BEAT: Behavior Evaluation and Anomaly Tracking, Game Bot Detection Framework in RPG Games

Huixia Cao
Netease Games
Tianhe Qu, Guangzhou Shi, China
caohuixia@corp.netease.com

Yunkun Li
Netease Games
Tianhe Qu, Guangzhou Shi, China
liyunkun@corp.netease.com

Zhaohao Liang
Netease Games
Tianhe Qu, Guangzhou Shi, China
gzlzh@corp.netease.com

ABSTRACT

In the realm of gaming, game bots represent a serious threat to the game balance, resulting in decreased enjoyment for legitimate players. For developers, the infiltration of bot players can lead to substantial loss of normal players, ultimately shortening the game's lifespan and reducing revenue. The proliferation of game bots has significantly impacted the gaming industry's ecosystem. Accurately identifying game bots and combating black market production is critical for ensuring secure game operations. However, the number and variety of RPG game bots has been on the rise in recent years, posing a significant challenge for security operations and making bot detection increasingly difficult. To this end, we propose BEAT: behavior evaluation and anomaly tracking, game bot detection framework in RPG games. It combines the game settlement data, click trajectory and movement trajectory data of players in the game behavior trajectory, generates multi-view data of players from multiple angles, and uses supervised and semi-supervised models to detect game bots. And the corresponding explanation of the model is given. BEAT has been deployed and applied in a number of domestic and foreign RPG games, greatly improved the detection efficiency, also achieved a very significant effect. Finally, our framework realized automatic model update iteration, which can not only detect known game bot types, but also automatically identify some new game bot types.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; **Cluster analysis**; **Classification and regression trees**; **Neural networks**.

KEYWORDS

game behavior trajectory, game bot detection

ACM Reference Format:
Huixia Cao, Yunkun Li, and Zhaohao Liang. 2023. BEAT: Behavior Evaluation and Anomaly Tracking, Game Bot Detection Framework in RPG Games. In *2023 6th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI 2023), December 22–24, 2023, Sanya, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3639631.3639683

ACAI 2023, December 22–24, 2023, Sanya, China

ACM ISBN 979-8-4007-0920-3/23/12…$15.00
https://doi.org/10.1145/3639631.3639683

1 INTRODUCTION

In the current landscape of game and computer industries, the illicit game practices commonly referred to as black industry include game bots, private services, and account theft. Among these, game hacking poses a significant challenge to the game industry and legal field. Game bots are programs that manipulate game data and automate repetitive tasks without the need for rest, which can cause significant harm to the game ecosystem. Game bots have become a major issue in the game industry, and identifying and cracking down on them is crucial for ensuring the safe operation of game businesses.

Game bots offers users unfair advantages in the game, such as accumulating more experience points or gaining more money or valuable items. These bots can perform tasks that normal players cannot, such as fast automatic release of skills or opening multiple game clients at the same time. As a result, game bot consumers can easily gain an unfair advantage and create a huge imbalance in the game ecosystem. Therefore, detecting game bots remains one of the most pressing problems in game security.

In the fields of computer and game technologies, game bots are typically quite varied, and they often infiltrate the entirety of RPG games. Traditional detection methods through game client technology have struggled to detect these bots, presenting a challenge. To address this, we propose the BEAT: behavior evaluation and anomaly tracking, game bot detection framework in RPG games, which utilizes artificial intelligence technology and multi-view data from game behavior trajectories to identify game bots. This includes analyzing behavior sequence data, click trajectory, and movement trajectory data within the game's behavior trajectory. Our BEAT framework provides a comprehensive solution for detecting game bots in RPG games:

- `Click trajectory image view`: we adopt CNN-based anomaly trajectory image classification model for classifying normal player trajectories and abnormal player trajectories.
- `Moving trajectory sequence view`: Unsupervised model is used to cluster coordinate the trajectory data of players moving in the game map, to obtain the cluster sequence generated by data labels, then calculate whether there is a cyclic pattern in the cluster sequence, finally determine whether it is game bot.
- `Game settlement data view`: The classification model and abnormal detection model are respectively adopted to detect the behavior sequence of the game bot, in order to explore and identify the different data patterns of the normal players and these bot.

- `Click trajectory image view`: we adopt CNN-based anomaly trajectory image classification model for classifying normal player trajectories and abnormal player trajectories.
- `Keyboard click sequence view`: We adopt a keyboard click sequence loop model to mine potential repetitive subsequences in keyboard click sequences, and use a small amount of manually labeled samples to assist in finding the model threshold for detecting keyboard scripts or keyboard macros.

The main contributions of BEAT are summarized as follows:

- We detect game bots based on multi-view data on the behavior trajectory of RPG games.
- We used a large number of real data sets in RPG games for detection, and it has been widely used in many RPG games. The impact of game bots attack is very obvious, and the detection efficiency of game bots has also been significantly improved, which has been highly praised by the game operation team.
- Our game bot detection model for trajectory data has the function of automatic update and iteration, which can find more unknown types of game bots and can be easily promoted to RPG games of the same type.
- We innovatively use a trajectory image drawing algorithm to convert click trajectory sequence into trajectory image for further recognition and detection. This approach effectively reduces the dimensionality and amount of data, and thus reduces the computational complexity of the overall algorithm. Furthermore, it improves the interpretability of the algorithm, as trajectory image are more intuitive to human than click trajectory sequence, and the differences between positive and negative samples are more discriminable.

