National College of Ireland

Anomaly Detection for Identifying Cheating Behaviours and Techniques in Online Gaming Using AI

MSc Research Project
MSc. In Cyber Security

Sachet Satish Karkera
Student ID: X21224838

School of Computing

National College of Ireland

Supervisor: Khadija Hafeez

`

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sachet Karkera		
Student ID:	X21224838		
Programme:	MSc. In Cyber Security	**Year:**	2023/24
Module:	Research Internship		
Supervisor:	Khadija Hafeez		
Submission Due Date:	31/01/2024		
Project Title:	Anomaly Detection for Identifying Cheating Behaviours and Techniques in Online Gaming Using AI		
Word Count:	7148	**Page Count:**	**21**

Signature: Sachet Karkera

Date: 30/01/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placedinto the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

`

Anomaly Detection for Identifying Cheating Behaviours and Techniques in Online Gaming Using AI

Sachet Satish Karkera

21224838

Abstract

Gaming has been gaining popularity since it has become mainstream and has become one of the largest sectors in terms of money, investment, and involvement. By hacking into the game's mechanics to alter the results of a particular match to their liking, some players may turn to illicit and immoral strategies to improve their performance. Cheats such as aimbots, wallhacks, and bots playing or impersonating as real players have been a threat to the gaming community. This compromises fair play and discourages people with no experience from attempting to become competent in a particular game. Employing a visual object detection algorithm, this research attempts to evaluate current cheat detection strategies while putting new methodologies for identifying Aim Bot, Wallhack, and Speedhacks in online gaming. Aim Bot detection includes statistical analysis and dynamic thresholding techniques to identify and flag instances of aim bot usage. Wallhack and Object detection utilises YOLOv8, an innovative algorithm, enabling real-time identification of wallhack usage. Speedhack detection incorporates tick rate analysis and pattern recognition to detect and flag instances of speedhack usage. This research intends to eliminate cheating methods while protecting the player’s privacy settings and their system while upholding the integrity of the online gaming community by using machine learning and artificial intelligence.

Keywords: Game Security, Threat & Cheat Detection, Aimbot Detection

1. Introduction

1.1 Motivation and Goal to be Achieved

Online gaming has been very popular and has been becoming increasingly prevalent for several years. Players can play online games from virtually any location if they have an internet connection. This made games more accessible for everyone and increased the number of participants. First-person shooting (FPS) games are the most played online games for a very long time. Every year, a new group of players joins the large and devoted player base of a game. Cheating greatly affects the gaming experience in online first-person shooter games due to their intensely competitive and hectic character plays. Players can easily obtain open-source cheating programs such as CoSmos, CheatEngine, and Artmoney. For an excess of $500–1000 US dollars, cheating devices that are more sophisticated and challenging for a game's anti–cheat system to identify can be purchased. The money made by game developers may be impacted by cheating, particularly if it causes a decline in the player base or a poor performance history for the game.

Malware and cheats are fundamentally related and share many features. Because it is essentially an untrusted code that, like malware, seeks to go undetected and exploit data, since

`

they conceal and disguise undesired code from users, cheat designers must be proficient in programming.

Most modern anti-cheating programs are mostly ineffective and may compromise gamers' privacy. Anti-cheat solutions typically function at the kernel level of the player's system to obtain a more profound understanding of the gaming environment and deter cheating. Although crucial for detecting cheating effectively, this strategy carries substantial privacy problems. Players are generally obligated to provide comprehensive permissions to these anti-cheat technologies, granting them access to key portions of their operating system. With this level of access, anti-cheat systems can effectively observe and examine several elements of a player's device, such as system processes, files, and even network activities. Although these precautions are put in place to detect and prevent cheating, players may inadvertently disclose personal information throughout this procedure. The usage of anti-cheat technologies operating at the kernel level has sparked worries over possible data breaches, as the broad permissions granted could unintentionally jeopardise user privacy. Developers face a dilemma in finding a middle ground between effective cheat detection and protecting player privacy when working on anti-cheat technologies (Bakkes, Spronck, and van Lankveld, 2012). Considering this, the goal of this project is to explore the use of machine learning techniques for non-intrusive information using sandboxed techniques for detecting cheaters and hackers.

