ISSN 2959-6157

# Leveraging Machine Learning: Advancements in Cheating Detection Strategies for Ensuring Fair Online Gaming Environments

**Lexuan Zhang[1,*], Yuhan Wei[2], Yitong Zhou[3]**

[1]*Beijing Jiaotong University, Beijing, China, lexuanzhang@icloud.com*
[2]*Shijiazhuang Foreign Language School, Shijiazhuang, China, weiy93285@gmail.com*
[3]*Ohio State University, Suzhou, China, arcyiiizzz@gmail.com*

**Abstract:**

Cheating detection in online gaming is a crucial challenge that affects the fairness and integrity of virtual environments. This literature review delves into the advancements made in the field of cheating detection, focusing on machine-learning-based approaches and encrypted network traffic analysis. Various methodologies, including Support Vector Machines, Logistic Regression, and GPU acceleration, are explored in detecting cheating behaviors within different gaming scenarios. The review also examines the application of supervised learning techniques in Unreal Tournament III, showcasing their potential in identifying cheating instances. Additionally, the challenges posed by limited labeled data and covariate shifts in encrypted network traffic analysis are addressed through innovative solutions like the GCI framework. Insights into the interplay of data attributes and classification performance are provided, offering directions for future research. Overall, this review contributes to the understanding of cheating detection strategies and their implications for maintaining equitable and enjoyable online gaming experiences.

**Keywords:**- Machine Learning, Online Gaming, Fair, Cheating Detection

## 1. Introduction

With the rise and robust development of the gaming industry, an increasing number of people are experiencing excitement and pleasure from the competitive nature of games, particularly in multiplayer modes. Various competitive mechanisms, such as leaderboards and scoring systems, have triggered individuals' desire for victory. With each successive triumph, players' excitement thresholds are elevated, leading people to gradually resist investing significant time and effort for a simple victory. Consequently, the gaming cheating industry stealthily emerged alongside this trend, even giving rise to commercialized cheating systems, forming a series of exclusive cheating frameworks.

Dialectically speaking, the continuous development of gaming cheating systems underscores the need for enhanced network security. It indirectly reveals areas in game development that require refinement while also highlighting the ongoing optimization of cheating techniques. However, it must be acknowledged that the rampant growth of cheating practices significantly impacts players' gaming experience by undermining fairness and introducing technical imbalances in multiplayer games.

In order to foster a positive cyclic atmosphere within the gaming industry, we should continuously optimize anti-cheating technologies. It's worth noting that detecting cheating behavior, due to its elusive nature, typically necessitates manual oversight. So we should focus our research efforts mainly on automatic identification techniques.

This literature review is based on three papers that focus on the use of machine learning techniques to detect cheating in online games. In the first two papers, through different experimental methods, the researchers classified the data and created a cheating detection model using various supervised learning techniques such as logistic regression, decision trees, naive Bayes, random forests, neural networks, and support vector machines by collecting game log data and pre-processing the data according to the established framework model of the anti-cheating system. The results of the study show that in most cases, the mentioned methods have a very high accuracy and provide some recommendations and analyses for online game developers. In addition, the third paper proposes a supervised learning approach to detect cheating based on encrypted network traffic. Due to limited training data labelling, the authors address this challenge by using relative density ratios to estimate the importance weights associated with training data instances. The authors also

demonstrate the server-side scalability of their proposed approach using Apache Spark
and achieve better performance using GPU acceleration. They collected and empirically evaluated network traffic data from Counter-Strike games and showed that their approach significantly outperforms other benchmark approaches in detecting cheating behaviour. All three articles comment on the field of anti-cheating and provide experimental models that have been validated for effectiveness, and present basic ideas for future directions in the development and optimisation of anti-cheating techniques. The aim of this paper is to synthesise and further extend these findings by proposing a comprehensive framework to automatically detect cheating behaviours and guarantee a fair and positive gaming experience in online games.

## 2. Background

Online gaming has transformed into a global phenomenon, engaging millions of players in immersive virtual environments. However, as the popularity of online games rises, so does the prevalence of cheating behaviors that undermine the integrity of these platforms. Cheating not only compromises the fairness of gameplay but also erodes the trust and enjoyment that players derive from the virtual worlds they inhabit. Understanding the motivations behind cheating and its repercussions on the gaming ecosystem is imperative in devising effective strategies to curb such behavior.