As one of the leading RPG game development companies in China, Netease games has established numerous game operation teams to address the challenges posed by evolving game bots. Despite the efforts made by these teams, detecting RPG bots remains a highly challenging task. Game bots in various RPGs exhibit similarities due to the comparable nature of RPG game system designs. This paper uses "`Diablo: Immortal`" as a case study, focusing on three in-game scenarios that are frequently targeted by game bots. It should be noted that the findings can also be applied to other game scenarios:

- `Rift task`: Due to the requirement of a secret key for players to access rift scenes, players must first complete reward tasks to obtain the key before entering the task scene. As rift scene tasks are relatively simple and repetitive, game bots can repeatedly enter these tasks to gain more experience or equipment.
- `Instance task`: These tasks refer to a series of separate game scenarios where players and their teammates can explore, adventure, or complete quests in a private area without interference from others. Players without teammates are not allowed to enter these private areas. These tasks are popular among players for their high rewards in terms of game experience and items. Consequently, game bots often exploit instance tasks by killing numerous monsters and automatically collecting in-game currency and experience.
- `Field task`: These tasks specifically involve players traversing field maps to defeat wild monsters for the purpose of leveling up, obtaining equipment, or acquiring in-game currency and experience.

2 RELATED WORKS

With the increasing variety of game bots, the methods for detecting game bots are constantly evolving. Researchers have proposed various detection methods, which can be generally categorized into three aspects: client-side, network-side, and server-side [13]. However, client-side detection may negatively impact the player's gaming experience. On the other hand, network-side detection methods based on traffic analysis, as described in [2] and [7], have found that the network traffic generated by human players differs from that generated by bot players. Nevertheless, these methods have a low accuracy rate and are not effective. Therefore, this study focuses on server-based game bots detection methods. In recent years, machine learning and data mining techniques have been widely used to analyze player behavior data generated on the server side to detect game bots. The following section provides a brief overview of the relevant research in this area.

Siu Francis et al. [30] proposed an approach based on a dynamic Bayesian network that only relies on game states and runs in the game server. Chen KT. et al. [3] introduced a trajectory-based approach to detect game bots using a naive Bayesian classifier without kernel density estimation. Stefan et al. [16] analyzed movement data of characters to extract waypoints and detect repeated paths. Luca et al. [5] introduced a novel framework that utilizes supervised learning techniques. In [18], Christian Platzer proposed an approach is based on the combat sequence each avatar produces when engaging an enemy, the difference between subsequent combat sequences is measured in terms of their levenshtein distance [14]. Yeounoh et al. [4] observed game playing behaviors and grouped them based on behavioral similarities using k-means and SVM algorithms. Jehwan et al. [17] constructed new features based on eigenvector centrality to detect game bots through social interactions in MMORPGs. Kang et al. [11] observed the behavioral characteristics of game bots and used four classifiers (decision tree, random forest, logistic regression, and naive Bayes) to detect repetitive tasks associated with gold farming and real money trading. In [15], Liu Daiping et al. introduced an aim-detect, which is a cascaded classifier designed to detect inconsistencies between the performance and skill level of aimbots. The aim-detect classifier aims to identify when the performance of aimbots does not align with the expected skillfulness. In [27], Jianrong Tao et al. proposed NGUARD, a method that combines supervised and unsupervised techniques to capture user patterns and identify game bots from player behavior sequences. The NGUARD method utilizes attention-based approaches to automatically distinguish between game bots and human players. In [29], M. Tsikerdekis et al. utilized time series data and developed a model that utilizes a limited set of features and observation time to ensure scalability and generalization across different domains. The model incorporates Long Short-Term Memory (LSTM) techniques to capture temporal dependencies in the data. In [26], Tao Jianrong et al. introduced a multi-view game cheating detection framework driven by explainable AI (XAI). The framework combines cheating explainers and cheating classifiers from different perspectives

to generate individual, local, and global explanations. These explanations contribute to evidence generation, reason generation, model debugging, and model compression. In [1], Acien Alejandro et al. proposed BeCAPTCHA, a CAPTCHA method that analyzes touchscreen information obtained during a single drag and drop task in combination with accelerometer data. BeCAPTCHA aims to enhance the security of CAPTCHA systems by utilizing additional data from touchscreen interactions. In [19], Qi X. et al. presented GB-GNN, a novel game bot detection model that leverages graph neural networks to model players' trajectories as graph-structured data. This approach enables the capture of complex behavioral patterns in players that are challenging to reveal using traditional sequential methods. In reference [28], the authors collect players' data and extract features to build a multilayer perceptron neural network model. They calculate each player's abnormal rate to identify game bots. Using K-means clustering, they further define a gray area for players with abnormal rates. Reference [12] by Yong et al. focuses on extracting the frequency of action time intervals (ATIs) for each identified action. They use features such as ATI average and ATI standard deviation as inputs for machine learning algorithms. In [23], Su et al. propose a reconstruction-based model called Mutiview GMTVAE. This model utilizes finger touch records collected by screen sensors to identify potential cheating players in a semi-supervised manner. Sha et al. propose a pre-train method called LocationTime2Vec in [31]. This method learns representations of trajectories from a large amount of unlabeled samples, embedding spatial and temporal information hidden in the trajectories. In [22], Yueyang et al. introduce FCDGame, which includes a Hierarchical Trajectory Encoder and a Crosspattern Meta Learner. These components capture the intrinsic characteristics of mobile games and address the issue of limited labeled data, respectively. Kanervisto et al. present a proof-of-concept method called GAN-Aimbot in [10]. They use mouse trajectory data to detect aimbots in online games. Finally, in [20], the paper analyzes the ReMouse dataset using statistical and advanced machine learning methods, including deep and unsupervised neural learning. The results show that different human users cannot generate identical or similar-looking sessions when performing the same or similar online tasks.

3 GAME DATA

The dataset in this paper is mainly taken from the product logs of online RPG games. The labels are mainly marked by game operation and test team. In consideration of player's privacy, we anonymize player's private information.

The following takes "`Diablo: Immortal`" as an example, we respectively extract three scene data about 500,000 of rift task, field task and instance task, the following is a detailed introduction of several data sets used in this paper.