Due to ethical considerations, most published works concentrate on data mining and behavioral analysis while using an imagined dataset. This paper uses easily accessible game logs and publicly available video footage as data to build Visual Cheat Detection for AimBot, Wallhacks, and Speedhack Detection. The Algorithm uses 200 photos from the dataset to train the model using YOLOV8. It can perform real-time player and enemy detection along with capabilities to check if a player is performing a wallhack or not. Additionally, Player Aimbot detection can also be achieved with a technique mentioned in this research. This model's outcomes would assist in strengthening the standard anti-cheat software strategy by adding an additional set of security checks.

1.2. Research Question

What would be the impact of cheating behaviors and techniques evolving over time in online gaming, especially First Person Shooter (FPS) Games, and how can AI models be implemented and adapted to stay effective in detecting the different forms of cheating?

1.3. Novel Contributions

The study presents a specialised Visual Cheat identification Algorithm that is specifically developed for the purpose of recognising cheating behaviours, such as Aim Bot, Wallhack, and Speedhacks. This algorithm employs YOLOv8, a visual object identification technique. This algorithm improves the effectiveness of cheat detection by promptly identifying both players and adversaries in real-time, while also respecting the privacy preferences of the participants. This research distinguishes itself from prior studies by employing authentic data derived from accessible game logs, developer portal videos, and publicly available footage. This approach enhances the reliability of the cheat detection model, as it avoids the use of fake datasets for ethical reasons. The proposed method for detecting Aimbot utilises statistical analysis, namely z-scores, to identify unusual movements in pitch and yaw angles, hence enhancing precision and adaptability. To identify Speed Hack, the game server architecture incorporates tick rate monitoring, which employs pattern recognition and predefined thresholds to precisely ascertain the update frequency between clients and servers, thus providing a comprehensive method. The study defines a complete set of evaluation criteria, which encompass accuracy, precision, recall, and F1-score. These criteria are visually depicted using precision-recall curves and confusion

`

matrices, so facilitating a better comprehension of the model's capabilities.

1.4. Cheating and Its Impact

There are more gamers than ever in the world of online gaming owing to the rise in gaming influencers. Additionally, since the player count has increased, cheating in these kinds of games has become more common than it was.

A recent survey by Irdeto Survey[1]conducted in 2022 finds that 79% of developers worry that cheating will damage their game's ecosystem, while over 85% of players have encountered cheating and were forced to stop playing because they were unable to progress through the game's learning curve and were continually losing, which prevented them from becoming better.

Cheating in online games seriously compromises fair play, damages community trust, and tampers with the integrity of virtual worlds. It establishes an unfair playing field, which may irritate players who are acting legitimately and reduce their pleasure in the game. The negative effects are made even worse by the economic consequences, which involve the devaluation of in-game items and the reinvestment of developer resources towards the implementation of anti-cheat systems (Chambers *et al.,* 2005). Cheating in esports has the potential to undermine the fairness of tournaments and harm the existence of the esports industry. Cheaters may face legal repercussions, such as bans and suspensions, and if cheating becomes so prevalent that it drives players away, the game's long-term survival may be in danger. In the end, cheating not only ruins the playing experience but also has serious repercussions for the online gaming community, business, and general well-being.

2. Literature Review and Background

Several researchers have dealt with several issues related to cheating and its negative impact on online gaming while putting different strategies into practice to stop cheating. However, in comparison to other areas, the amount of research performed on this topic is less. The main cause of this is that most gaming firms employ proprietary anti-cheat software. By withholding most of the information, they make sure the software is tough to hack into and is kept secret to maintain secrecy. Furthermore, any study done on this kind of software is regarded as proprietary by private enterprises.

