a surge in popularity. As more players join, advanced AI-based cheats have emerged. These cheats simulate human gameplay, sending mouse inputs, making them hard to detect and counter. Therefore, this research presents a novel approach that utilizes telemetry data analysis to identify and counteract cheating in FPS games. The main objective of this study is to develop an innovative anti-cheating system that can effectively detect and prevent players from exploiting AI-based cheats to gain unfair advantages. To achieve this, extensive telemetry data is collected during gameplay. The data contains the real-time cursor position when the player is playing the game. Besides, Machine learning and deep algorithms are applied to analyse the telemetry data and distinguish between human player behaviour and AI-driven cheating patterns. Decision Tree, Random Forest, LSTM, and CNN are applied for this research. And in the final evaluation, CNN's accuracy reached around 80% which proves it is a possible mode to be used for this problem. The significance of this research lies in its contribution against cheating in FPS games, particularly those exploiting AI technologies to gain unfair advantage. The proposed telemetry-based approach offers a solution to safeguard competitive gaming and insight into the game company based on this novel way for further experiments.

Keywords: Telemetry, Anti-cheating System, First-person shooter games, AI-driven cheating

1. Introduction

With the rapid progression of technology, First-person shooter (FPS) games, including notable titles such as CSGO, VALORANT, and Overwatch have seen a significant surge in popularity [1]. These games immerse players in highly realistic virtual environments, delivering unparalleled gaming experiences [2]. However, alongside this popularity, there has been an unwelcome increase in cheating incidents that undermine the fairness and integrity of the gaming community [3]. Cheating not only compromises the experiences of honest players but also presents a formidable challenge to game developers striving to maintain a level playing field.

Currently, anti-cheating software can easily detect and counteract traditional cheats that fetch and modify game data [4]. However, cheaters have evolved, now employing AI and Computer Vision (CV) technologies to evade detection. Current anti-cheating mechanisms, such as VAC, delve into the user's system, raising privacy concerns [5-8]. Statistical-based systems like FairFight, though privacy-compliant, have shown ineffectiveness in some games [9]. The infiltration of CV-based cheats, empowered by sophisticated algorithms like YOLOv5, exacerbates the challenge, rendering traditional anti-cheating systems impotent [10-11].

To address these challenges, this research proposes a novel solution that hinges on telemetry data analysis to detect and counter AI-driven cheating in FPS games (System Overview is shown in Figure 1). This research aims to overcome the limitations of existing anti-cheating mechanisms while safeguarding player privacy and enhancing detection effectiveness. The subsequent sections of this paper elaborate on the proposed system's methodology, delve into telemetry data analysis, and evaluate AI and CV detection algorithms.

In conclusion, mitigating AI-based cheating is crucial to preserving the integrity and enjoyment intrinsic to FPS games [12]. This study is committed to advancing a balanced and equitable gaming environment, leveraging innovative technologies and approaches.

