XAI-Driven Explainable Multi-view Game Cheating Detection

Jianrong Tao(✉), Yu Xiong, Shiwei Zhao, Yuhong Xu, Jianshi Lin, Runze Wu, Changjie Fan
Fuxi AI Lab, NetEase Games, Hangzhou, China
{hztaojianrong,xiongyu1,zhaoshiwei,xuyuhong,linjianshi,wurunze1,fanchangjie}@corp.netease.com

***Abstract*—Online gaming is one of the most successful applications having a large number of players interacting in an online persistent virtual world through the Internet. However, some cheating players gain improper advantages over normal players by using illegal automated plugins which has brought huge harm to game health and player enjoyment. Game industries have been devoting much efforts on cheating detection with multi-view data sources and achieved great accuracy improvements by applying artificial intelligence (AI) techniques. However, generating explanations for cheating detection from multiple views still remains a challenging task. To respond to the different purposes of explainability in AI models from different audience profiles, we propose the EMGCD, the first explainable multi-view game cheating detection framework driven by explainable AI (XAI). It combines cheating explainers to cheating classifiers from different views to generate individual, local and global explanations which contributes to the evidence generation, reason generation, model debugging and model compression. The EMGCD has been implemented and deployed in multiple game productions in NetEase Games, achieving remarkable and trustworthy performance. Our framework can also easily generalize to other types of related tasks in online games, such as explainable recommender systems, explainable churn prediction, etc.**

***Index Terms*—explainable artificial intelligence, cheating detection, online game, industrial application**

I. Introduction

Online games are ubiquitous on modern gaming platforms, including PCs, consoles and mobile devices, and span many genres, including first-person shooters (FPS) [1], strategy games and massively multiplayer online role-playing games (MMORPG) [2], etc. Online gaming represents one of the largest and fastest growing Internet business sectors in the world. It has made the biggest advances by a big margin with progressively more and more players from a variety of ages, nationalities, and occupations getting involved with gaming online.

Despite the great success of online gaming, the game industries have suffered serious threats from game cheating [3]. Game bots in MMORPG are automated programs that reach the system kernel and perform continuously tough or tedious tasks without requiring the rest periods that human players require. Accordingly, game bots can easily achieve great superiority over honest players, leading to the huge imbalance in the in-game ecosystem. The perspective plug-in is common in FPS, which adds several plug-in elements to interface game. Players can get huge information invisible in the game and take advantage of this to win the game. It causes unfair competition and harms the honest players.

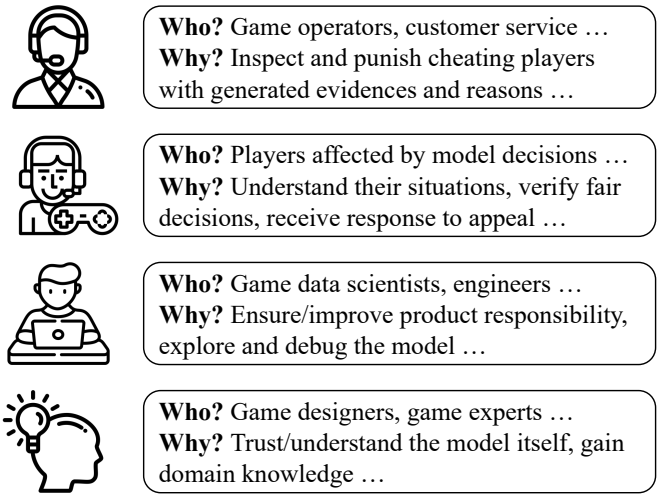


Fig. 1: Diagram showing the different purposes of explainability for game cheating detection sought by different audiences.

Game cheating detection remains one of the most urgent problems that game publishers need to address. Over the last decade, empowering games with artificial intelligence has become a trend which unlocks the data modalities (tabular data like character portraits, sequential data like behavior sequences, image data like client screenshots, graph data like social graphs, etc.) and enables multi-view solutions for cheating detection.

- **Portrait view.** [4] used a time series technique with a MLP neural network model to discriminate human users from game bots based on a set of discriminating portrait features: some features are related to the player and other are related to the game.
- **Behavior view.** [5] employed a combination of supervised and unsupervised methods to detect game bots based on user behavior sequences, considering the different behavioral patterns between human players and game bots. An auto-iteration mechanism was also designed for the concept drift.
- **Graph view.** [6] examined in-game transactions to reveal real money trading (RMT) by constructing a social graph of virtual goods exchanges in an online game and inferred RMT transactions by comparing the RMT transactions crawled from an out-game market.
- **Image view.** [7] developed a real-time automated wallhack

978-1-7281-4533-4/20/$31.00 ©2020 IEEE

detection system using Convolutional Neural Networks. By using Class Activation Maps, the network finds suspicious areas within a screenshot that improves the credibility of the model's performance and makes debugging datasets much more efficient.
- **Multi-view.** [8] proposed a multi-view attention networks (MVAN) for real money trading detection in online games, which utilizes the strong expressiveness of portrait view, behavior view and graph view data.