2.1. Data Mining Methods for Cheat Detection

Data mining algorithms are used (Philbert, 2018)to look for specific cheating software in a host machine's memory. Data mining techniques have demonstrated their accuracy in protecting systems and providing solutions for experts to appropriately assess when needed. The same ideas should hold true for cheat-related executables even though this study does not address harmful executables. But new exploits are continuously being developed, and a game seems to be protected by little more than an impractical data mining technique. Furthermore, because the software for anti-cheat has access to the entire memory of the host, utilizing this as a stand- alone solution would raise confidentiality concerns.

2.2. Cheat Detection Methods based on Vision

Deep neural networks (DNN) had been used for the detection of cheating (Jonnalagadda *et al.,* 2021).This study offered a fresh method for spotting online game cheating. The technique uses a vision-based algorithm to examine the visual data generated by the game and identify instances of cheating. For accuracy and robustness, the system uses DNN to classify the game

[1] https://irdeto.com/news/irdeto-survey-reveals-increased-worry-of-cheating-and-tampering-highlighting-the- need-for-adequate-game-protection

`

as real or forged and uses aggressive defense and confidence estimates. The technology outperforms current anti-cheat methods in accuracy and hostile attack defense. The paper also discusses the method's limitations, such as false negatives and positives with high variance, and the necessity for a lot of information that is labeled for the model to get trained.

(Zhang, 2021) also mentions AI object detection. Gaming cheats like Aimbots and wall hackers affect gamers, according to a study. It proposes visual anti-cheat detection using recorded gameplay of a popular online first-person shooter. Unfortunately, time and resource constraints prevented real-time cheat detection in this research. Visual object recognition was used instead.

2.3. Player Behavioural-Based Cheat Detection

A novel technique for identifying wall-hacking is proposed by(Laurens *et al.,* 2007). The technique used four characteristics and player behavior analysis to forecast the likelihood of cheating by a player. The article shows promising results in differentiating between players who are cheating and those who are not, which sets it apart from other commercialized methods already in use. However, there isn't a strong mathematical basis for the measurements that are employed, thus further work will eventually be needed to examine, enhance, and create new, more complex metrics. Moreover, the method might not work against other forms of cheating, and the benefits of a standard architecture might be compromised by other player collusion tools.

Another machine learning technique developed by (Willman, 2020)identified popular first-person shooting game cheaters. This covered an introduction to online game cheating research, a machine learning guide, and the building and evaluation of a cheater detection neural network. The findings suggest the method can detect game cheaters. The next phase of this research provided a perspective into methodologies for machine learning and how to improve model performance with more data. This work's disadvantages included using limited data for testing and the model's accuracy and only evaluating the algorithm on one game.

(Tian, Brooke, and Bosser, 2016) presented their use of the suggested architecture to differentiate between biased and fair players while also providing an active method as opposed to a passive one to defend the fairness of the games. Even if they weren't robust, they demonstrated that behavioral models could be implemented as an anti-cheating strategy that could help outlaw cheaters.

(Alayed, Frangoudes and Neuman, 2013) offered a behavior-based method for identifying cheaters in online games using machine learning classifiers. The logs were stored on the server, where an attribute generator was processed, and an information analyzer examined them. The authors completed several trials using various kinds of models for detection, and they achieved good accuracy results. The developers' cheat detection policy sets the threshold's quantity. It can range from low to high, but it shouldn't be too harsh or else many seasoned, moral gamers will be called cheaters. The authors also suggest that more research be done, to improve the algorithms. The limitations include the exclusive implementation to multiplayer games and its requirement for game log access, which isn't always possible.