Players resort to cheating for various reasons. Competitive pressures, the desire for in-game rewards, and the pursuit of social recognition drive some individuals to exploit vulnerabilities within the game mechanics. The allure of gaining an unfair advantage, whether through automated scripts, aimbots, or wallhacks, can be irresistible, leading players down a path that damages the gameplay experience for others. Cheating diminishes the sense of accomplishment, challenges the balance of the game, and alienates honest players, thereby undermining the very essence of fair competition and camaraderie that defines online gaming.

The impact of cheating extends beyond individual experiences. Game developers are tasked with the formidable challenge of maintaining the integrity of their virtual universes. Cheating can introduce a negative spiral, as legitimate players become frustrated and may eventually abandon the game, leading to dwindling player communities and reduced revenues for developers. To counteract these negative outcomes, developers are compelled to invest resources into anti-cheating measures, diverting efforts away from enhancing gameplay or introducing new features.

In light of these challenges, the application of machine learning techniques has emerged as a promising approach to combat cheating behaviors. The complexity and scale of online gaming environments make manual detection of cheating nearly impossible. Here, machine learning shines, offering the potential to analyze vast datasets, recognize patterns, and differentiate between normal and suspicious behavior. This technology-driven approach allows for the efficient identification of cheaters while minimizing the intrusion on players' privacy and gaming experience.

In the pursuit of effective cheating detection methodologies, recent research has explored diverse avenues. Notably, the integration of machine learning algorithms, such as Support Vector Machines and Logistic Regression, holds promise in identifying cheating behaviors with high accuracy. These algorithms can decipher intricate patterns in gameplay data, providing insights that facilitate the detection of anomalous activities.

Moreover, the evaluation of encrypted network traffic, as seen in the GCI framework, demonstrates an innovative response to the challenges of limited labeled data and distributional shifts. By leveraging the relative density ratio and GPU acceleration, researchers strive to improve the accuracy and scalability of cheating detection mechanisms. These advances collectively contribute to the ever-evolving landscape of cheating detection in online gaming.

In conclusion, the backdrop of cheating in online gaming is a complex interplay of motivations and consequences that extends beyond individual experiences. As the gaming community grapples with the pervasive issue of cheating, machine learning emerges as a formidable ally in maintaining fair and enjoyable virtual environments. Through innovative methodologies and interdisciplinary research, the field advances toward a future where cheating becomes a rarity, preserving the essence of competition and camaraderie that defines the world of online gaming.

## 3. Behavioral-Based Cheating Detection in Online First Person Shooters Using Machine Learning Techniques

The original text[1] introduces a machine-learning-based method for detecting cheating in online games. The research team developed an FPS game using the Unity3D game engine and performed data preprocessing by collecting game logs. Subsequently, they created a cheating detection model using algorithms such as Support Vector Machines and Logistic Regression. Experimental results demonstrate that this method achieves high accuracy in both single and multi-class cheating detection. By ex-

clusively utilizing game logs for cheating detection, the approach avoids privacy infringement concerns, and the paper provides practical insights for game developers.

The process of combating cheating behavior in online games involves a series of interconnected steps, each contributing to an enhanced understanding of player interactions and the identification of irregularities. Data collection was carried out through 18 distinct deathmatch games, characterized by a player employing cheating software and another adhering to standard gameplay. This rich dataset served as the foundation for subsequent analyses.

Feature extraction played a pivotal role, where data was categorized into distinct time frames (10s, 30s, 60s, and 90s). Employing the proven techniques of logistic regression and support vector machines (SVM), two prominent classifiers, yielded a comprehensive evaluation of player behavior. Leveraging both linear and radial basis function (RBF) kernels, SVM's parameter optimization ensured the adaptability of the model's soft boundaries.

The most crucial data table in this paper is Table 1. This table presents the results of testing three players using different models and frame sizes. The table is divided into three sections: Table 1(a) displays the number of instances collected for each frame size, Table 1(b) showcases the best results obtained using the models from the first three sections, and finally, Table 1(c) demonstrates the optimal outcomes achieved using the model from the fourth section. Within Table 1(c), the author illustrates the detection accuracy for each cheating type and normal gameplay, with the authors selecting True Positive Rate (TPR) to capture the detection accuracy of each model when encountering unknown cheating types.