However, it is far from enough that AI models reach high accuracy for game cheating detection. As mentioned in Figure 1, different audience profiles in online games show great demands for the explainability in AI models for cheating detection.

- **Game operators and customer service** pay more attention on inspecting and punishing cheating players while asking the question that could we give evidences or reasons that this player is cheating or normal?
- **Players affected by model decision** may appeal to customer service to understand their situations of being punished and verify fair decision by receiving response with generated individual model explanations.
- **Game data scientists and engineers** are responsible for designing and delivering AI models and leverage AI explanations to ensure and improve product responsibility. They also simplify model development by exploring and debugging models with local and global model explanations.
- **Game designers and game experts** can discover domain knowledge and new functionalities by understanding and trusting the model itself. More mechanism and intervention can be designed and released to prevent the cheating.

Considering these highlighted above, the usefulness and fairness of AI for game cheating detection will be gated by our ability to understand, explain and control them. The field of XAI has seen great potentiality since we desire the ability to explain black-box models in understandable terms to a human.

The contributions of our study are three-fold:

1) To the best of our knowledge, this is the first work that introduces the explainable multi-view game cheating detection which applies XAI for cheating detection in online games from different views.
2) We propose an **e**xplainable **m**ulti-view **g**ame **c**heating **d**etection framework (EMGCD)[1] which combines cheating explainers to cheating classifiers from portrait, behavior, image and graph views to generate individual, local and global explanations which contributes to the evidence & reason generation and model debugging & compression.
3) We evaluate explainable multi-view game cheating detection with real-world datasets. Extensive experiments show the accuracy of classification and the rationality of explanation. Numerous interesting and valuable findings have been discovered and presented. Our work has been implemented and deployed in NetEase Games and received very positive reviews from the game operator teams.

[1]Data and code available here: https://fuxiailab.github.io/EMGCD/

The rest of the paper is organized as follows. We give a comprehensive description of game data in Section II. In Section III, we present the proposed EMGCD in detail. Section IV presents the experimental results and Section V presents the applications in practice.

II. Game Data

A. Logs in Online Games

Various activities performed by the players or state change events of the clients are recorded in the form of structured game logs through the servers. The logs consist of the following information:

- Timestamp: when a specific activity or event occurs.
- Log ID: an ID that identifies the type of player activity (hunting monster, trading virtual money, client screenshot, etc.) or the state change event type (level up, virtual money increasing or decreasing, mouse trajectory moving, etc.).
- Source player's information: the player's role ID, gender, level, class, virtual money, knead face time and VIP, etc.
- Target player's information: the target player's information if the player interacts with another player.
- Detailed information: depending on each activity or event type, detailed information related to the activity or event.

We collect the game logs from Justice Online[2] and Knives Out Plus[3], which are popular gaming products released by NetEase Games[4]. Justice Online is one of the most successful MMORPGs in China which created a breathing virtual world. Knives Out Plus is a next-generation FPS which truly restored the natural and built environment and incorporated more Chinese elements in design. We have collected more than 400 billion different types of game logs from Justice Online and Knives Out Plus which contain more than 60 million character creation activities. The cheating players are identified and labeled by NetEase Games' operator teams. Considering the privacy, the characters in our datasets are ensured by anonymizing all personal identifiable information.

B. Character Portraits Construction

We build a universal game character portrait system from real-time game logs which covers multiple different topics including account information, device information, basis information, social information, task information, gameplay information, item information, money information, etc. We extract 523-dimensional portraits from the character portrait system of Justice Online. Examples and descriptions of character portraits can be found in Table I.

C. Behavior Sequences Construction

Each player's behavior sequence is composed of lists of events ordered by time stamp which contain four features as followed:

- Timestamp: when the specific event of the player occurs.

[2]https://n.163.com/
[3]https://hyp.163.com/
[4]http://game.163.com/

- Event ID: the current event conducted by a player, for example, using a certain skill, obtaining a certain item, etc. Examples and descriptions of player behaviors can be found in Table II.
- Interval: the time interval in seconds that has passed between the last and the current game event of a player.
- Level: the current game level for the players. The lowest level of each player is 1 and the highest is raised regularly.

TABLE I: Examples of character portraits in Justice Online.