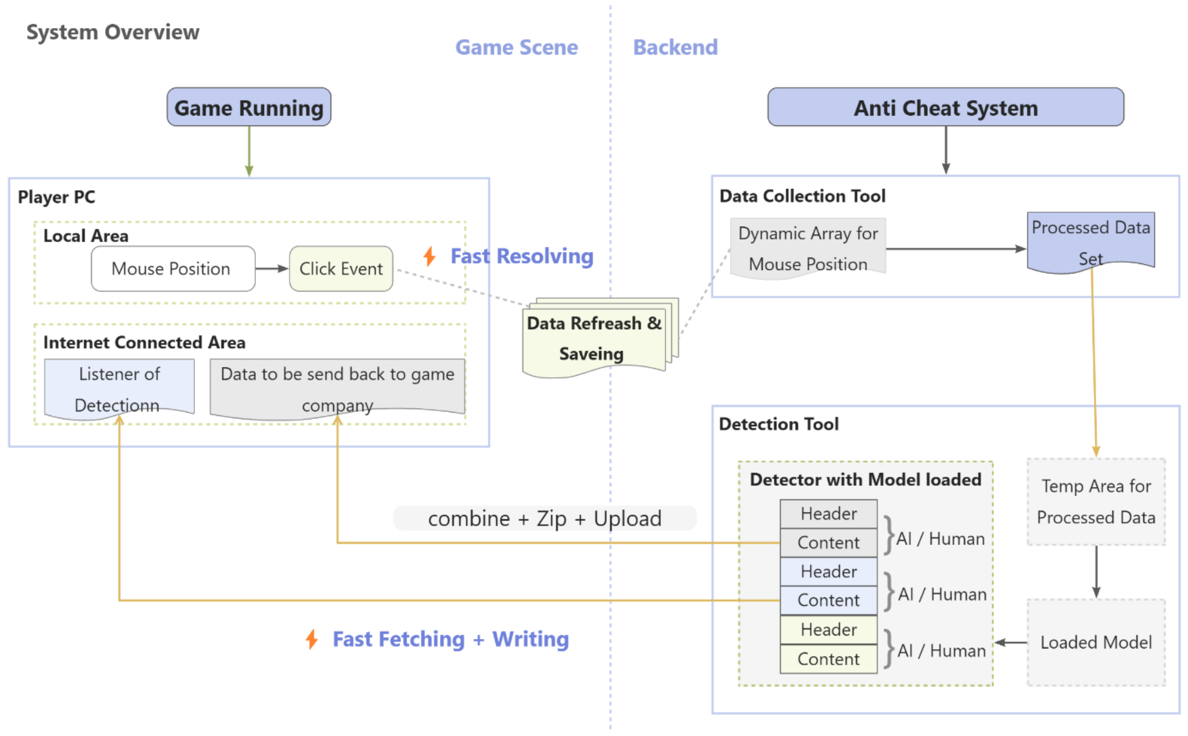

Figure 1. System overview. (Photo/Picture credit: Original)

2. Method

The primary objective of this research is to detect AI-assisted cheating in FPS games using mouse coordinate data. By analysing data and machine learning techniques, this research aims to identify unusual player behaviours and enhance anti-cheating systems.

2.1. Workflow

The flowchart of this research is shown in Figure 2.

1) Data Collection and Pre-processing: The cornerstone of this research is the collection of mouse coordinate data from players during gameplay sessions. Although in other related work, many different kinds of data are collected, this research believes that only collecting mouse coordinates can still offer profound insights into player behaviours and potential cheating mechanisms. Once collected, the data will undergo a thorough cleaning and pre-processing phase to ensure its readiness for subsequent analysis and model training.

2) Data Analysis: The cleaned mouse coordinate data will be subjected to exploratory data analysis techniques. The objective is to identify patterns and trends that might be indicative of AI-assisted cheating. Visualization techniques, especially time-series plots of mouse movements, will be instrumental in understanding the player's behaviours. Besides, After the analysis of these cursor data, several features would be extracted. These features will be the key to running deep learning.

3) Telemetry-Based AI Detection: The telemetry data, in this case, refers to mouse coordinates, which will be the primary input for the anti-cheating systems in this project. AI algorithms (Deep Learning) will be developed to analyze this data in real-time, aiming to detect any anomalies or patterns that suggest cheating.

4) Machine Learning Model Training: Machine learning (Deep Learning) models will be trained using the mouse coordinate data and the features extracted to differentiate between normal gameplay and potential AI-assisted cheating. Feature engineering will be crucial, as the data is limited to mouse coordinates. For this project, Decision Tree, Random Forest, LSTM and CNN are explored to determine which ones are best suited for this specific dataset and problem because they are all proven to be suitable for similar work [13-19].

5) Evaluation: The evaluation phase will focus on determining which models are most suitable for the task and identifying potential models for future work. Using metrics like accuracy, the performance of each model will be assessed. The goal of evaluation in this project is not to find the best model to solve the problem but to find a possible model that is suitable for further research.