2.4. Machine Learning Enabled Detection of Bot amongst Authentic Players

(Mitterhofer *et al.*, 2009) suggested a novel method of bot exposure that circumvents the issues associated with client-side solutions by relying solely on the server end evaluation of a player's behavior. In order to achieve this, the study makes use of a feature inherent in bots: an executable script that controls the bots and executes a precise set of actions automatically that are repeated frequently. The method determines two crucial factors that aid in the identification of bots: the frequency with which a character traverses the path and the degree of recurrence of the route. This approach's drawback is that it was tested on a minor scale, with a bot that

`

simply followed specific paths, and the assessments for both detections considered no gameplay activity.

(Dunham, 2020) highlighted the use of machine learning for Counter Strike Global Offensive cheat detection. To counteract them, it also addresses several cheats that can be noticed and exploited, as well as machine learning solutions. It made use of the Nvidia deep neural network library, CUDA & cuDNN. It looks for and finds a solution to the Counter-Strike cheat detection issue using machine learning. The technique is based on two main parts: the creation of metrics and characteristics that may be used to accurately categorize cheating, and a network structure that emulates the way people naturally analyse video footage (repeating neural networks). The restrictions include the need for network architecture to be modified to appropriately take data format into account and the identification and development of specific cheating metrics.

(Kotkov *et al.*, 2018) conducted a comprehensive study on online gaming bot identification, focusing on applications that enable resource storage. It highlights knowledge gaps and offers a classification of bot detection methods. The authors also propose directions for future research, including utilizing a wider variety of feature spaces and expanding the types of machine learning algorithms used for detection. Two of the paper's drawbacks include its focus on an online game and the small number of feature regions used in the research under examination. Moreover, the study falls short of offering a thorough assessment of the effectiveness of several bot detection systems. Future work will involve creating more bot identification tools and supplementing the literature analysis of relevant studies.

2.5. Comparison Table

Title	**Author**	**Main Focus**	**Cheat Focus**	**Methodology & Drawbacks**
Data Mining Methods for Cheat Detection	Philbert, A. (2018)	Applying data mining tools to identify instances of cheating	Identification of illegitimate cheat software within the memory of the host machine	Utilisation of data mining techniques that may raise confidentiality concerns.
Cheat Detection Methods Based on Vision	Jonnalagadda, A. et al. (2021)	Utilising deep neural networks to detect cheating	Identification of fraudulent events with vision-based algorithms	A vision-based approach utilising deep neural networks (DNN) is employed for the classification of actual objects in a game.
	Zhang, Q. (2021)	Utilising artificial intelligence to prevent cheating.	Utilising recorded gameplay of a widely played online first-person shooter to discover and identify instances of cheating through visual analysis.	Aim was to create a real-time visual object recognition system. Resource limits limited object detection accuracy.
Player Behavioural-Based Cheat Detection	Laurens et al. (2007)	Detection of cheating by study of player behaviour	Utilising player behaviour analysis to predict the probability of cheating from an aimbot perspective.	Methods including the analysis of player behaviour and attributes, which has drawbacks

`

	Willman (2020)	Utilising machine learning to detect and identify individuals who engage in cheating behaviour	Constructing and assessing a neural network designed for detecting cheating behaviour that derives from wallhacking and strafing.	The machine learning technique has limits in terms of data testing and analysing the algorithm, as it is only applied to one game.
	Tian, Brooke, and Bosser (2016)	Applying behavioural models to distinguish between players who exhibit bias and fair play.	Offering a proactive approach to safeguard the integrity of games from individuals engaging in cheating behaviour.	Utilising behavioural models as a countermeasure against cheating.
	Alayed, Frangoudes, and Neuman (2013)	Utilising machine learning classifiers to detect and identify individuals who engage in cheating behaviour in online games.	Algorithmic approach for identifying individuals who engage in cheating behaviours from illicit powerups by unfair means.	Experiments have been conducted using several models to detect issues, however these experiments have been limited to multiplayer games and require access to game logs.
Machine Learning Enabled Detection of Bot amongst Players	Mitterhofer et al. (2009)	Identifying automated programmes by analysing player actions on the server side	Detecting automated accounts by the recognition of predetermined and repetitive behaviours that derives from cheating.	The evaluation of player behaviour in games such as World of Warcraft is done on the server-side, but it has limits in terms of scalability and assessing gameplay activity.
	Dunham (2020)	Applying machine learning techniques to detect cheating in Counter Strike Global Offensive.	Analysing various cheating methods and leveraging machine learning techniques to counteract them.	The methodology involves utilising the Nvidia deep neural network library, CUDA, and cuDNN to change the network architecture and create cheating measures.
	Kotkov et al. (2018)	An extensive investigation into the identification of bots in	Identifying areas with insufficient knowledge and suggesting potential avenues for future investigation	The work solves the bot detection confusion by focusing primarily on a game. However, thorough assessment of