Portrait Name	Portrait Description
physical_memory_size	the size of computer physical memory
watch_movie_acm_pct_avg	the cumulative average percentage of storyline watching
2w_d_avg_team_chat_cnt	the daily team chats in the last two weeks
2w_d_avg_world_send_msg_cnt	the daily world channel chats in the last two weeks
2w_d_avg_use_expression_cnt	the daily average expression usage in the past two weeks
2w_task_giveup_rto	the proportion of abandoned tasks in the last two weeks
2w_bl_task_rto	the proportion of branch tasks in the last two weeks
1m_d_avg_pvp_time_rto	the daily average pvp duration percentage in the last month
guild_fund	the current guild funding
45level_f_bl_task_acm_num	the total number of branch tasks completed before level 45
display_memory_localized_size	the size of computer display memory
2w_equip_play_time_rto	the proportion of equipment play duration in the last two weeks
2w_create_team_rto	the proportion of teams created in the last two weeks
nie_lian_time	the time spent pinching face for the first time
2w_equip_score_upgrade	the equipment score upgraded in the last two weeks
2w_sjtx_time_rto	the proportion of time spent in the SJTX task in the last two weeks

TABLE II: Examples of player behaviors in Justice Online.

Behavior Name	Behavior Description	Behavior Name	Behavior Description
AntiAddiction	anti-addiction	UseMoney	use money
Exp	gain experience	RefreshRanklistNormally	refresh rank list normally
RpcCheat	cheat by RPC	FinishLoadingScene	record the time of scene loading
AntiCheaterAcceleration	cheat by acceleration	GmSetForbidKey	punished by gm setting
GetMoney	get money	LogSkillIntervalInfo	record the skill interval
VipLogin	login as VIP	BGDSJ_RES	finish the BGDSJ task
FashionClothesInfo	put on fashion clothes	TiLi	use the energy
Chat	send a message	OnCheatMaxSpeed	cheat to max speed

Figure 2 visualizes the behavior sequences of a cheating player and a normal player in Justice Online, which gives us a general idea of how behavior sequence looks alike. Each slot represents an event, and different events are assigned different colors to differentiate them. There are a total of 85,751 different events in Justice Online and the average length of player behavior sequences in Justice Online is 13,220. We sample behavior sequences with a length of 200 and keep the events with a frequency of top 800.

(a) A cheating player in Justice Online

(b) A normal player in Justice Online

Fig. 2: Visualization of the behavior sequence examples. The behavior sequence of normal players shows more diverse than that of cheating players.

D. Client Images Construction

For the game cheating detection task in Knives Out Plus, we intercept the game client's picture at the moment the player shoots and kills the opponent, so that we could determine whether the player has used a perspective plug-in based on whether there are non-game elements on the screenshot. To protect the privacy of players, we only take screenshots of the resources in the game. The game client image example of Knives Out Plus is shown in Figure 3.

Fig. 3: Visualization of the client image examples.

E. Social Graphs Construction

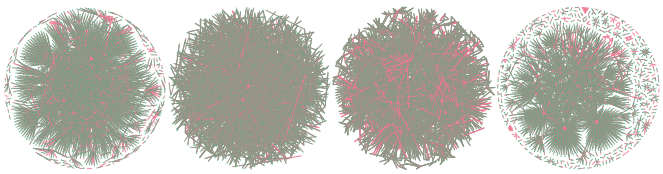
We construct four different types of graphs from the game logs of Justice Online which include a transaction graph, a friendship graph, a teaming graph and a chatting graph. They are visualized in Figure 4 and build up a multi-relational graph. Detailed social graph comparison can be seen in Table III.

The transaction graph shows assets exchange relations between players in the virtual world. Edges indicate the virtual currency of established transactions between players.

The friendship graph is built upon unidirectional friendship in online games. A player can send an invitation to another player and remove friends from his friendship lists.

The teaming graph is made up of collaborative relations between players. A team is temporally formed with the same goal and is disbanded after achieving the goal.

The chatting graph expresses the communication relationship between players. A character can send a private message to other players for individual communications.



(a) Transaction (b) Friendship (c) Teaming (d) Chatting

Fig. 4: Visualization of the social graph examples.

TABLE III: Social graph comparison in Justice Online.

Graph	‖V‖	‖E‖	‖D‖	directionality	description
transaction	14,778	126,283	8.5453	directed	assets exchange
friendship	26,202	141,303	5.3928	directed	friendship
teaming	27,032	694,202	51.3615	undirected	teamwork
chatting	14,136	143,763	10.17	directed	message delivery

III. THE EMGCD

We present the **e**xplainable **m**ulti-view **g**ame **c**heating **d**etection framework (EMGCD) shown in Figure 5 which contains three core modules: game data construction, cheating classifiers and explainers, applications for audiences.

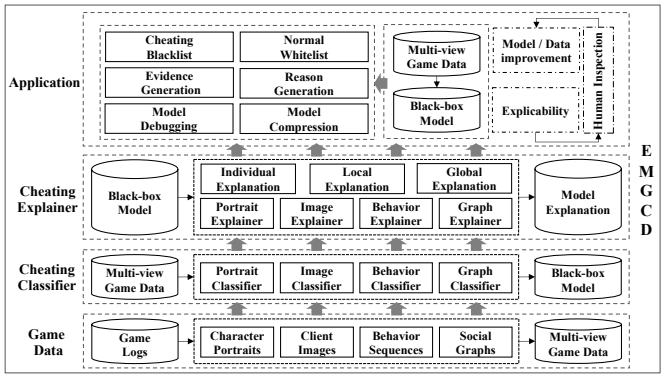


Fig. 5: The proposed EMGCD framework.