3.1 Click Trajectory Sequence Data

The sequence formed by the coordinates of the player's click on the game interface. The same interface button or icon will be different according to the screen resolution set by the player. We mainly learning the rift task and instance task. Of course, other tasks can be similarly analyzed. Taking one game bot player's click data as example, the sequence as follow: $P_i(x_i, y_i), i \in [1, 2, \ldots, n]$. Click trajectory sequence: $[P_1, P_2, P_3, \ldots, P_n]$.

Also, we can visualize the training data of the click trajectory sequence by picture. The following graph shows the normal and abnormal data, as shown in Fig 1.

3.2 Moving Trajectory Sequence Data

The trajectory sequence formed by the coordinates of the player's movement in the game map. In this paper, we mainly learning the moving trajectory data of players in the field map, and other tasks can be similarly analyzed. For example, a task using a game bot player's movement sequence data in the game map, the sequence as follow: $M_i(x_i, y_i), i \in [1, 2, \ldots, n]$. Moving trajectory sequence: $[M_1, M_2, M_3, \ldots, M_n]$.

Also, we can visualize the training data of the moving trajectory sequence by picture. The following graph shows the normal and abnormal moving trajectory data, as shown in Fig 2.

3.3 Game Settlement Data

The game settlement data when the player exits a certain game task in the end, as shown in the Table 1, it mainly including:

- `Task information`: map id, task level, completion result, cost time, reward.
- `Player information`: character level, vip level, payment information, team members.
- `Details`: number of monsters killed, total damage, health recovery, maximum damage, maximum double hit, actual experience gained, item id, whether to turn in items, etc.

We're focusing on rift task and instance task, the same approach can be applied to other RPG tasks.

3.4 Keyboard Click Sequence Data

Keyboard click sequence data refers to a sequence of keyboard events collected windows over a period of time as a result of a player click the keyboard. We followed the Microsoft Windows application development documentation for data collection. Each keyboard event consists of three fields, as shown in formula (3-1): relative timestamp, virtual key code (vkey), and keyboard message flag. The relative timestamp refers to the time elapsed from the first event to a certain event, in milliseconds (ms). Obviously, the relative timestamp of the first event is 0. The virtual key code (vkey) indicates the specific key that triggers a certain event, for example, 0x11 represents the Ctrl key, and 0x42 represents the B key. The keyboard message flag contains other information about a certain event, including repeat count, scan code, extended key flag, context code, previous key state flag, and transition state flag. The keyboard message flag plays an important role in parsing keyboard click sequence data. A complete keyboard click should consist of three consecutive keyboard states: press, hold (which may last for multiple events), and release. By checking the previous key state flag and transition state flag, we can obtain the accurate state of a keyboard event, thereby summarizing multiple keyboard click events into a complete keyboard click.

$$[[0, 0x1B, 0x00010001], \ldots](3-1)$$

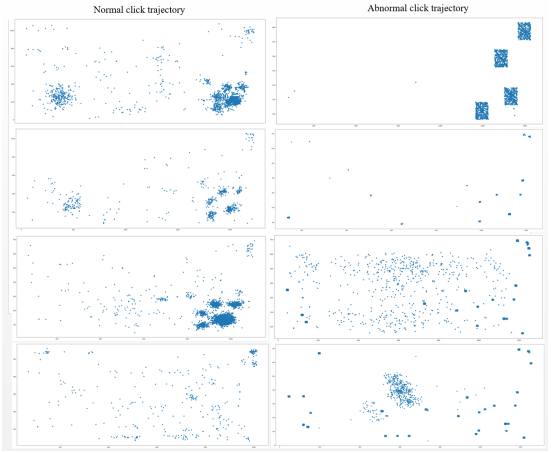

Figure 1: compare with player and bots click trajectory

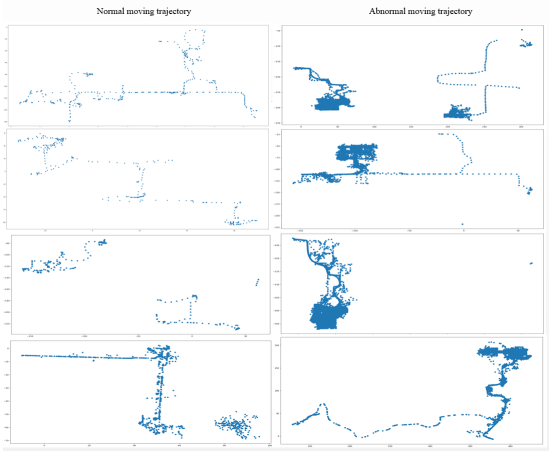

Figure 2: compare with player and bots Moving trajectory

Table 1: Example of Game Behaviors in Diablo Immortal Online

Name	Description	Name	Description
gender	gender of role	exp_actual	actual experience
is_offerings	whether or not player turned in the scarab item	played_time	task time
reward	mission rewards	vip_level	vip level
member_num	number of team	map_id	map id
kill_num	number of monsters killed	zone_id	zone id
damage_dealt	total damage	defense_rating	rate of the defence
return_blood	restore health	score	score of the task
hurt maximum	damage	role_level	role level
exp_potential	potential experience	faction	faction of role

4 FRAMEWORK

This paper proposes a framework BEAT: behavior evaluation and anomaly tracking, game bot detection in RPG games, which mainly includes several modules such as trajectory data generation, model training and application of bot detection as shown in Fig 3.

4.1 Trajectory Data generation

Game trajectory data is mainly extracted from game click trajectory log, movement trajectory log, game settlement log and game keyboard log, etc. Through data preprocessing technology such as segmentation and sampling, corresponding click trajectory sequence and image data, movement trajectory sequence and image data and settlement data are generated according to different game scenes, such as rift, instance and field three task scenes, to provide multi-view data for the game bot detection of RPG games.