**Table 1: Test Set Results**

(a) Number of instances

| Frame Size | Number of instances | | | |
|---|---|---|---|---|
| | L | AF | Normal | Total |
| 30 | 59 | 41 | 58 | 158 |
| 60 | 29 | 24 | 32 | 85 |

(b) Accuracy results for Parts 1, 2 and 3

| Part | Best Selected Results | | | |
|---|---|---|---|---|
| | Accuracy | Frame Size | Classifier | |
| 1 | 78.8% | 60 | SVM-L | |
| 2 | 94.1% | 60 | SVM-RBF | |
| 3 | 98.8% | 60 | SVM-L | |

(c) Accuracy results for Parts 4

| Cheat Type | Best Selected Results | | | | |
|---|---|---|---|---|---|
| | TPR for L | TPR for AF | TPR for Normal | Frame Size | Classifier |
| L | 96.6% | 2.4% | 100% | 60 | SVM-RBF |
| AS | 100% | 0% | 100% | 60 | Both |
| AM | 89.7% | 0% | 100% | 60 | SVM-RBF |
| SA | 89.7% | 0% | 100% | 60 | Both |
| AF | 55.2% | 100% | 96.9% | 60 | SVM-L |

According to the analysis of the results in Table 1(c), the SVM-L and SVM-RBF models, along with data from frame sizes 30 and 60, yield the best cheating detection outcomes. For the "Aimbot" cheating type, the most accurate feature is "Mean Aim Accuracy," which consistently ranks highly across different frame sizes. Regarding the "Auto-Fire" cheating type detection, the most informative features are "Fire-on-Visible" and "Fire-on-Aim," as these capture instances of cheating behavior where firing occurs immediately upon having a target in the crosshair, as high-

lighted by the authors.

The setting of detection thresholds emerged as a critical element, guided by a nuanced interplay of model accuracy and developer policies. In the absence of labeled data, this study creatively addressed the challenge by proposing a flexible threshold approach. As an example, a 60-second time frame and a 50% threshold were employed, a strategy adaptable to both real-time and offline scenarios.

Evaluating the efficacy of the proposed cheating detection methods involved comprehensive testing across multiple players and time frames. SVM-L and SVM-RBF classifiers, along with 30-second and 60-second intervals, were employed. The true positive rate (TPR) provided valuable insights into the detection accuracy across various forms of cheating and normal gameplay, thus affording a thorough comprehension of the model's performance.

By assimilating the advances from various fronts, this study achieved significant progress. The inclusion of manifold feature learning, utilization of diverse classifiers such as logistic regression and SVM, and an encompassing experimental design adapted to commercial online games contributed to the overall evolution of cheating detection methodologies. Moreover, the introduction of flexible threshold settings allowed for pragmatic adjustments, enhancing the model's adaptability in addressing different forms of cheating.

In summary, this study represents a substantial advancement in the realm of cheating detection. Leveraging a holistic approach that combines intricate feature extraction, precise classifier selection, pragmatic threshold settings, and real-world application, this research significantly improves the accuracy, adaptability, and applicability of cheating detection methods. As compared to previous works, this study's comprehensive treatment of feature selection, diversified classification employment, comprehensive experimental design, and adjustable threshold strategies collectively contribute to a more robust and effective cheating detection paradigm. Through these integrated methodologies, the study holds profound implications for the advancement of fair and enjoyable online gaming experiences.

# 4. A Cheating Detection Framework for Unreal Tournament III: A Machine Learning Approach

This paper[4], using Unreal Tournament III as a medium, introduces a novel framework for automatically detecting cheating behaviors through supervised learning techniques within the conceptual domain of machine learning.

The framework consists of three main components: (i) an extended game server for collecting game data, (ii) a processing backend responsible for preprocessing data and detecting cheating behaviors, and (iii) an analysis frontend for result analysis.

Through experimental analysis involving three human players, three game maps, and five different supervised learning techniques, the paper also presents the specific method of training the training set using default parameter configurations. In the experiment, five different supervised learning methods, including decision trees, naive Bayes, random forests, neural networks (backpropagation algorithm), and support vector machines, were employed to detect cheating behaviors in the game Unreal Tournament III, confirming the effectiveness of the framework.