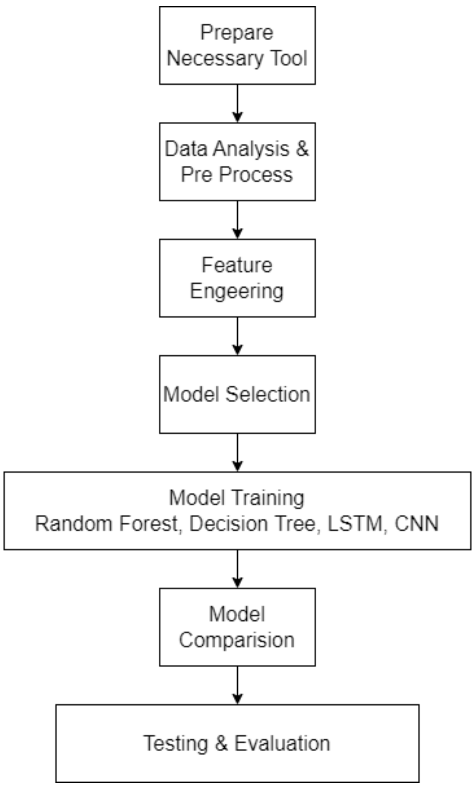

Figure 2. Workflow chart. (Photo/Picture credit: Original)

2.2. Implementation

2.2.1. Data Collection Process: The Mouse Data Collection Tool is integral to this study, designed to adhere to privacy norms, ensuring that it only gathers data from the researcher's gameplay. Engineered to the Windows operating system, the tool is crafted using the C++ programming language, complemented by the Windows API.

As shown in Figure 3, in the initial phase, global variables are established, laying the foundation for the program. These variables encompass a mouse hook, a vector for storing mouse coordinates, a maximum position threshold, a recording status indicator, a flag denoting the data's origin (human or AI), and variables to capture the previous mouse position and button status.

The recording process begins with the start recording function, which operates in a separate thread, clearing existing mouse position data and capturing the current position. It enters a loop to continuously record mouse positions until the 'shouldRecord' variable is deactivated, ensuring the vector size remains within a defined threshold. The end recording function is triggered by setting 'shouldRecord' to false, posting a quit message, and unhooking the mouse hook. The MouseHookProc function handles mouse events, performing operations based on distinct events. The main function, the tool's entry point, creates a thread for the start recording function to accommodate the continuous coordinate recording and conditional data writing. The process continues with the setting of the mouse hook, entering a message loop, and concludes with the unhooking of the mouse hook and the completion of the recording thread.

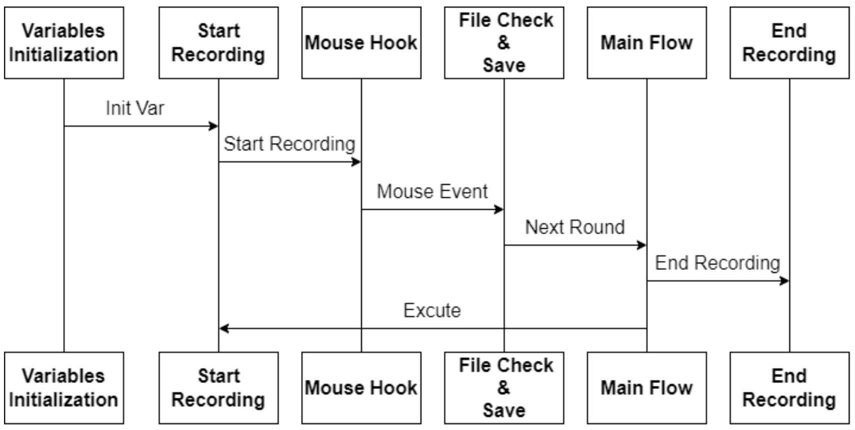

Figure 3. Process of data collection. (Photo/Picture credit: Original)