4.2 Model Training

For the three task scenarios of rift, instance and field, different data types are trained offline according to the above generated training data, as shown in the Table 2:

Due to the RPG game bot can usually set up similar to the mouse macro function fast automatic release skills, bots can also open multiple game clients at the same time let the bot program automatically play the game, also can open the game recording function repeated, even fixed a few points to patrol the move to automatically gain experience, and so on. Considering the cyclic pattern usually shown by game bots, we design a cycle model. The principle of this model is: if there are sub-sequence which the length of more than 3 repeated cycles for many times(over threshhold ω) in the sequence data every day, the data will be identified as game bot. Let's show the algorithm as follow:

The cycle model is mainly applied in click trajectory sequence data and movement trajectory sequence data, which proved to be very effective in game bot detection applications of RPG games. Then we use the game data and label data trained many supervised and unsupervised model to detect game bot. By the way, we generated train data by game data and the label data which mainly marked by game operation, test team and our cycle model detection result. The following shows the specific trained model:

- `Click trajectory sequence model`: Firstly, we used the click cycle model to judge whether there is a cyclic pattern in the click trajectory sequence. At the same time, we used the button cycle model to judge whether there is a sub-sequence in the cycle pattern according to the clicked button or icon sequence. Then we generated train data and test data by

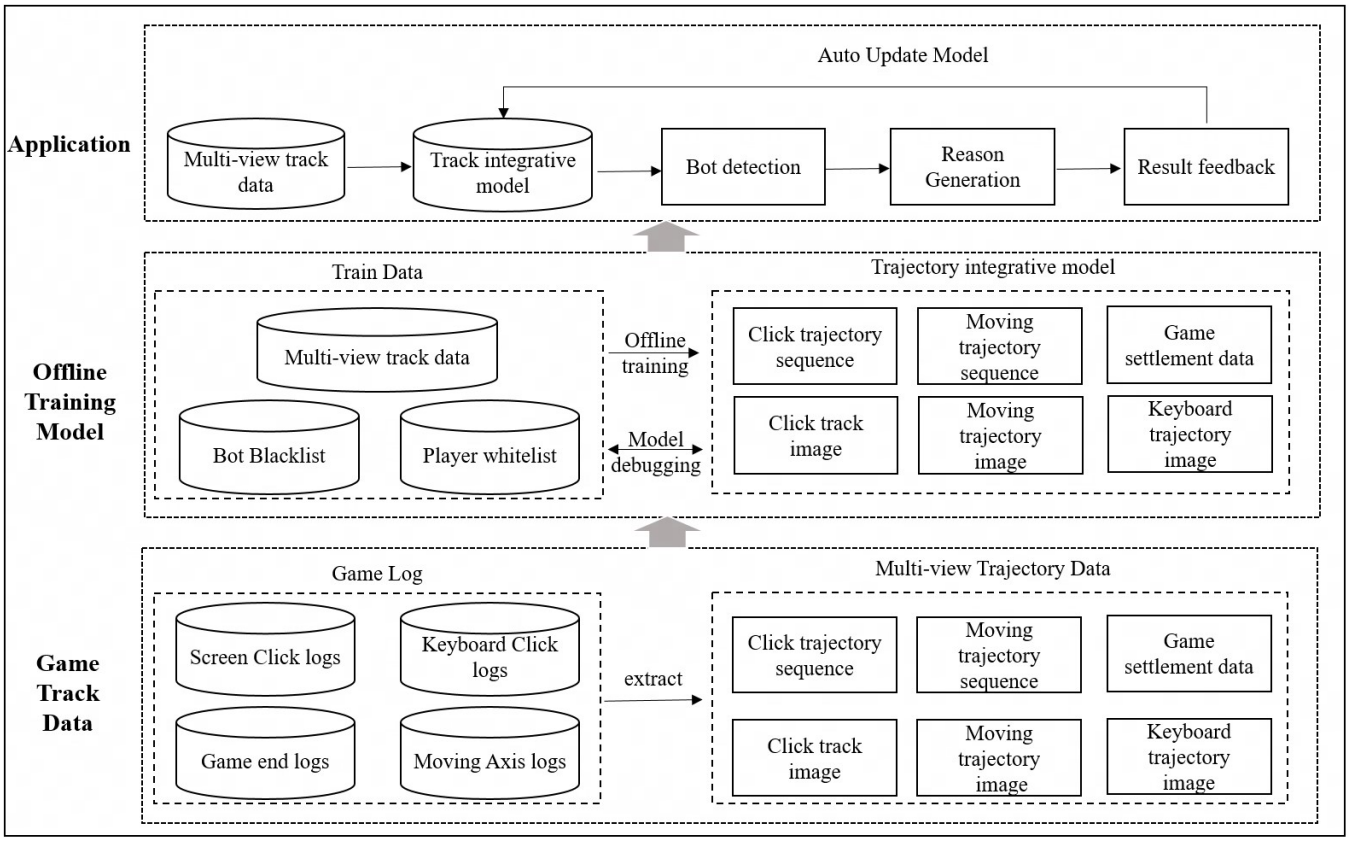

Figure 3: The BEAT Framework

Table 2: Bot Detection Classifier

view	algorithm
Click trajectory sequence data	seq2vec + RNN/CNN/RCNN/transformer + attention + fgm
Moving trajectory sequence data	DBSCAN, Kmeans, Mean shift, repeat model
Game behavior trajectory data	RForest, XGBoost, LightGBM, CatBoost, MLP, Transformer, VAE
Trajectory image	CNN+VAE, ViT
Keyboard trajectory image	RNN, Transfomer

Algorithm 1 Cycle model of click trajectory Sequence

Require: length n,
threshold ω
click trajectory sequence $[P_1, P_2, P_3, \ldots, P_n]$
for $i \geq 0$ and $i \leq n-3$ **do**
for $k \geq i$ and $k \leq n$ **do**
count sequence repetitions: $Count_{[P_i,\ldots,P_k]}$
if $Count_{[P_i,\ldots,P_k]} > \omega$ **then**
$[P_i, \ldots, P_k]$ is repeat
end if
end for
end for
Ensure: $[P_1, P_2, P_3, \ldots, P_n]$ is a repeat sequence

label data and our cycle model detection result. The method of word2vec is adopted to transform the sequence data into seq2vec. Finally, We trained respectively the classification model of CNN/RNN/RCNN/Transformer+GAN for detection of game bot.