`

		online gaming.		other bot detectors was not provided.

3. Research Methodology

The research methodology functions as a methodical structure that directs the entirety of the research process, guaranteeing a structured and dependable strategy to tackle the research objectives.

3.1. Methodology

A Structured, practical, and flexible approach was taken into consideration to perform this research. Initially, the business understanding and the future impact of the research were considered along with the existing practices that are already present. During the business understanding phase, the problem is defined, and goals and objectives are outlined. Comprehension of items and objects that are detrimental to cheat detection and exploring the dataset are the main goals of the data understanding phase. The dataset is then pre-processed in the data preparation stage which involves cleaning, filtering, and labeling the data to make it appropriate for model training. This includes activities like resizing images, classifying objects, and dealing with missing data. Next, the dataset is divided into test, validation, and training sets. Yolo (You Only Look Once) (Ultralytics YOLOv8 Docs, 2023) is the algorithm of choice for modeling, with versions and customization as required to be considered. The prepared dataset is used to train the model. Metrics like precision and recall are used to evaluate the model's performance during evaluation and objects are detected as per the model which was trained.

3.2. Language and Environment

Python's large libraries for image processing and machine learning, together with its adaptability, make it an essential component. Python is frequently used for preparing data, training models, deploying them, and evaluating them. The popular Python distribution environment Anaconda makes development even easier by offering a complete package and dependency management environment. Installing libraries like OpenCV, NumPy, Ultralytics and torchvision [2] are all necessary for creating and executing the YOLO algorithm which is made easier by the package manager Conda.

3.3 Game Architecture & Components

The architecture of a game is of paramount significance in influencing the overall gaming experience, namely in areas such as user identification, game tracking, and inculcating anti-cheat mechanisms.

3.3.1 Game Architecture Flowchart

User identification and account security are the first steps of a first-person shooter game. Users will be asked to login again or register if their credentials are invalid. The architectural framework tracks player achievements and statistical data across games, improving personalised experiences. Matchmaking consent requires players to choose to record their data

[2] https://docs.ultralytics.com/

`

for multiplayer activity, distinguishing between human and bot-controlled matches. A user's client system's kernel-level anti-cheat features are seamlessly integrated, identifying and flagging suspicious player behaviour for further investigation. Players who cheat are banned from the game, ensuring fair play.