Game data construction. As shown in Section II, game data could be extracted from game logs and represented in many different views e.g., tabular (character portraits), sequential (behavior sequences), visual (client images), or graphical (social graphs).

Cheating classifiers and explainers. The algorithm we employ for classification and explanation might depend on the use-case, model type, data format, etc. We select the post-hoc explaining for a given classifier that started with a black box model and prob into it with a companion model to create explanations. Table IV shows the classifiers and explainers in our EMGCD framework. In practice, we appreciate the explanations of these bold ones in front of the candidates.

TABLE IV: Cheating classifiers and explainers.

View	Method
Classification	
Portrait	RF [9], XGBoost [10], LightGBM [11], CatBoost [12], etc.
Behavior	CNN, LSTM, BLSTM, ABLSTM, Transformer [13], etc.
Image	AlexNet [14], VGGNet [15], GoogleNet [16], ResNet [17], DenseNet [18], etc.
Graph	GCN [19], GAT [20], etc.
Explanation	
Portrait	**TreeSHAP** [21], Anchors [22], LIME [23], inTrees [24], LORE [25], etc.
Behavior	**DeepExplainer** [26], NLPExplainer [27], NLPLIME [23], etc.
Image	**GradientExplainer** [28], CAM [29], Grad-CAM [30], etc.
Graph	**GNNExplainer** [31], GraphLIME [32], etc.

The explainers attribute a tabular model's prediction to its features, attribute a sequential model's prediction to individual events, attribute a visual model's prediction to its pixels and attribute a graphical model's prediction to its subgraphs. What's more, one explanation does not fit all and here we propose three types of explanations.

- **Individual explanation** shows the explanations associated with individual predictions (i.e. what was it about the features of this particular player that made her be punished).
- **Local explanation** summarizes the local groupness from individual explanations. We generally sample 2000 samples and perform supervised clustering [21] over these samples and find large number of very explanatory groups/clusters.
- **Global explanation** shows the entire predictive model to the user to help them understand it (e.g. global feature importance, whether obtained directly or in a post hoc manner).

Applications for audiences. The black box model continues to provide the accurate prediction while explanations help improving human interactions/inspections. Explanation needs vary depending on the type of the user who needs it and also the problem at hand. We demonstrate several novel and practical applications below which are successfully applied.

- **Evidence and reason generation** for the game operator teams and customer service teams by exposing attributions. They observe our generated individual explanation and judge whether it is sufficient to form an evidence or form a causal reason for game cheating. When a player makes a banned appeal, they can show the model explanations as evidences of cheating.
- **Model debugging** is helpful to many different audiences. Data scientists/engineers can apply the model's explanations to discover spurious correlations and label leaks and make timely adjustments to the model's data or structure. As for mis-predictions, they could attribute a model misclassification to the features which are responsible for it.
- **Model compression.** Feature attribution values are commonly used to identify the most important features used in model prediction. We can employ these most important features to perform model compression from the data and structural level. For example, we use only the most considerable portraits to train the tabular model, and only the most important behaviors or fragments to train the sequential model, and crop the unimportant image pixel areas to train the visual model. By modifying training samples or features, we use less data and smaller model structures to train better models.

IV. EXPERIMENTS

A. Experimental Settings

We partition the datasets and obtain the non-overlapped training set (80%) and testing set (20%) respectively. All the models are trained on the training set, and we execute the grid search strategy to locate the best parameters on validations by 5-fold cross validation. We measure the performances on the testing set with AUC (area under the curve) and ACC (accuracy) for game cheating detection. The training time (for all the training set), inference time (for all the testing set) and explaining time (for one sample on average) are also recorded for the computing efficiency comparison. We explain all the models in different views and show many interesting findings discovered from the explanations of the models with the best performance.

B. Performance Comparison

Table V shows the performance comparison of the game cheating detection among different competing methods. XGBoost achieves the highest AUC and ACC among all the methods in portrait view. Transformer is optimal considering AUC and ACC results among all the methods in behavior view. GoogleNet outperforms other methods for game cheating detection in the image view. Trade_GCN transcends other methods for game cheating detection in the graph view. Table V also demonstrates the significant training time, inference time and explaining time which indicates that problems that were previously intractable for exact computation are now inexpensive.