- `Moving trajectory sequence model`: In this paper, we used clustering model likely Kmeans, Mean shift, DBSCAN to cluster coordinate trajectory data of players moving in the game map. Then use the cluster sequence to calculate whether there is cyclic mode in the cluster sequence, to determine whether it is a game bot.
- `Game settlement data model`: It refers to the settlement data of players after they quit a certain game task. According

to the number of monsters players killed, total damage, recovery health, maximum damage, maximum double hit data, actual experience acquisition, cost time, payment information, player level and other data, the classification model is trained respectively for two kinds of tasks: rift and instance scenes. For example, RF, xgboost, lightgbm, catboost, MLP, transformer and VAE method to detect the game bot, so as to dig out the normal players and game bots different data patterns.

- `Trajectory image model`: To further explore the potential information in click trajectory sequences, we propose a click trajectory draw algorithm that can draw one or more click trajectory images from a click trajectory sequence, as shown in Fig 4. A click trajectory sequence C is composed of multiple consecutive trajectory point p_i , and each trajectory point p_i contains a timestamp t_i and click coordinates x_i and y_i , as shown in formula (4-1).
$C = [p_1, p_2, \ldots, p_N], p_i = (t_i, x_i, y_i), i \in [1, 2, \ldots, N](4-1)$
In real-world scenarios, due to the diversity of player devices, the resolutions of clients often vary. At different physical resolutions, game clients will adapt UI, art assets, and other elements accordingly. To eliminate the interference caused by resolution differences, we normalize the click coordinates in the click trajectory sequence using the resolution reported by the client before drawing the click trajectory graph, as shown in formula (4-2).
$p'_i = (x'_i, y'_i) = (x_i \frac{R_x^{out}}{R_x^{in}}, y_i \frac{R_y^{out}}{R_y^{in}})(4-2)$
Here, (R_x^{in}, R_y^{in}) represents the resolution reported by the client, while (R_x^{out}, R_y^{out}) represents the size of the click trajectory graph that we define. Clearly, the normalized click trajectory sequence C' can be expressed in the form of formula 4-3:
$C' = [p'_1, p'_2, \ldots, p'_N](4-3)$
We use timestamps to determine the order of the trajectory points and draw them point by point on a blank image with a size of
(R_x^{out}, R_y^{out}). We also draw a line segment between the previous and current points to represent the order information of the trajectory points in pixel space. The output image is shown in Fig 4. The click trajectory drawing algorithm has two benefits: first, it reduces the data volume by one order of magnitude, thus reducing the computational cost of the algorithm and model in subsequent pipelines; second, it transforms sequence data into image data, making it possible to introduce computer vision algorithms and models. In addition, image data is more intuitive for humans, and can be efficiently annotated and compared, while sequence data is more difficult to understand and hard to annotate.
Based on the trajectory image draw algorithm and image similarity measurement algorithm, combined with convolutional neural networks, we have developed a trajectory image model.
- `Keyboard trajectory model`: By analyzing potential patterns in the sequence and calculating the repetition rate of specific patterns throughout the sequence, combined with the periodicity of the patterns, the model comprehensively evaluates the degree of abnormality in the sequence, thus recalling potential abnormal sequences.

4.3 Application of bot detection

Our framework incorporates an automated iterative update mechanism to enhance the accuracy of the model through frequent model updates. Furthermore, our model regularly extracts the most recent game data to serve as training data. For instance, when new game bot data is manually identified or when the bot detection model detects blacklist data, the training samples are updated accordingly. This means that the model undergoes automatic updates and iterations, enabling it to adapt to the ever-changing nature of game bots. As a result, the model's generalization and robustness are significantly improved.

5 EXPERIMENT

In this section, we will use real data set of RPG game to evaluate the effect of our BEAT framework. We will divide the data set according to the training set (80%) and test set (20%), to compare the accuracy, recall rates and F1 score of various models respectively. Details are as follows:

5.1 Click trajectory sequence view

We design a cycle model which mainly to detect whether players click the game screen in a fixed cycle mode to gain game experience, for more accurate detection of click cycle pattern. Such as click cycle model: Calculated the subsequence click trajectory sequence cycles, for example, click on the coordinates sequence repeated 50 times. Button cycle model: click on the corresponding trajectory sequence interface button and icon sequence of subsequence cycles, for example, click on the coordinate transformation for sequence after the interface button or icon repeat 40 times. If it is found that there are sub-sequence loops in the click trajectory sequence more than threshold (ω) times, the player is considered to have used the game bot auto click game to gain experience. The detection principle of the above two models is intuitive and easy to understand, which is suitable for model explanation to the game operation and players. The performance of cycle model are shown in the Table 3.

Finally, according to the test data, cycle model detection results and manual marked label data, training data is generated. The method of word2vec is adopted to convert the serialized data into seq2vec, and the classification model of CNN/RNN/RCNN/Transformer+GAN is trained respectively for detection of game bot. The comparison effect is shown in the Table 4.

5.2 Moving trajectory Sequence view

The data of player's movement trajectory behavior in the game map are sorted according to time to generate the player's moving trajectory sequence data. We used the clustering algorithm to cluster the game user's movement trajectory sequence in the game map to obtain the corresponding cluster sequence of data labels. The label data was made up of the cluster id after data clustering, for example: [1, 2, 3, 4...] such as cluster id.