The results show that all supervised learning techniques are capable of correctly classifying nearly 90 percent of the samples in the test examples. In this experiment, the author uses data from training and test sets to evaluate the performance of different methods. By comparing the classifier's classification results for samples in the test set, their accuracy and misclassification can be determined. In relation to the paper's results and relevant discussions, the paper primarily presents the statistical indicators calculated for each frame of data. These indicators encompass various aspects of data, such as players' shooting frequency when the target opponent is visible, aiming accuracy during shooting, types of weapons used, and the health status of target enemies. Each frame of data is labeled based on whether a player employs cheating systems, with players using cheating systems being marked as cheating instances. In conclusion, the content covers framework introduction, application of supervised learning techniques, experimental design, performance evaluation, and statistical data analysis.

## 4.1 . Unreal Tournament III

Unreal Tournament III is a very popular commercial first-person shooter with impressive rendering capabilities based on the Unreal Engine, which also utilizes NVDIA's PhysX engine to accurately simulate the physics and dynamics of the game world. UT3 was developed using the Unreal Script programming language, a Java-like scripting language interpreted by the Unreal Engine.

A two-tier architecture is a client-server architecture in which the client application connects and interacts directly with the server. This architecture usually consists of a front-end client and a back-end server and has the distinct advantage of being flexible: the two-tier architecture allows for greater flexibility in customizing and configuring the application, as both the client and the server can be customized and optimized as needed. This makes the application more adaptable to specific needs and requirements.

### 4.2 . Commercial Cheating Systems

The paper begins by describing two important features of a cheating system: the aiming and firing assistance system and the radar system. The aiming and shooting assistance system includes auto-targeting and auto-firing features. Auto-targeting helps players target more easily and includes advanced features such as slow targeting, visible viewpoints, and targeting for different parts of the body. Autofire is a relatively simple mechanism that automatically fires when a suitable target appears under the aiming star. Radar systems are a common feature in commercial cheat systems, including 2D radars and
3D radars that show the location of all other players on the map.
In this article, the authors also mention that commercial cheat systems often provide some manual assistance features, such as the automatic location of opponents and extended information about their status. However, the source code of commercial cheat systems is not available and cannot be extended or modified. In addition, the licensing policies of commercial cheating systems may limit the experimental setup for data collection. Therefore, the authors chose to develop their own cheating system that includes the main features provided by the commercial system.

### 4.3 . Supervised Learning Techniques

Decision trees[10] use the tree induction algorithm in RapidMiner[7], using gain ratio as a splitting criterion; Naive Bayesian classifiers[8] use Laplace correction to prevent the effect of zero probability; Random forests[2] are made up of ten random trees, using gain ratio as a splitting criterion; Neural networks[5] use a feedforward neural network with a single hidden layer, and the training algorithm is backpropagation, with the number of hidden nodes set to (number of attributes + number of categories)/2+1 and sigmoid activation function is used for all nodes; Support Vector Machine[11] uses JMySVMLearner in RapidMiner and internally uses dot kernel function for training.

### 4.4 . Experiment Design

This section partially describes in detail the information related to the log collection in the experiment, including the participants, number of matches, map selection, AI settings, and the enablement and settings of the cheating system.
Logs were collected from eight matches on two different maps (Biohazard and Rising Sun) involving two players (an expert player and a novice player) in the experiment. Each player played two 20-minute deathmatch matches against a CPU-controlled AI on each map. One of the matches was played against a novice AI (i.e., the easiest difficulty in the game), while the other was played against a God-level AI (i.e., the hardest difficulty in the game). During each match, players were given the option to enable or disable the cheat system as they wished. However, they were asked to play the game for roughly the same amount of time with and without the cheat system enabled. In all experiments, the settings for the cheat system were the same, i.e., players could not change the settings; aim assist was enabled, and slow aiming was turned on.
The final section of the experimental design section focuses on the author's construction of a test set from a completely new set of logs to obtain an unbiased and reliable assessment of the performance of the learned model. The authors collected logs from 39 matches involving three different maps and three different levels of human players. Each match was a two-minute deathmatch between the human player and a CPU- controlled AI. During these matches, players could not turn the cheat system on or off; in 18 of the matches, the cheat system was active, while in the remaining 21 matches, the cheat system was inactive. When the cheating system was active, the configuration was exactly the same as when the training set was previously constructed. Finally, the authors preprocessed the collected data by extracting 30 seconds of data from each recorded match to construct a test set containing 39 examples.