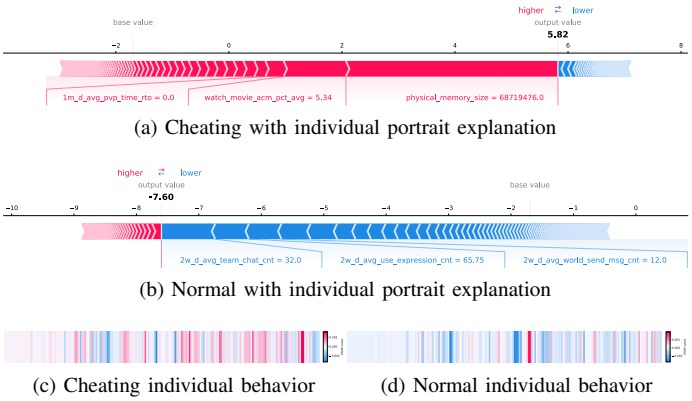
TABLE V: Game cheating detection performance comparison among different methods.

Algorithm	AUC	ACC	Time_tra	Time_inf	Time_exp
Portrait View					
RF	0.9764	0.9516	31s	0.2408s	0.0591s
XGBoost	**0.9876**	**0.9724**	**9m38s**	**0.0004s**	**0.1745s**
LightGBM	0.9859	0.9659	53m10s	1.0286s	0.0226s
CatBoost	0.9834	0.9674	23m21s	0.1652s	0.0034s
Behavior View					
CNN	0.9667	0.9561	1h32m40s	1m20s	23s
LSTM	0.9389	0.9376	4h10m56s	2m42s	33s
BLSTM	0.9506	0.9418	7h30m27s	3m24s	1m5s
ABLSTM	0.9661	0.9507	8h11m52s	4m24s	1m54s
Transformer	**0.9836**	**0.9694**	**10h53m17s**	**6m9s**	**2m35s**
Image View					
AlexNet	0.5259	0.6727	1h36m53s	3m27s	15s
VGG	0.5030	0.7605	55m43s	1m33s	2m39s
GoogleNet	**0.9644**	**0.9147**	**1h12m17s**	**5m10s**	**1m2s**
ResNet	0.9351	0.8940	42m25s	1m26s	3m6s
DenseNet	0.9268	0.9065	1h25m40s	10m19s	5m54s
Graph View					
Trade_GCN	**0.9832**	**0.9795**	**2h50m3s**	**48s**	**2m53s**
Team_GCN	0.9759	0.9715	2h44m22s	53s	3m12s
Chat_GCN	0.9742	0.9584	2h58m39s	49s	2m44s
Friend_GCN	0.9808	0.9713	2h48m11s	48s	3m37s
Trade_GAT	0.9669	0.9537	6h3m36s	45s	3m37s
Team_GAT	0.9576	0.9516	6h57m47s	42s	4m2s
Chat_GAT	0.9581	0.9495	6h5m47s	46s	3m47s
Friend_GAT	0.9598	0.9525	5h59m6s	46s	3m55s

C. Portrait Explanations

Individual portrait explanations. The cheating player in Figure 6a has a large computer physical memory size, and it is very likely that the cheating studio is opening a good deal of accounts in batches. At the same time, his average percentage of storyline watching is very small since cheating players are not as interested in the storyline as normal players. They are more concerned about how to profit from the game at a low cost that leads to his small proportion average daily PVP duration in the past month. The most obvious evidence for the normal player in Figure 6b is that he has more chats. His average number of chats on the team, the number of emojis used, and that of comments on the world channel within two weeks are higher.

Local portrait explanations. Four clusters are found for game cheating detection in Figure 7a. For the players in the cheating cluster 1, their average percentage of storyline watching are all 0. For those in normal cluster 2, they use more silver in the game. In another cheating cluster 3, the players have larger computer physical memory size, and the average percentage of storyline watching is relatively low. At last, players in normal cluster 4 conduct more chats.



(a) Cheating with individual portrait explanation

(b) Normal with individual portrait explanation

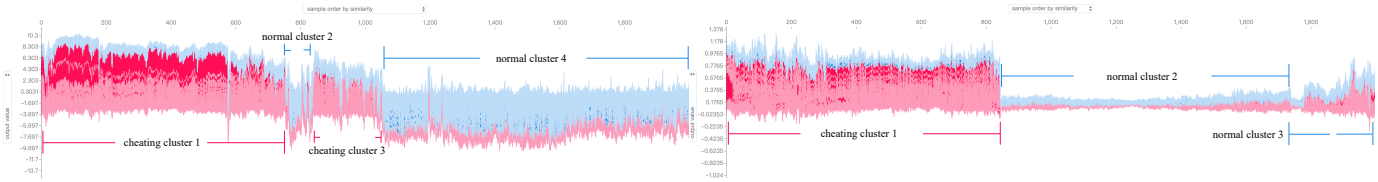
(c) Cheating individual behavior (d) Normal individual behavior

Fig. 6: Individual explanations with SHapley Additive exPlanation (SHAP) [21] values which indicate the attribution of portraits and behaviors. Colors in plots are smoothed as the larger the SHAP value, the redder the color, and the smaller the SHAP value, the bluer the color.