Detecting the cyclic mode of the cluster sequence obtained after the clustering of the game movement trajectory sequence that is, to

Table 3: Cycle Model Performance Comparison With Click Trajectory Sequence

Rift task				Instance task			
Model	Precision	Recall	F1	Model	Precision	Recall	F1
Click Cycle Model	0.95	0.93	0.94	Click Cycle Model	0.93	0.9	0.92
Button Cycle Model	0.94	0.89	0.92	Button Cycle Model	0.92	0.77	0.84

Table 4: Classification Model Performance Comparison With Click Trajectory Sequence

Rift task				Instance task			
Model	Precision	Recall	F1	Model	Precision	Recall	F1
seq2vec+BiLSTM + attention	0.977	0.979	0.978	seq2vec+BiLSTM + attention	0.998	0.997	0.998
seq2vec+CNN + attention	0.981	0.973	0.977	seq2vec+CNN + attention	0.997	0.998	0.998
seq2vec+BiLSTM+attention+fgm	0.982	0.981	0.981	seq2vec+BiLSTM+attention+fgm	0.998	0.997	0.998
seq2vec+CNN+ attention+fgm	0.977	0.978	0.978	seq2vec+CNN+ attention+fgm	0.998	0.997	0.998
seq2vec+RCNN+fgm	0.978	0.982	0.98	seq2vec+RCNN+fgm	0.999	0.997	0.998
seq2vec+transformer+fgm	0.977	0.98	0.979	seq2vec+transformer+fgm	0.999	0.998	0.998

detect whether the subsequence of the clustering cluster sequence with the length of more than 3 repeats the cycle for several times. When the number of the cyclic pattern exceeds a certain threshold, model can judge whether the user uses the game bot. For example, in the detection of fixed hanging, the rule is that the player moves back and forth between several fixed points. In order to detect this cyclic pattern of moving back and forth, the cluster in the range of fixed points needs to be de-duplicated, so as to calculate whether the player moves back and forth among several clusters. For the continuous identical cluster in the movement trajectory cluster sequence, de-duplicated and normalized are carried out. Therefore, the sub-sequence cycle times of the cluster sequence are calculated again. If the maximum cluster sequence cycle times is found to be more than threshold (ω) times, it is considered that the user has used the fixed point patrol hanged machine and other external hangers.

Taking the RPG game "`Diablo: Immortal`" as an example, this paper obtains the sequence of moving trajectories in the game map and combines it with the clustering algorithm in machine learning to obtain the cluster sequence corresponding to the game sequence data, then detects whether there is a cyclic pattern in the cluster sequence. If the maximum number of cycles of the moving track cluster sequence in map is more than threshold (ω) times per day, which can be considered that the user is very likely to turn on the fixed point patrol hanging up or used other game bot. Examples of detailed bot detection data are as follows: The movement trajectory sequence data of a player using the bot in the game map, $M_i(x_i, y_i), i \in [1, 2, 3, \ldots, n]$ generating movement trajectory sequence: $[M_1, M_2, M_3, \ldots, M_n]$.

The corresponding moving trajectory sequence data is converted into clustering cluster sequence, we can see that in Fig 5.

Finally, we can calculate the cycle pattern of the cluster sequence data. Corresponding cluster sequence cycle pattern, where maximum cluster sequence loops pattern times, we can get the cluster label sequence and count the number of sequence repetitions. Let's show the algorithm as follow:

Algorithm 2 Cluster and Cycle model of Moving trajectory Sequence

Require: length n,
threshold ω
moving trajectory sequence $[M_1, M_2, M_3, \ldots, M_n]$
Ensure: generate cluster label sequence $[L_1, L_2, \ldots, L_n]$
for $i \geq 0$ and $i \leq n - 3$ **do**
for $k \geq i$ and $k \leq n$ **do**
count sequence repetitions: $Count_{[L_i,\ldots,L_k]}$
if $Count_{[L_i,\ldots,L_k]} > \omega$ **then**
$[L_i, \ldots, L_k]$ is repeat
end if
end for
end for
Ensure: $[M_1, M_2, M_3, \ldots, M_n]$ and is a repeat sequence

Table 5: Cluster Model Performance Comparison With Moving Trajectory Sequence

Field task			
Model	Precision	Recall	F1
Mean shift+ cycle model	0.84	0.81	0.83
Kmeans+ cycle model	0.78	0.87	0.82
DBSCAN+ cycle model	0.88	0.82	0.84

The comparison of the effect of the clustering cycle model of the moving trajectory sequence is shown in the Table 5. It can be seen that DBSCAN has a better effect among the effects of the three clustering algorithms.

5.3 Game settlement data view

It refers to the settlement data of players after they quit a certain game task. According to the number of monsters players killed, total damage, recovery health, maximum damage, maximum combo data, actual experience acquisition, task time, payment information,

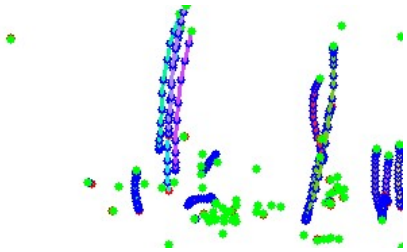

Figure 4: Example of Trajectory Image

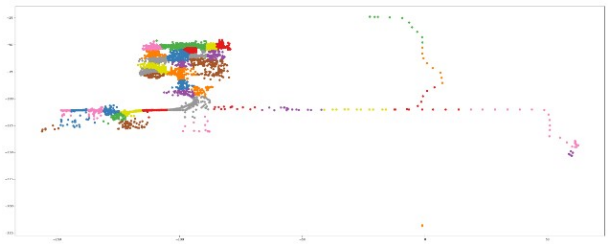

Figure 5: cluster visualization

player level and other data, the classification model is trained respectively for two kinds of tasks: rift and instance task. For example, RF, xgboost, lightgbm, catboost, MLP, transformer and abnormal detection model Isolation forest,VAE method to detect the game bot, at the same time we can dig out the normal players and the different data patterns of the game bot. The model effect comparison is shown in the Table 6 , we can see that the catboost model has better effect.