### 4.5 . Detailed Description of the Experimental Results– The Relationship Between the Discrimination of False Positives and False Negatives and the Dependency Between Data Attributes

In the process of processing data, it is extremely important to correctly identify and classify individual specific data points and to correctly associate them, which will affect the judgment of experimental results and data reporting to a certain extent. The authors explore the relationship between false negatives and positives and data attributes by analyzing and discussing the results of a cheating detection experiment.
Firstly, the definition: in this experiment, we define false negatives to mean the misclassification of real cheating as no cheating and false positives to mean the misclassification of no cheating as cheating.
The experimental results show that it is possible to have false positives when using different techniques for cheating behavior detection. In this case, a false positive refers to misclassifying an example of actually honest behavior as cheating. However, in the reported confusion matrix, only the plain Bayesian classifier did not misclassify any of the samples without cheating behaviour as having cheating behaviour (2.B), i.e., it did not produce false

positives. In contrast, the support vector machine reported one false positive misclassification in the confusion matrix with the same overall accuracy (2.E).

**Table 2: confusion matrix results**

| (A) | | |
|---|---|---|
| Predicted Class | True Class | |
| | Cheater | Honest |
| Cheater | 17 | 2 |
| Honest | 1 | 19 |
| (B) | | |
| Predicted Class | True Class | |
| | Cheater | Honest |
| Cheater | 17 | 0 |
| Honest | 1 | 21 |
| (C) | | |
| Predicted Class | True Class | |
| | Cheater | Honest |
| Cheater | 17 | 1 |
| Honest | 1 | 20 |
| (D) | | |
| Predicted Class | True Class | |
| | Cheater | Honest |
| Cheater | 18 | 4 |
| Honest | 0 | 17 |
| (E) | | |
| Predicted Class | True Class | |
| | Cheater | Honest |
| Cheater | 18 | 1 |
| Honest | 0 | 20 |

False positives may be more critical than misses when considering the practical demands of the problem. Since the goal of a cheat detection system is to identify cheaters and prevent them from accessing online gaming services, a false positive can result in an innocent player being mistaken for a cheater and prevented from using a service for which he/she may have paid a fee.

We want to know where to start to improve the accuracy of our experimental results by exploring what factors false negatives and positives are directly related to so that we can begin to optimize the technique. False positives represent a strong dependency on data. This is because, in cheating detection, it is not only the characteristics of individual data points that need to be considered but also the correlation between data points. In a given situation, misclassifying an example of honest behavior as cheating may be due to the fact that the features or attributes of a particular data point are similar to those of the cheating behavior but do not represent cheating in its entirety.

To summarise, for the relationship between false negatives and data attributes, from the experimental results, the performance of the plain Bayesian classifier is relatively good, suggesting that the interdependence between data attributes is not very significant in its impact.

In order to solve this problem, researchers need to carefully analyze the relationship between the data and determine which features or attributes may lead to the appearance of false positives. By further optimizing the classification algorithm, the occurrence of false positives can be reduced, and the accuracy and reliability of the cheating detection

system can be improved.

### 4.6 . Summary

This study presents a framework for the automatic detection of cheating behavior in Unreal Tournament III. The framework constructs five different decision models using supervised learning techniques, including decision trees, plain Bayes, random forests, neural networks, and support vector machines. The performance of these five learning models is compared by applying them to a previously collected test dataset. It was shown that all supervised learning techniques were able to correctly classify nearly 90 percent of the test samples, successfully testing the framework's effectiveness.

## 5. GCI: A Gpu-Based Transfer Learning Approach For Detecting Cheats of Computer Game

The primary focus of this paper[6] is to use supervised machine-learning techniques for online game cheating detection in encrypted network traffic. The authors propose a method that addresses the challenge of limited labeled data and covariance shift during model training by estimating important weights associated with training data instances through relative density ratios. They also introduce a technique to learn model parameters automatically for estimating relative density ratio from existing data.

Supervised machine learning techniques are used for online game cheating detection in encrypted network traffic because they provide a generic solution for most multiplayer games. By analyzing the encrypted network traffic, the author aims to detect cheating behaviors and distinguish them from normal gameplay. This approach is beneficial because it does not rely on the availability of the game's source code and can be applied to various games without the need for game-specific detection mechanisms.

To handle the massive traffic data consistently encountered on the server side, the authors employed a distributed computing framework, Apache Spark, to implement their approach. The authors further explored using Graphics Processing Units (GPUs) as an alternative hardware solution to accelerate the performance of the cheating detection mechanism. By collecting network traffic of players in the Counter-Strike game, they concluded that their innovative approach significantly outperforms other benchmark cheating mechanisms.