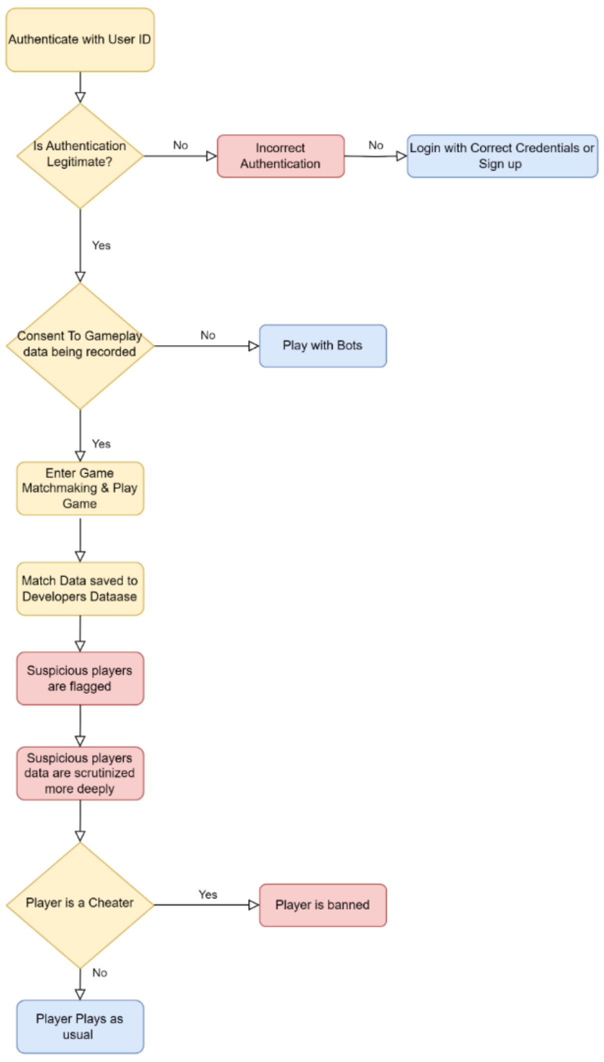

Authenticate with User ID
Is Authentication Legitimate?
No
Incorrect Authentication
No
Login with Correct Credentials or Sign up
Yes
Consent To Gameplay data being recorded
No
Play with Bots
Yes
Enter Game Matchmaking & Play Game
Match Data saved to Developers Dataase
Suspicious players are flagged
Suspicious players data are scrutinized more deeply
Player is a Cheater
Yes
Player is banned
No
Player Plays as usual

Figure 1: Game Flow for Anti-Cheats

The architectural design of a system for playing games is dependent upon the utilization of the client-server model, which collaboratively influences and shapes the complete gaming experience. The client-server framework functions as an essential structure that enables efficient data synchronization between the devices of players and the game server. The synchronization process guarantees a consistent and integrated game environment for all individuals involved in a multiplayer game. Furthermore, it assumes an important role in the implementation of anti-cheating protocols, actively identifying and addressing questionable behaviors to uphold fairness in the gaming environment. The scalability of the model is of equal importance, as it enables game servers to adapt to fluctuations in player numbers and effectively handle real-time interactions, assuring a seamless and prompt experience as the player base grows. The client-server model and tick rate collectively establish the fundamental framework for a dynamic and pleasurable online gaming environment.

`

The primary function of the server is to consistently update the game state at a high frequency, which is achieved by maintaining a specific tick rate. The Tick Rate parameter determines the frequency at which the server changes its game state within a given time frame.

3.3.2 Tick Rate

The tick rate[3], commonly known as the server's tick rate, is a crucial characteristic within the field of first-person online games. It has a considerable impact on the overall gameplay experience and the competitive dynamics involved. The tick rate is a metric that quantifies the rate at which the game's server updates and processes information. It is often measured in ticks per second, denoted as Hz.

A Tick can also be seen as a single capture in the frame of the game where tick(0) is the initial snapshot of the game and tick(N) gives the final snap of the game. It must be noted that the tick rate at the client and the server's end would always differ from each other due to network latency (Alkhalifa, 2016). On average, there would be around a difference of 5 ticks ideally. However, the increased latency also causes a lot of discrepancies in the game with significant lag which causes performance loss and frustration among the players.

3.3.3 Recoil

Pulling the trigger in a game creates recoil[4] and is expected to change the course of a player's gun from where they are aiming to off-target slightly for first-person shooter games. This can be measured by the length of time the gun fires and decreases the player's accuracy. This strategy is thought to filter out cheating players and retain the fair players in the game. Counter-Strike 2 is the game that this thesis focuses on. Every weapon in this game features a unique set of random recoil patterns. It's been observed that even extremely good players are unable to make up for recoil, which is a noticeable distinction between cheaters and gamers as cheaters can easily make up for recoil.