5.4 Trajectory image view

To validate the feasibility of the abnormal trajectory image classification model based on the trajectory drawing algorithm and convolutional neural networks, we first transformed all the click trajectory sequences in the training set and test set into click trajectory images using the same drawing algorithm, and then manually labeled the training set and test set into multiple abnormal sample classes and one normal sample class. Different abnormal sample classes have different trajectory features and belong to different types of plug-in scripts.

To find the most suitable convolutional neural network framework for the trajectory image, we compared the precision, recall, and F1 scores under different convolutional neural network frameworks, as shown in Table 7, where the input size of each model was fixed at [320, 180].

The MobileNetV3 series and ResNet series models achieved better results. Among them, ResNet-50 and ResNet-101 achieved almost the same result, indicating that for trajectory images, the depth of the ResNet-50 model is sufficient. Considering the effectiveness and computational cost of the model, we ultimately choose to use MobileNetV3-s in production environment.

In order to find the appropriate input size for the trajectory image, we used MobileNetV3-s as the base model and compared the precision, recall, and F1 score under different input sizes, as shown in Table 8.

According to Table 8, the result of the input size ratio of 16:9 is better than that of 1:1. This may be because the resolution of the most players' devices is 16:9, so maintaining the original aspect ratio helps to preserve potential relationships in the image. Considering the effectiveness and computational cost of the model, we choose 320*180 in production environment.

6 APPLICATION

In the fields of game security, it is crucial that we provide players with clear evidence and reasoning behind our game bot detection service. While our game bot detection model is highly accurate, it's equally important to explain the underlying principles of our game bot detection. When players lodge complaints about being banned from logging in due to game operations, we can utilize the principles and results of our model as evidence against cheating players. This approach not only informs them of the reasons for their punishment but also ensures their conviction.

Moreover, our model incorporates an automatic iterative updating mechanism to continuously enhance accuracy and adapt to ever-evolving game bot. Regular updates to the model through resampling enable automatic iteration, resulting in improved generalization and robustness. In case of an incorrect prediction, our model will be promptly updated, thus minimizing false positives and enhancing overall performance. Ultimately, our game bot detection service aims to foster fair and secure gameplay for all users, and we remain committed to refining and enhancing our model to achieve this goal.

7 CONCLUSION

This paper introduces a game bot detection method framework based on multi-view trajectory data in RPG games using the BEAT approach. The framework is evaluated using a dataset from RPG games in the real world. Through numerous experiments, the accuracy classification and interpretation of the framework have been proven rational, yielding interesting and valuable findings. Up to now, we have implemented BEAT in over 30 NetEase games, resulting in a minimum monthly saving of 30 million RMB and achieving outstanding outcomes. Additionally, the implementation and deployment of our framework in RPG games have received highly positive evaluations from the game operator's team.

In the future, in terms of data, we will focus on integrating more types of trajectory data, such as combining voice and text data, in terms of algorithms, we will utilize large language models to optimize game bot detection algorithms, thus further identifying more types of game bot. Meanwhile, we plan to extend the integrated model of multi-view trajectory data to encompass more game types while continuously improving the model's generalization ability and robustness.

ACKNOWLEDGMENTS

Thanks to Netease games operation related team for their help and feedback...

Table 6: Classification Model Performance Comparison with Game Settle Data

Rift task				Instance task			
Model	Precision	Recall	F1	Model	Precision	Recall	F1
RandomForest	0.97	0.99	0.98	RandomForest	0.99	1	0.99
XGBoost	0.97	0.98	0.98	XGBoost	0.98	1	0.98
LightGBM	0.97	0.99	0.98	LightGBM	0.99	1	0.99
CatBoost	0.97	1	0.98	CatBoost	0.99	1	1
MLP	0.92	0.93	0.93	MLP	0.83	0.89	0.86
Transformer	0.97	0.94	0.96	Transformer	0.79	0.89	0.84
VAE	0.97	0.61	0.75	VAE	0.61	0.99	0.75

Table 7: Classification Model Performance Comparison with Trajectory Image

Model	Precision	Recall	F1
VGG-16[21]	0.87	0.91	0.89
GoogleNet[24]	0.88	0.90	0.89
Inception V3[25]	0.90	0.92	0.91
MobileNetV1[9]	0.88	0.90	0.89
MobileNetV3-s[8]	0.93	0.92	0.92
MobileNetV3-l	0.93	0.93	0.93
ResNet-34[6]	0.93	0.91	0.92
ResNet-50	0.94	0.93	0.93
ResNet-101	0.94	0.93	0.93

Table 8: Input Size for the Trajectory Image Performance Comparison

Model	Precision	Recall	F1
224*224	0.92	0.91	0.91
320*320	0.93	0.91	0.92
640*640	0.93	0.92	0.92
320*180	0.93	0.92	0.92
640*320	0.94	0.92	0.93

REFERENCES

[1] Alejandro Acien, Aythami Morales, Julian Fierrez, Ruben Vera-Rodriguez, and Oscar Delgado-Mohatar. 2021. BeCAPTCHA: Behavioral bot detection using touchscreen and mobile sensors benchmarked on HuMIdb. *Engineering Applications of Artificial Intelligence* 98 (2021), 104058.

[2] Kuan-Ta Chen, Jhih-Wei Jiang, Polly Huang, Hao-Hua Chu, Chin-Laung Lei, and Wen-Chin Chen. 2006. Identifying MMORPG bots: A traffic analysis approach. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*. 4–es.