The structure of their GCI[3] (Game Cheating Identification) framework is depicted in Fig 1. Their training dataset includes game traffic information from players, marked as either ' cheating' or'normal'. They intend to identify cheaters using a test set filled with game traffic data from various players. This test set is then randomly split into two segments.

One segment collaborates with the training set to formulate the classification model, while the other is utilized to assess the effectiveness of their methodology. Initially, features are extracted from the gaming traffic data provided by players. They believe there's a distributional disparity, termed co-variate drift, between the training and test sets, likely arising from sampling variations. With this covariate drift in mind, they apply a Gaussian kernel model to gauge the relative density ratio linked to the training data. They recognize that using instance weights can offset the distributional biases in training data, making trained classifiers with weighted samples more adept at generalizing to test instances. Consequently, their GCI method incorporates weighted training samples for predicting test sample labels. Their procedure culminates in creating a bias-modified classifier, which then uses its predictions to pinpoint potential cheaters in the subsequent test data.

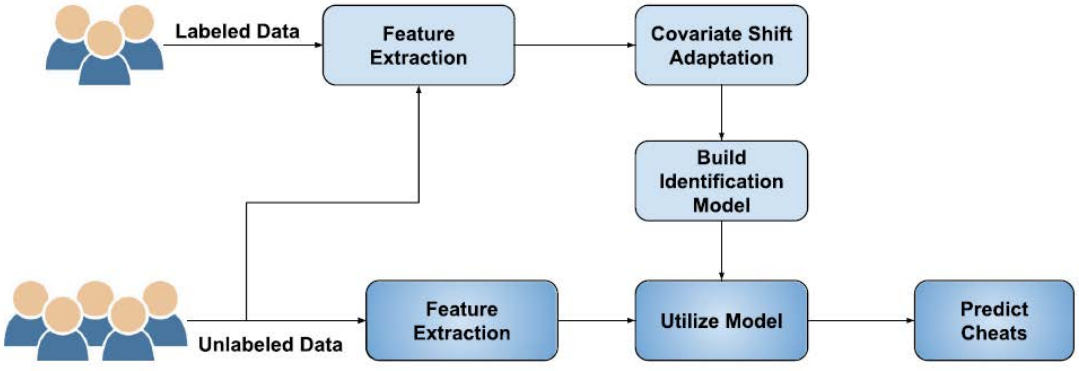


**Figure 1: confusion matrix results**

The first step is feature extraction, which uses the methodology called BIND introduced[9]. This paper utilizes the BIND method to extract features solely from the packet header information in encrypted packet traces. The main

objective is to discern patterns in network traffic to detect cheating behaviors by examining consecutive packets for potential dependencies. BIND investigates packet sequences, such as single bursts (or uni-bursts) and bursts (two neighboring uni-bursts going in opposite directions). Bursts can be from client to server (uplink) or from server to client (downlink), as shown in Fig 2. Bursts possess attributes like size, time, and direction, where size is the combined lengths of its packets, and time is the gap between the first and last packet timestamps in a burst. Features from Bi-Bursts typically fall into four groups: Dn-Up-Burst size and time and Up-Dn-Burst size and time. Each trace yields distinct tuples that capture these four categories. Both Dn-Up and Up-Dn categories define tuples based on adjacent uni-burst sequences in terms of their size and time. These features are merged to create a comprehensive feature array from each trace. Multiple traces represented this way make up the training and testing sets. Fig 2 showcases a trace with uplink and downlink sequences, explaining how sizes and times of bursts are determined and represented in tuples.

As mentioned before, covariance shift is another challenge because the assumption that the training set and test set have similar distributions may not hold true in many scenarios in the real world. The GCI's Covariate Shift Adaptive (CSA) relative density ratio estimation using a Gaussian kernel model can be used to address this issue. As mentioned before, covariance shift is another challenge because the assumption that the training set and test set have similar distributions may not hold true in many scenarios in the real world. The GCI's Covariate Shift Adaptive (CSA) relative density ratio estimation using a Gaussian kernel model can be used to address this issue. This statistical method has the potential to recognize patterns and anomalies in the in-game behavior of players. By analyzing gameplay data with the Gaussian kernel, inconsistencies associated with the aforementioned cheats like Aim-bot, Speed-hack, and Wall-hack can be highlighted. Essentially, the Gaussian kernel works by transforming the input data into a higher-dimensional space, enabling a clearer separation of cheat-induced behaviors from genuine gameplay patterns. Thus, by mapping game behaviors onto this model, the system can effectively identify and isolate instances of cheating, ensuring a fair play environment.