2.2.2. Feature Engineering: The process of transforming raw mouse movement data into a structured format suitable for machine learning and deep learning applications is executed with precision. As shown in Figure 4, each XML file containing mouse data is loaded and processed. The data, once extracted, is processed to compute a set of distinct features, each offering insights into different aspects of mouse movement dynamics. Subsequently, sklearn's StandardScaler standardises these features to ensure uniformity and consistency, enhancing the efficacy of the learning algorithms. The processed and scaled features are then aggregated into a dataset, facilitated by the custom PyTorch MouseDataset class. This dataset is serialised and stored, ensuring its readiness for model training and evaluation tasks.

Each feature extracted and computed from the raw data plays a pivotal role in characterizing mouse movement patterns. The specifics of each feature are listed below:

1) Coordinates: The X and Y coordinates, extracted directly, represent the spatial positions of the mouse, serving as foundational data points for further computations.

2) Difference with Previous Coordinates: This feature calculates the spatial displacement between consecutive coordinates, offering insights into the direction and magnitude of mouse movements.

3) Moving Speed: Derived by dividing the coordinate differences by a specific time interval (perMsGetCoursor), it quantifies the velocity of mouse movements, shedding light on the user's pace of interaction.

4) Acceleration: Acceleration is computed as the rate of change in speed, providing nuanced insights into the variations in mouse movement speed, indicative of user responsiveness and behaviour dynamics.

5) Back Movement Flag: A binary feature indicating the occurrence of backward mouse movements. It is instrumental in identifying specific patterns and anomalies in user interactions.

6) Label: Extracted from the XML's Flag attribute, it distinguishes data generated by humans from that by AI, serving as the target variable for supervised learning tasks.

Each feature contributes uniquely to the holistic understanding of mouse movement dynamics, and their collective integration into the dataset ensures a comprehensive basis for training robust machine learning and deep learning models.

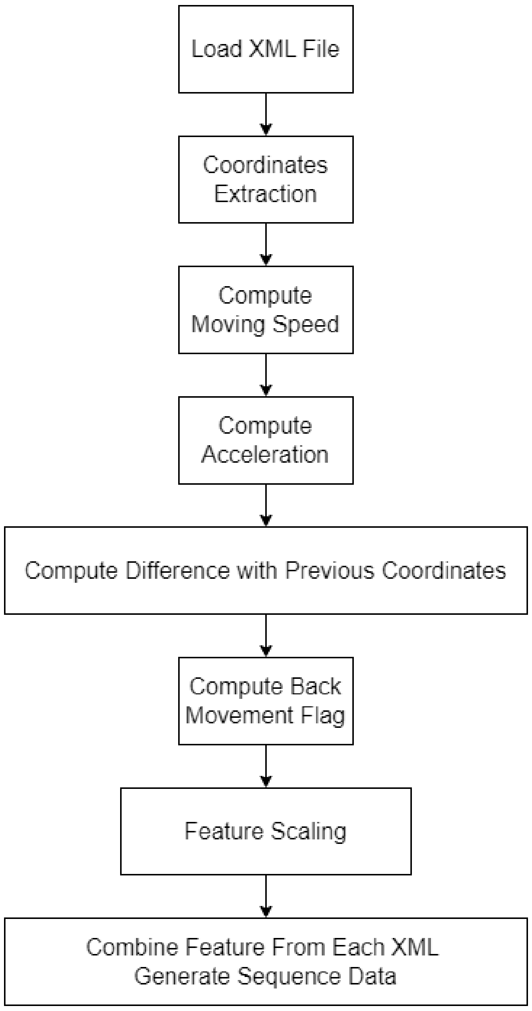
Load XML File
Coordinates Extraction
Compute Moving Speed
Compute Acceleration
Compute Difference with Previous Coordinates
Compute Back Movement Flag
Feature Scaling
Combine Feature From Each XML Generate Sequence Data

Figure 4. Data process flow. (Photo/Picture credit: Original)

2.2.3. Model Training: In the quest to understand mouse movement patterns, a methodical and structured model training was executed, utilizing both deep learning and traditional machine learning techniques. The process was grounded in several extracted features, each playing a vital role in the training and analysis.