[3] Kuan-Ta Chen, Andrew Liao, Hsing-Kuo Kenneth Pao, and Hao-Hua Chu. 2008. Game bot detection based on avatar trajectory. In *International Conference on Entertainment Computing*. Springer, 94–105.

[4] Yeounoh Chung, Chang-yong Park, Noo-Ri Kim, Hana Cho, Taebok Yoon, Hunjoo Lee, and Jee-Hyong Lee. 2015. A behavior analysis-based game bot detection approach considering various play styles. *arXiv preprint arXiv:1509.02458* (2015).

[5] Luca Galli, Daniele Loiacono, Luigi Cardamone, and Pier Luca Lanzi. 2011. A cheating detection framework for unreal tournament iii: A machine learning approach. In *2011 IEEE Conference on Computational Intelligence and Games (CIG'11)*. IEEE, 266–272.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[7] Sylvain Hilaire, Hyun-chul Kim, and Chong-kwon Kim. 2010. How to deal with bot scum in MMORPGs?. In *2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR 2010)*. IEEE, 1–6.

[8] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*. 1314–1324.

[9] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[10] Anssi Kanervisto, Tomi Kinnunen, and Ville Hautamaki. 2022. GAN-Aimbots: Using Machine Learning for Cheating in First Person Shooters. *IEEE Transactions on Games* (2022).

[11] Ah Reum Kang, Seong Hoon Jeong, Aziz Mohaisen, and Huy Kang Kim. 2016. Multimodal game bot detection using user behavioral characteristics. *SpringerPlus* 5 (2016), 1–19.

[12] Yong Goo Kang and Huy Kang Kim. 2023. Quick and easy game bot detection based on action time interval estimation. *ETRI Journal* 45, 4 (2023), 713–723.

[13] Denis Kotkov, Gaurav Pandey, and Alexander Semenov. 2018. Gaming bot detection: A systematic literature review. In *Computational Data and Social Networks: 7th International Conference, CSoNet 2018, Shanghai, China, December 18–20, 2018, Proceedings 7*. Springer, 247–258.

[14] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. Soviet Union, 707–710.

[15] Daiping Liu, Xing Gao, Mingwei Zhang, Haining Wang, and Angelos Stavrou. 2017. Detecting passive cheats in online games via performance-skillfulness inconsistency. In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 615–626.

[16] Stefan Mitterhofer, Christopher Kruegel, Engin Kirda, and Christian Platzer. 2009. Server-side bot detection in massively multiplayer online games. *IEEE Security & Privacy* 7, 3 (2009), 29–36.

[17] Jehwan Oh, Zoheb Hassan Borbora, Dhruv Sharma, and Jaideep Srivastava. 2013. Bot detection based on social interactions in MMORPGs. In *2013 International Conference on Social Computing*. IEEE, 536–543.

[18] Christian Platzer. 2011. Sequence-based bot detection in massive multiplayer online games. In *2011 8th International Conference on Information, Communications & Signal Processing*. IEEE, 1–5.

[19] Xianyang Qi, Jiashu Pu, Shiwei Zhao, Runze Wu, and Jianrong Tao. 2022. A GNN-Enhanced Game Bot Detection Model for MMORPGs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 316–327.

[20] Shadi Sadeghpour and Natalija Vlajic. 2023. ReMouse Dataset: On the Efficacy of Measuring the Similarity of Human-Generated Trajectories for the Detection of Session-Replay Bots. *Journal of Cybersecurity and Privacy* 3, 1 (2023), 95–117.

[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[22] Yueyang Su, Di Yao, Xiaokai Chu, Wenbin Li, Jingping Bi, Shiwei Zhao, Runze Wu, Shize Zhang, Jianrong Tao, and Hao Deng. 2022. Few-shot Learning for Trajectory-based Mobile Game Cheating Detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3941–3949.

[23] Yueyang Su, Di Yao, Jingwei Li, Baoli Wang, Jingping Bi, Shiwei Zhao, Runze Wu, Jianrong Tao, and Hao Deng. 2022. Trajectory-Based Mobile Game Bots Detection with Gaussian Mixture Model. In *International Conference on Artificial Neural Networks*. Springer, 456–468.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[25] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2818–2826.

[26] Jianrong Tao, Yu Xiong, Shiwei Zhao, Yuhong Xu, Jianshi Lin, Runze Wu, and Changjie Fan. 2020. Xai-driven explainable multi-view game cheating detection. In *2020 IEEE Conference on Games (CoG)*. IEEE, 144–151.

[27] Jianrong Tao, Jiarong Xu, Linxia Gong, Yifu Li, Changjie Fan, and Zhou Zhao. 2018. NGUARD: A game bot detection framework for NetEase MMORPGs. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 811–820.

[28] Woei-Jiunn Tsaur, Chinyang Henry Tseng, Chin-Ling Chen, et al. 2022. Effective Bots' Detection for Online Smartphone Game Using Multilayer Perceptron Neural Networks. *Security and Communication Networks* 2022 (2022).

[29] Michail Tsikerdekis, Sean Barret, Raleigh Hansen, Matthew Klein, Josh Orritt, and Jason Whitmore. 2020. Efficient deep learning bot detection in games using time windows and long short-term memory (lstm). *IEEE Access* 8 (2020), 195763–195771.

[30] Siu Fung Yeung, John Lui, Jiangchuan Liu, and Jeff Yan. 2006. Detecting cheaters for multiplayer games: theory, design and implementation. (2006).

[31] Sha Zhao, Junwei Fang, Shiwei Zhao, Runze Wu, Jianrong Tao, Shijian Li, and Gang Pan. 2022. T-Detector: A Trajectory based Pre-trained Model for Game Bot Detection in MMORPGs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 992–1003.