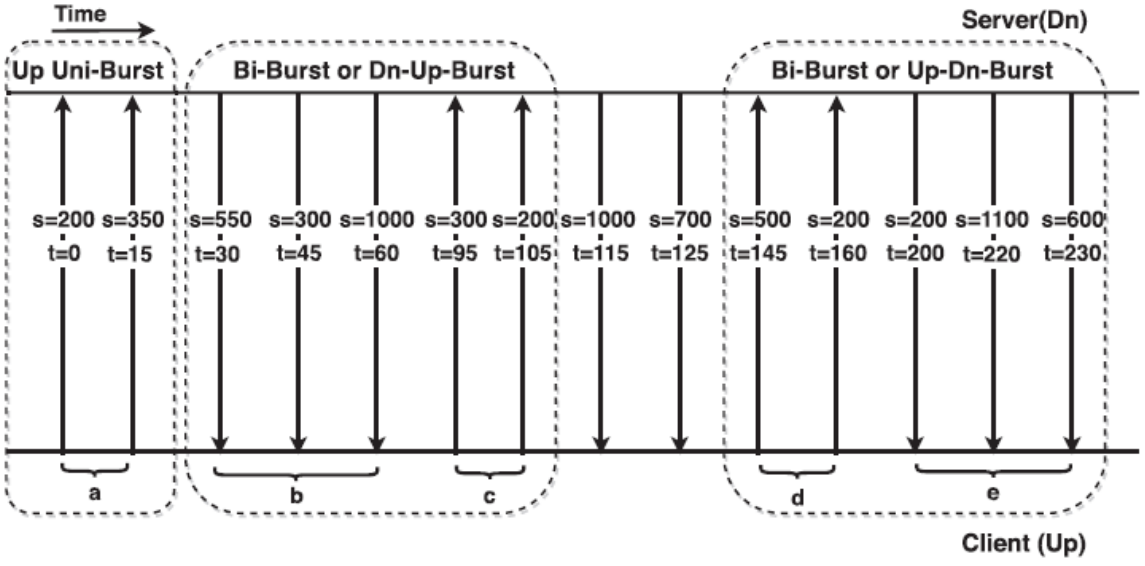


**Figure 2: confusion matrix results**

In the referenced study, the GCI framework was used on the server side to detect cheat by encrypting the game's network traffic. Three key rationales underpinned this decision. Firstly, the encryption provided a game-agnostic cheat detection approach, as deciphered traffic inherently engaged game-specific variables. Secondly, encrypted traffic acted as an initial checkpoint before delving into more detailed and time-intensive detection methods. Lastly, given the proprietary nature of most games, encrypted traffic facilitated a simpler evaluation process. Although the mechanism delineated in the study isn't confined to a specific game, its server-side construction processes a vast amount of traffic data from diverse player cohorts. Such extensive data processing can potentially strain memory and overall performance. To counteract these challenges, the study's authors employed Apache Spark, enhancing the processing speed for the task.

Researchers aim to harness the general-purpose program-

ming capabilities of GPUs to establish an economical yet scalable cheat detection solution for MMOGs. Modern Graphics Processing Units (GPUs) have proven instrumental in en- hancing performance across numerous real-world applications, presenting an avenue for cost-effective parallel processing. By capitalizing on the GPUs' superior memory bandwidth and parallel processing capabilities, there is a notable reduction in the computational load on the CPU. The most time-consuming components of the researchers' methodology encompass parameter learning and hyper-parameter estimation for the estimator. The hyper-parameter values of the SVM model are refined through a comprehensive grid search, focusing particularly on parameters like the regularization coefficient, which helps strike a balance between decision boundaries and misclassification. For enhanced processing efficiency, ThunderSVM, a GPU-optimized toolkit, is integrated. This toolkit is specifically tailored for SVM applications, strongly emphasizing algorithmic parallelization and memory optimization, ensuring a notable reduction in operational run-time.

In conclusion, this paper proposes a supervised machine learning approach called GCI (Game Cheating Identification) to detect cheating in MMOGs during gameplay. The authors address the challenge of limited labeled data by utilizing the relative density ratio and developing a technique for automatic parameter estimation. The scalability of the approach is demonstrated using Apache Spark for server-side cheat detection, and the performance of GPU-based implementation is compared with Spark and baseline approaches. Empirical evaluation on real-world network traces from Counter-Strike shows significant improvement in cheat detection compared to other methods. The feature extraction methodology, BIND, is used to capture patterns in network traffic and identify cheating behavior.