1) CNN (Example training process is shown in figure 5 below) :

Input Layer: The foundational data, the X and Y coordinates, were fed into the model, setting the spatial context of mouse interactions.

Convolutional Layers: These layers processed the differences between consecutive coordinates. By examining spatial displacement, the layers discerned direction and magnitude nuances in mouse movements.

Fully Connected Layers: Integrated the more intricate features like moving speed and acceleration. Here, the speed, obtained by parsing the difference in coordinates over a time interval, was processed alongside acceleration, providing the model with insights into variations in mouse movement velocity. These metrics, together, highlighted user behaviour dynamics and responsiveness.

2) LSTM:

The LSTM, given its sequential data processing nature, was particularly effective in comprehending the "Back Movement Flag". Its architecture was attuned to capturing instances of backward mouse movement and understanding anomalies in user interactions [20].

Traditional Machine Learning Techniques:

3) Random Forest:

This ensemble model, with its multitude of decision trees, utilized all the features. Notably, it leaned on the "Label" feature, extracted from the XML's Flag attribute, to distinguish human-generated data from AI, enhancing its classification prowess.

Decision Tree:

The hierarchical structure of this model was adept at processing features like acceleration and the back movement flag. Given the tree's nature, the decision nodes were often rooted in these pivotal features, ensuring a thorough examination of user patterns. In summation, the synthesis of these models, bolstered by the extracted features, furnished a holistic perspective on mouse movement patterns. Each feature was meticulously integrated into the training process, ensuring that every aspect of mouse behaviour was analyzed and understood.

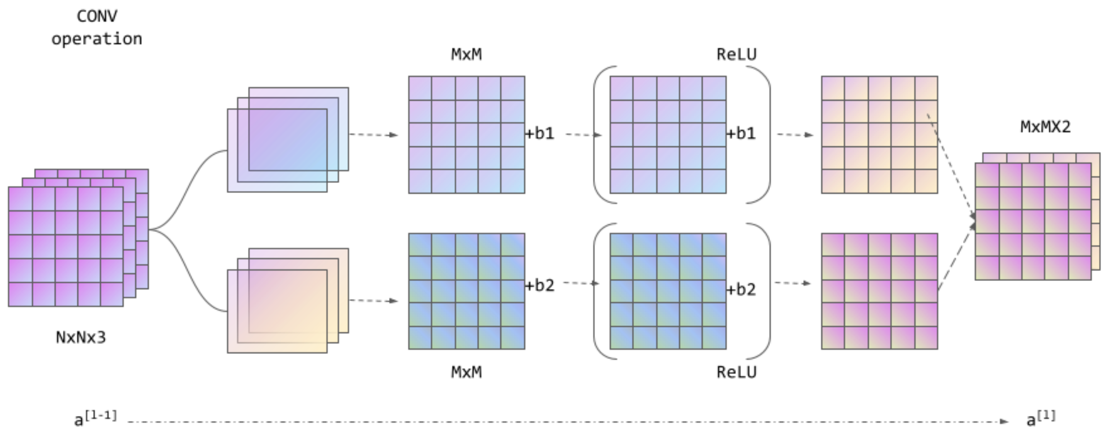

Figure 5. Training example of CNN [21].