Global portrait explanations. As demonstrated in Figure 8a,8b, the cheaters' computer physical memory and display memory are generally large, and are basically high-end equipment of cheating studios. Normal players are more inclined to watch the game storyline, and spend more time pinching their faces. The cheating players rarely chat, while normal players not only speak in the team, but also make comments on the world channel, and send some emojis. The cheaters regularly create teams and take part in more side missions. Since cheating players do not have strong operability like normal players, the percentage of mission failures and abandoned missions is relatively high. Normal players prefer to involve in PVP gameplay such as SJTX. They are also more active in guild activities. For players with large guild funds, the normal possibility is relatively large, but there are also guild groups with concentrated cheating players. Equipment upgrading is one of the core gameplay of the game. Although the proportion of cheaters participating in this type of gameplay is relatively large, normal players improve their equipment even faster.

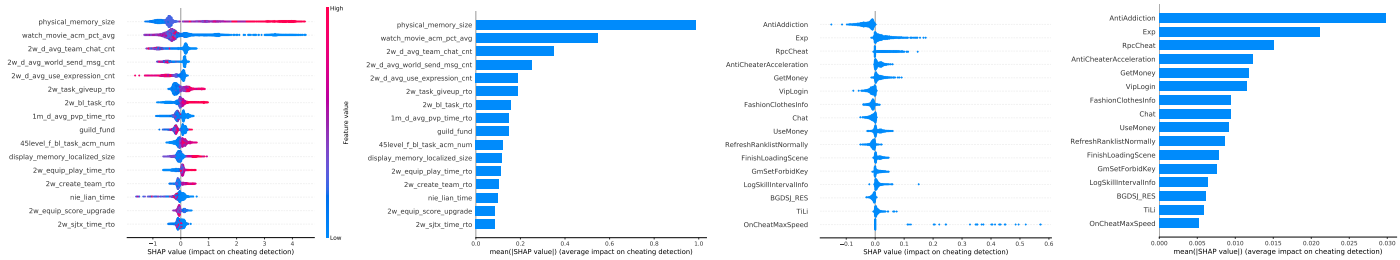
D. Behavior Explanations

Individual behavior explanations. The cheating player in the red block in Figure 6c continuously accepted the task, completed or abandoned the task, and gained money through the game task. In the dark red part at the end of the behavior sequence, this player began to frequently transfer money to other accounts. The normal player in the blue block in Figure 6d participated in some guild activities, and made comments on many channels. But there is a dark red part in the middle that he left the guild, but later he joined a new guild soon.



(a) Game cheating detection with local portrait explanation (b) Game cheating detection with local behavior explanation

Fig. 7: Supervised clustering with individual explanations identifies among 2,000 individuals distinct subgroups of players that share similar reasons for specific roles. Each prediction was explained using individual explanations, and then clustered using hierarchical agglomerative clustering (imagine a dendrogram above the plot joining the individuals). Red feature attributions push the score higher, while blue feature attributions push the score lower (as in Figure 6 but rotated 90').



(a) Global portrait explanation (b) Global portrait explanation bar (c) Global behavior explanation (d) Global behavior explanation bar

Fig. 8: Global explanations of XGBoost models and Transformer models on game cheating detection. Every individual is run through the model and a dot is created for each feature attribution value, so one player gets one dot on each feature's line. Dot's are colored by the feature's value for that player and pile up vertically to show density.

Local behavior explanations. The results in Figure 7b indicate that there are 3 clusters in game cheating detection. The cheating cluster 1 has been punished for PRC cheating, accelerator cheating and GM setting. The normal cluster 2 is charactered by having recharged to buy VIP and fashion clothes, etc. The players in normal cluster 3 participate in PVP gameplay such as BDGSJ, etc.

Global behavior explanations. As illustrated in Figure 8c and Figure 8d, only normal players will normally trigger anti-addiction system reminder, and are willing to talk to other players during the game. The cheaters have had PRC cheating punishment, accelerator cheating punishment and GM setting punishment. They constantly receive quests to improve experience, and consume physical energy more and faster. The cheating players continue to profit from the game while normal players will spend a moderate amount of money in the game. Generally, only normal players will recharge to buy VIP and buy fashion clothes in the game. Normal players will pay attention to their ranking in the game and take part in some gameplays like BDGSJ. As cheating players use special game bots, there will be some bugs related to scene loading or skill releasing interval, which will be recorded in the game logs too.