## 6. Conclusion

In the realm of cheating detection within online gaming, the studies reviewed have collectively illuminated several influential contributions. The introduced methods and frameworks offer valuable insights into tackling the persistent challenge of identifying and mitigating cheating behaviors in diverse gaming environments. These findings not only advance the field of cheating detection but also hold implications for the broader landscape of maintaining fairness and integrity in online gaming experiences.

The presented research outlines the efficacy of machine-learning-based techniques in identifying cheating behaviors within the context of online games. By employing approaches like Support Vector Machines and Logistic Regression, re-markable accuracy rates have been achieved across various forms of cheating and gaming scenarios. The utilization of game logs for detection purposes highlights the potential for practical anti-cheating solutions that respect player privacy while effectively addressing developers' concerns.

The comprehensive approach outlined for Unreal Tournament III demonstrates the applicability of supervised learning techniques in detecting cheating instances. The integration of distributed computing frameworks and the exploration of GPU acceleration underscore the adaptability and scalability of these methodologies, aligning them with the needs of real-world server-side processing.

Furthermore, in the domain of encrypted network traffic, the literature provides insights into addressing limited labeled data and the challenge of covariate shift. The proposed GCI framework introduces an innovative solution through relative density ratios, enabling more accurate training data instance weighting. The incorporation of GPU-based acceleration and parameter learning offers efficient cheating detection mechanisms that outperform benchmark methods.

The studies reviewed also shed light on the interplay between different data attributes and their influence on classification performance. This understanding provides valuable directions for future enhancements in detection accuracy and reliability. By addressing practical challenges associated with encrypted traffic analysis, the reviewed research contributes to the broader discourse on preserving fairness and transparency in online gaming environments.

To summarize, the literature explored in this review collectively advances the field of cheating detection within online gaming. The methodologies and insights shared offer practical solutions, laying the foundation for more robust anti-cheating mechanisms. Through these contributions, game developers, researchers, and gaming communities are better equipped to ensure equitable and enjoyable experiences for players across various online gaming platforms.

## Acknowledgment

We extend our sincere gratitude to Professor William Nace for his invaluable guidance and support throughout the development of this literature review. Professor Nace's expertise, insights, and mentorship have greatly enriched our understanding of the field of cheating detection in online gaming. His thoughtful feedback and encouragement have been instrumental in shaping the direction and content of this review. We are truly grateful for his dedication and contributions, which have significantly contributed to

the quality and depth of this work.

## References

[1] Hashem Alayed, Fotos Frangoudes, and Clifford Neu-man. “Behavioral-based cheating detection in online first-person shooters using machine learning techniques”. In: *2013 IEEE conference on computational intelligence in games (CIG)*. IEEE. 2013, pp. 1–8.

[2] Leo Breiman.“Random forest, vol. 45”. In: *Mach Learn* 1 (2001).

[3] B. Dong et al. “GCI: A transfer learning approach for detecting cheats of computer game”. In: *Proc. IEEE Int. Conf. Big Data*. 2018, pp. 1188– 1197.

[4] Luca Galli et al. “A cheating detection framework for unreal tournament iii: A machine learning approach”. In: *2011 IEEE Conference on Computational Intelligence and Games (CIG’11)*. IEEE. 2011, pp. 266–272.

[5] Simon Haykin. “Neural Networks: A Comprehensive Foundation, MacMillan College Publishing Co”. In: *New York* (1994).

[6] Md Shihabul Islam et al. “GCI: A GPU-Based Transfer Learning Approach for Detecting Cheats of Computer Game”. In: *IEEE Transactions* on *Dependable and Secure Computing* 19.2 (2020), pp. 804–816.

[7] Ingo Mierswa et al. “Yale: Rapid prototyping for complex data mining tasks”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 935–940.

[8] Tom M Mitchell. *Machine learning*. 1997. [9] K. Al-Naami et al. “Bimorphing: A bi-directional bursting defense against website fingerprinting attacks”. In: *IEEE Trans. Dependable Secure Comput.* (2019).

[10] RC Quinlan.“4.5: Programs for machine learning morgan Kaufmann Publishers Inc.”. In: *San Francisco, USA* (1993).

[11] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT Press, 2018.