3. Result & Evaluation

To deeply analyze mouse movement patterns, several models were rigorously evaluated, namely: Decision Tree, Random Forest, LSTM, and CNN. Their performance was benchmarked against metrics including accuracy, recall, precision, and F1 score.

$$Accuray = \frac{True\ Postive + \ True\ Negative}{Total\ Observation} \tag{1}$$

$$Recall = \frac{True\ Postive}{True\ Postive + False\ Negative} \tag{2}$$

$$\text{Precision} \ = \frac{True\ Postive}{True\ Positive + False\ Positive} \tag{3}$$

$$\text{F1 Score} \ = \frac{Percision * Recall}{Percision + Recall} \tag{4}$$

3.1. Initial Evaluation

Table 1. Model performance with 25 cases.

Model	Accuracy	Recall	Precision	F1 Score
Decision Tree	44%	0%	0%	0%
Random Forest	44%	0%	0%	0%
LSTM	44%	7.14%	50%	12.5%
CNN	100%	100%	100%	100%

As shown in Table 1 above, the CNN model exhibited impeccable performance with an accuracy of 100% in the initial evaluation on a test set of 25 cases. This suggests a robust model capable of discerning intricate patterns in the dataset, with no errors in classification. On the contrary, both Decision Tree and Random Forest models stagnated at 44% accuracy and zeroed out on other metrics. Such uniform predictions, especially by models that are typically versatile, indicate a likely class imbalance or potential inadequacies in feature extraction or representation.

The LSTM's performance was marginally better than the Decision Tree and Random Forest models but was dwarfed by the CNN. It managed an accuracy of 44%, and some level of precision and recall, signifying its capacity to make diverse predictions but still lacking in accuracy.

3.2. *Extended Evaluation*

Table 2. Model performance with 177 cases.

Model	Accuracy	Recall	Precision	F1 Score
Decision Tree	44.07%	0.00%	0.00%	0.00%
Random Forest	44.07%	0.00%	0.00%	0.00%
LSTM	42.37%	1.01%	20.00%	1.92%
CNN	75.14%	86.87%	73.50%	79.63%

For a more comprehensive evaluation, the test cases were extended to 177. As shown in Table 2 above, the CNN model's performance was slightly decreased but still significant at 75.14% accuracy. This dip, while expected with a larger and potentially more diverse test set, still underscores CNN's superiority in handling this specific problem, especially when juxtaposed against the other models.

While the Decision Tree and Random Forest models maintained their performance levels, the LSTM showed slight variations. Although the models of decision trees and random forests maintain a relatively stable accuracy, they are not convincing. This is because they continually assume that all test data is artificial. This validates, in part, that traditional machine learning has no way of distinguishing the information in these mouse coordinates.

3.3. Inner Detail of CNN & LSTM

Model complexity can impact deep learning outcomes [22]. Complexity can enhance pattern recognition but may also introduce overfitting if not matched with sufficient data. This research examined the CNN and LSTM models to understand this relationship.

Table 3. Model parameters and sizes of CNN.

Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 16, 1500]	448
ReLU-2	[-1, 16, 1500]	0
Conv1d-3	[-1, 32, 1500]	1,568
ReLU-4	[-1, 32, 1500]	0
Linear-5	[-1, 16]	528
ReLU-6	[-1, 16]	0
Dropout-7	[-1, 16]	0
Linear-8	[-1, 2]	34
Total params		2578
Trainable params		2578
Non-trainable params		0
Input size (MB)		0.05

Table 3. (continued).

Forward/backward pass size (MB)	1.10
Params size (MB)	0.01
Estimated Total Size (MB)	1.16

Table 4. Model parameters and sizes of LSTM.

Layer (type: depth-idx)	Param #
Total params	36,226
Trainable params	36,226
Non-trainable params	0
Total mult-adds (M)	0
Input size (MB)	0.05
Forward/backward pass size (MB)	0.00
Params size (MB)	0.00
Estimated Total Size (MB)	0.05

As shown in Tables 3 and 4, the LSTM, with 36,226 parameters, achieved an accuracy of 44%. In contrast, the CNN, having 2,578 parameters, reached 100% accuracy in initial tests. These results indicate that performance is not directly correlated with parameter count. The architecture of the CNN, designed for spatial data such as mouse movements, likely contributed to its superior performance. Additionally, the CNN demonstrated faster training speed, even with its larger memory requirement.