E. Image Explanations

Figure 9a shows the important pixels of the model to determine the player cheating. The player uses the perspective plugin-in to mark the position of the opponent with a red frame, and at the same time, the player's ID and the remaining health are marked above the frame. For game cheating detection, we cluster important pixels in the individual explanations, and we observe several typical clusters in Figure 9b. Firstly, the plug-in interface appears in the upper left corner. Secondly, the plug-in interface appears in the lower right corner. Thirdly, it exists a large number of red or green frames indicating the position of the opponent just above the middle of the plug-in interface. We aggregate the locations of important pixels. On the whole, the game content display panel and mini-map above the game interface have no plug-in elements, and neither do the game operation panel and chat window below the interface.

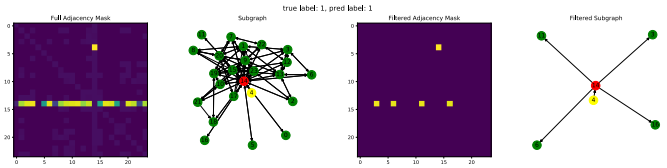
(a) Individual image explanations (b) Local image explanations

Fig. 9: Image explanations for cheating detection.

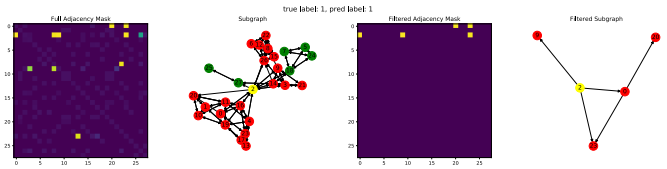
F. Graph Explanations

From the filtered subgraph of Figure 10a, we can observe a typical real money trading [8]. The node 14 is a gold banker

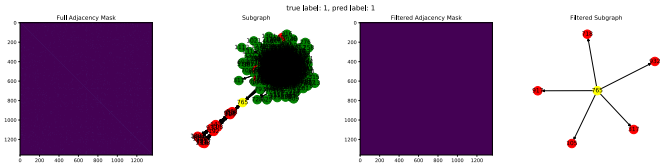
who gathers virtual goods from gold farmers and sell them to gold buyers. The nodes 3,6,11,16 are gold buyers who purchase virtual money with real money from the gold banker. The node 4 is a gold farmer who works full time playing games by using automated programs or by hiring low-cost laborers. As can be seen from Figure 10b and Figure 10c, cheating players are always in the same friend or team circle. Figure 10d demonstrates that the node 3 is a gold buyer. He constantly consults gold bankers (nodes 4,10,33) for real money trading. We cluster the filtered subgraphs in Figure 10, and it is noticeable that there are different structure types under the same relationship. These different types of structural information can be used as a rule to directly search for cheating players.



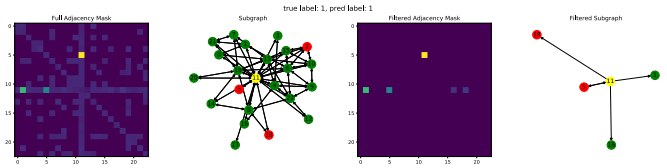
(a) Game cheating detection with individual transaction explanation



(b) Game cheating detection with individual friendship explanation

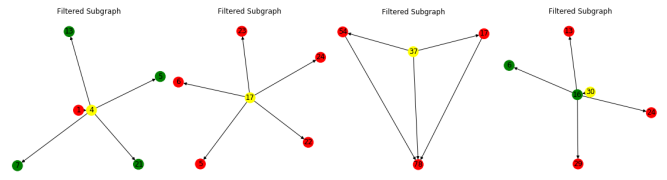


(c) Game cheating detection with individual teaming explanation



(d) Game cheating detection with individual chatting explanation

Fig. 10: (a)-(b) Individual explanations in graph view.



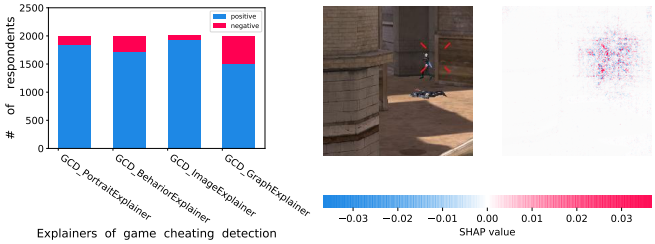
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4

Fig. 11: (a)-(e) Local explanation in transaction view.

V. Applications

A. Evidence and Reason Generation

To validate that the explanations are the most natural and intuitive, we run user studies of game operators and game designers. Participants are not selected for artificial intelligence expertise. Evaluation criteria for explanations consists of truth, usefulness, relevance, coherence with prior belief and generalization. Figure 12a presents the feedback of the user studies for evidence and reason generation. We received about 87.46% of positive feedback from the game operator teams and the image explainer reached a maximum of 96.4%.