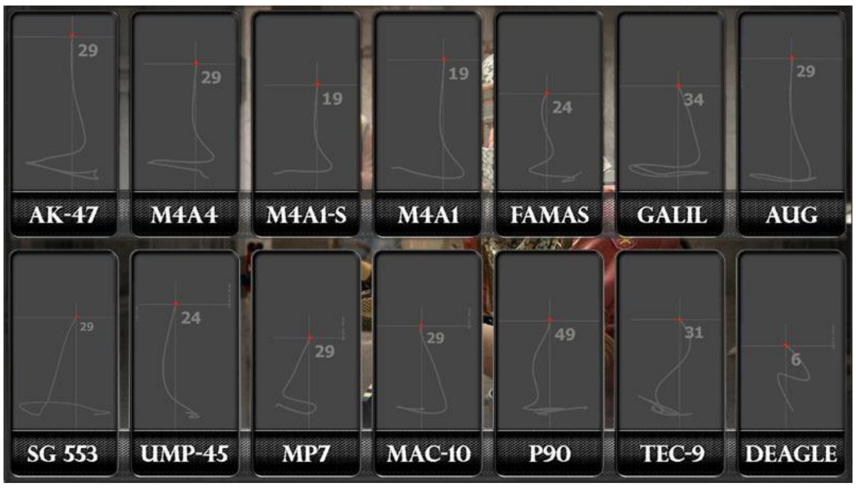


Figure 2: Recoil (source: Recoil | Counter-Strike Wiki | Fandom (no date) Counter-Strike Wiki)

[3] https://dotesports.com/counter-strike/news/how-does-tick-rate-work-in-counter-strike-2
[4] https://counterstrike.fandom.com/wiki/Recoil

`

3.3.4 Cheats

A. Aimbots

Aimbots[5] are a prevalent form of fraudulent scripts or software utilized in first-person shooter (FPS) video games. These hacks grant players an unfair advantage through the automated targeting and aiming of adversaries, frequently achieving remarkable levels of precision.

B. Wallhacks

Wallhack[6] is a prevalent form of cheating that is frequently implemented in first-person shooter (FPS) video games by leveraging this hack, players gain the ability to observe surfaces such as walls, terrain, and other obstacles present in the game environment.

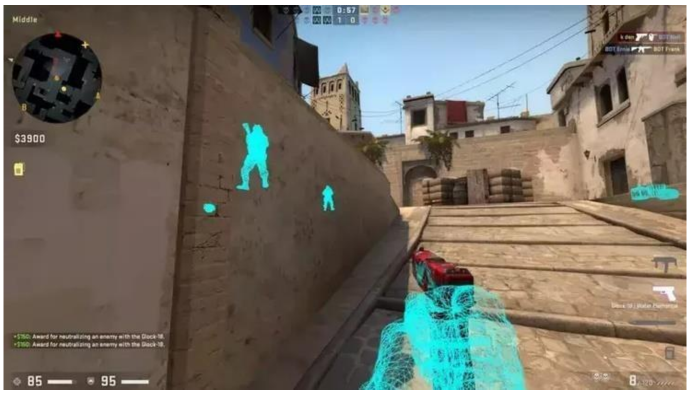


Figure 3: Wallhack (source: Capper. Tom (2022) - What Are Wallhacks and How Do They Work? - PrivateCheatz.)

C. Speedhacks

Particularly common in online multiplayer games, speed hackers [7]constitute a form of cheating in which players gain an unfair advantage by modifying the character's movement speed.

D. Map Hacks

Map hacks are a form of cheating that occurs in video games, specifically multiplayer online games, wherein players exploit or manipulate the game's minimap or map to obtain an unfair advantage.