4. Discussions

After analysing the problem, feature engineering, model selection and training, testing, and comparing the models, this experiment practically obtained some positive results. Simply analyzing a player's mouse position is somehow enough to determine whether a player is cheating or not. And in the comparison, the CNN model shows a far better performance than other models. Based on the above, the following are the advantages and disadvantages of the experiment, as well as the room for future expansion.

Strengths:

1) Focused Approach: One of the study's major strengths is its concentrated approach to analyzing real-time cursor position data. This provides a clear pathway for detecting AI-driven cheating patterns, streamlining the detection process without being overwhelmed by excessive data types.
2) Comparative Model Analysis: The rigorous evaluation of different models offers a comprehensive view of their performance metrics. Such a comparative study is essential to ascertain the best-fit model for this specific problem.

Limitations:

1) Dataset Limitations: The research is based on data that the researcher collected, which might not fully capture the diverse range of player behaviours, especially when it comes to different AI cheating software [23].
2) Model Specificity: The study's focus on one specific AI cheating software might limit its generalizability. Different AI cheating software might present varied cursor movement patterns, which the models might not detect as effectively.
3) Over-reliance on Cursor Data: While the study's focus on cursor position data is its strength, it might also be its limitation. Other data types, such as keyboard input and in-game statistics, which were used in other studies, could provide a more holistic view of player behaviour.

Future Research Directions:

1) Diversified Data Collection: Future research should consider gathering data from a more diverse player base, encompassing different skill levels, gaming platforms, and multiple AI cheating software [24-25].
2) Real-time Cheat Detection: With the rise of online competitive gaming, real-time cheat detection will be invaluable. Research should explore models that can efficiently operate in real-time without compromising game performance.
3) Collaborative Approach: Collaborating with professional FPS players, game developers, and other stakeholders can offer a more rounded perspective, refining the data collection process and model evaluation.
4) Exploration of Hybrid Models: Instead of relying on a single model, future research could delve into hybrid models that combine the strengths of CNNs, LSTMs, and other algorithms to achieve even better accuracy.
5) Model Training Against Advanced AI Cheats: As AI cheating software evolves, so should the detection models. Training the models against advanced AI cheats will ensure they remain effective in the ever-evolving gaming landscape.

5. Conclusion

This research addressed the emerging challenge of AI-based cheating software in FPS games, which threatens the fairness and integrity of competitive gaming. By analyzing real-time cursor position data during gameplay, the study aimed to differentiate between human and AI-driven cheating patterns. The primary revelation was that solely examining cursor position data effectively distinguishes between these patterns. Contrastingly, other studies have integrated more diverse data types, including keyboard input and in-game statistics. The current model, grounded on data collected by the researcher, might not be wholly representative, necessitating a broader data collection in future endeavours. The study's focus was limited to one AI cheating software, further narrowing its applicability. Future directions include the collection of diversified data, examination of multiple cheating software, real-time cheat detection, and collaborations with both professional FPS players and game developers. Additionally, cross-platform testing is essential given the variance in gameplay characteristics across platforms. This research's significance lies in aiding game companies to understand AI cheating complexities and subsequently design countermeasures. It also underscores the need for a judicious balance between data collection, player privacy, and game performance. Importantly, CNNs were identified as particularly effective models for this problem, consistently outperforming LSTMs. However, this study does not simply conclude that CNNs are necessarily superior to LSTMs in handling this event because, in general, LSTMs have a greater advantage in handling very large data volumes and time-spanning datasets. For this experiment, the dataset taken was not large enough and relatively homogeneous, so the unknown potential of LSTMs may be discovered in subsequent experiments. Although this study has come to very illuminating conclusions, there are still many future research directions that can continue to be tested in depth. For example, training against AI cheating software and AI anti-cheating software, the introduction of larger training datasets, hybrid training of models, etc.