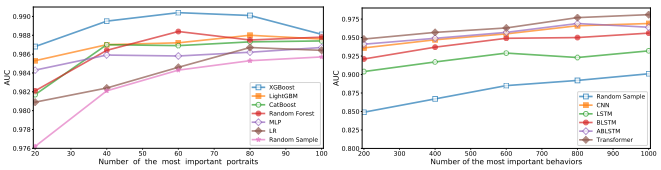
(a) (b)

Fig. 12: (a) User studies for evidence and reason generation. (b) Model debugging with the image explainer.

B. Model Debugging and Compression

The game cheating detection model recognizes normal pictures as cheating pictures in Figure 12b. We analyze the explanations of the image explainer and find that the red crosshairs of the players are similar to the plug-in elements, which led us to mistakenly identify normal players as cheating players. By introducing more similar negative samples in the training set, we quickly improve the accuracy of the model and reduce the error blocking of the online model.

Figure 13 demonstrates the prediction comparison using the same classifier for input with top-k features selected by different explainers. As can be seen in Figure 13a, the predictions with portrait features selected by portrait explainers are better than those selected from random samples, among which XGBoost based explanation results showed the most outstanding performance. We also find that for game cheating detection, we only need 80 features to achieve near-optimal performance. Figure 13b illustrate the similar conclusion for the behavior view, among which Transformer based explanation results show the most prominent performance. We also find that we only need 800 behaviors to achieve nearly optimal performance for game cheating detection. We observe in Figure 6 that the behaviors at different positions are of different importance. For game cheating detection, we merely need to retain the behavior sequences with a length of about 500. Through the above feature selection based on model explanations, we have greatly compress our model and improve the efficiency of model training and online inference.



(a) GCD with portraits (XGBoost) (b) GCD with behaviors (Transformer)

Fig. 13: (a)-(b)Detection for input with top-k features.

VI. Conclusion and Future Work

In this paper, we first introduced the explainable multi-view cheating detection in online games and proposed an EMGCD framework which was evaluated with real-world datasets from NetEase Games. Extensive experiments show the accuracy of classification and the rationality of explanation. We have also discovered and presented quite a lot of interesting and valuable findings from the model explanations. What's more, we implemented and deployed the framework in NetEase Games and received very positive reviews from the game operator teams. We are working for the explainers of the multi-view data fusion based black-box models and expanding our work to more applications in online games like explainable recommender system, explainable churn prediction, etc. We are also focusing on developing an XAI platform for explanations across AI lifecycle with pre & post deployment for AI models. Demos will come soon for the following demo tracks.

VII. Acknowledgements

Thanks to the game operator teams and game designer teams for the helpful feedback. Thanks to Jiaheng Qi and Han Yan for the additional experiments. Thanks to Xiaochuan Feng, Jiaxi Li for the visualization of the datasets and the experimental results.

References

[1] S. Morris, "Wads, bots and mods: Multiplayer fps games as co-creative media." in *DiGRA Conference*. Citeseer, 2003.

[2] S. R. Saikia, "A survey of mmorpg architectures," 2008.

[3] J. Yan and B. Randell, "A systematic classification of cheating in online games," in *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, 2005, pp. 1–9.

[4] M. L. Bernardi, M. Cimitile, F. Martinelli, and F. Mercaldo, "A time series classification approach to game bot detection," in *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics*, 2017, pp. 1–11.

[5] J. Tao, J. Xu, L. Gong, Y. Li, C. Fan, and Z. Zhao, "Nguard: A game bot detection framework for netease mmorpgs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 811–820.

[6] E. Lee, J. Woo, H. Kim, and H. K. Kim, "No silk road for online gamers! using social network analysis to unveil black markets in online games," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1825–1834.

[7] J. Hwang, "Beating wallhacks using deep learning with limited resources," in *Proceedings of the 2018 World Wide Web Conference*, 2019.

[8] J. Tao, J. Lin, S. Zhang, S. Zhao, R. Wu, C. Fan, and P. Cui, "Mvan: Multi-view attention networks for real money trading detection in online games," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 2536–2546.

[9] A. Liaw, M. Wiener *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[10] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[11] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in neural information processing systems*, 2017, pp. 3146–3154.

[12] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, 2018, pp. 6638–6648.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[21] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," *arXiv preprint arXiv:1802.03888*, 2018.

[22] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[23] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you?: Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2016, pp. 1135–1144.

[24] H. Deng, "Interpreting tree ensembles with intrees," *International Journal of Data Science and Analytics*, vol. 7, no. 4, pp. 277–287, 2019.

[25] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, "Local rule-based explanations of black box decision systems," *arXiv preprint arXiv:1805.10820*, 2018.

[26] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3145–3153.

[27] H. Liu, Q. Yin, and W. Y. Wang, "Towards explainable nlp: A generative explanation framework for text classification," *arXiv preprint arXiv:1811.00196*, 2018.

[28] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 3319–3328.

[29] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[30] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[31] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 9240–9251.

[32] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *arXiv preprint arXiv:2001.06216*, 2020.