References

[1] Murray J Chesney T and Hoffmann J R 2018 Social Interactions in Virtual Worlds (Cambridge: Cambridge University Press) pp. 13–42.

[2] Professeur 2018 Forsaken Caught Cheating at Extresland; Optic India Disqualified. https://www.hltv.org/news/25118/forsaken-caught-cheating-at-extremesland-optic-india dis-qualified.

[3] Egri-Nagy A and Törmänen A 2020 8th Int. Symp. Comput. Net (CANDAR). (Naha, Japan; IEEE) pp. 9-18.

[4] Sycore 2019 Tom Clancy's The Division 2 Reversal, Structs and Offsets. https://www.unknowncheats.me/forum/tom-clancy-s-the-division/320082-tom-clancys-division-2-reversal-structs-offsets.html?s=f2b918ae67cfc343417d1763e199d8dc.

[5] Brinkmann M 2014 Steam's VAC protection, now scans and transfers your DNS cache. https://www.ghacks.net/2014/02/16/steams-vac-protection-now-scans-ans-transfers-dns-cache/#:.
[6] GabeNewellBellevue 2014 Value, Vac and Trust. https://www.reddit.com/r/gaming/comments/1y70ej/valve_vac_and_trust/.
[7] Hemmingsen M 2021 Sport Ethic.Philos. 15, 435.
[8] Steam 2023 Steam Search. http://store.steampowered.com/search/?category2=8.
[9] Ultralytics 2023 ultralytics yolov5. https://pytorch.org/hub/ultralytics_yolov5/.
[10] Yang W Rifqi M Marsala C and Pinna A 2018 Int. Joint Conf. Neural Networks (IJCNN) (Rio de Janeiro, Brazil; IEEE) pp. 1-8.
[11] Cui Y, Si M and Li Q 2013 2nd Int. Conf. Electri. Eng., Big Data Algo. (EEBDA) (Changchun, China; IEEE) pp. 1012-1017.
[12] Ahn C Grizzard M and Lee S 2021 Frontiers psych. 12 666518.
[13] Sampath A K Gomathi N 2017 J. Cent. South Univ. 24 2862.
[14] Ali J Khan R Ahmad N and Maqsood I 2012 Int. J. Comp. Sci. 9 272.
[15] Ashley 2020 Cheating in FPS Games - Why Do Players Cheat and What Can Be Done? https://www.esports.net/news/cheating-in-fps-games-why-do-players-cheat-and-what-can-be done/:text=A%20recent%20thread%20on%20Reddit%20indicates%20that%20VAC,records%20any%20domain%20name%20look-ups%20on%20your%20machine.
[16] Khalifa S 2016 Machine Learning and Anti-Cheating FPS Games (London: University College London Press).
[17] Kotsiantis S. B 2013 Artif. Intell. Rev. 39 261.
[18] Kacprzyk J 2016 Lecture Notes in Networks and Systems. (Springer International Publishing AG)
[19] Rehman A. U Malik A. K Raza B and Ali W 2019 Multimed. Tools. Appl. 78 26597.
[20] Mohana J Yakkala B Vimalnath S Benson Mansingh P. M, Yuvaraj N, Srihari K, Sasikala G, Mahalakshmi V, Yasir Abdullah R, Sundramurthy V. P and others 2022 J. Healthc. Eng. 2022 1892123.
[21] Saravia E 2021 ML Visuals. https://github.com/dair-ai/ml-visuals.
[22] Saleem M. A Senan N Wahid F Aamir M Samad A and Khan M 2022 Math. Probl. Eng. 2022 7313612.
[23] LeCun Y Bengio Y and Hinton G 2015 Nature 521 436.
[24] Pinto J P Pimenta A and Novais P 2021 Mach. Learn. 110 3037.
[25] Sledevič T Serackis A and Plonis D 2022 Agriculture 12 1849.