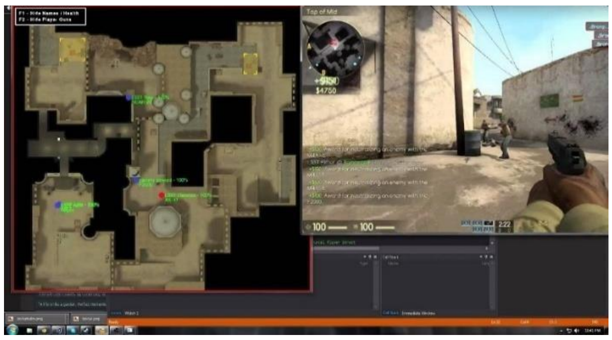


Figure 4: Maphack (source: CSGO External Maphack V0.2 - Obsta)

[5] https://chatbotsjournal.com/what-are-aimbots-and-how-do-they-work-4936794e5453
[6] https://www.privatecheatz.com/what-are-wallhacks-and-how-do-they-work/
[7] https://combatarms.fandom.com/wiki/Speed_Hack

`

3.2. Dataset

The original dataset utilized in the study was taken from the video game Counter Strike 2. There are 200 images in all that comprise the dataset. Yolov8, which stands for "You Only Look Once," is used to develop the AI model. YOLO is the product of Ultralytics, which is an open-source platform on vision AI algorithms, the most well-known vision AI in the world that is used for object detection and visualization. MakeSense is used to annotate every image according to our requirements and produce a legitimate data set. The annotation of what AI should search for was done using the following labels.

Label	**Description**
Gun	Detects the player gun for recoil and perspective of the player.
Wall	Detect walls in the game to check if a player can see through walls and kill enemies.
Enemy	Detects enemy for speed hacks and wall hacks.

The YOLO Algorithm [8]needs data to be structured in a specific way for it to read, train, and test data(Diwan, Anirudh and Tembhurne, 2023). There are two primary folders namely images and labels with each folder containing 2 sub-folders namely train and val. They are described as follows:

Training Subset (train): Most of the dataset, including 200 photos, was contained in the training subset (also known as "train"). 'Images' and 'labels,' two subfolders, were created from the 'train' subset to facilitate access and use during training.

Validation Subset (Val): This subset, which consists of 20 photos, was set aside especially for validation. The 'val' subgroup was similarly divided into 'images' and 'labels' subfolders.

3.3 Yolo and Its Variants

YOLO V8 has 5 variants namely n,s,m,l & x. Each model exhibits a unique level of precision in detecting and performing tasks. It can be observed how YOLO performs much faster and more accurately than its predecessors.

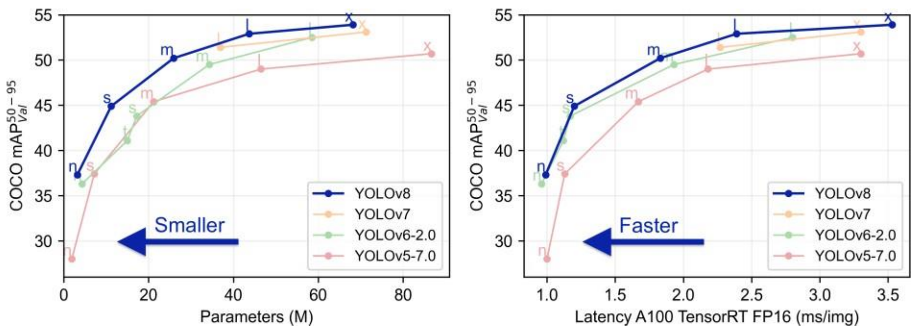


Figure 5: YOLO Variants (source: Home - Ultralytics YOLOv8 Docs (2023))

The graph provides the correlation between the quantity of parameters and the delay in several YOLOv5-YOLOv8 models using an A100 TensorRT FP16 GPU. Latency is quantified as the

[8] : https://github.com/ultralytics/ultralytics

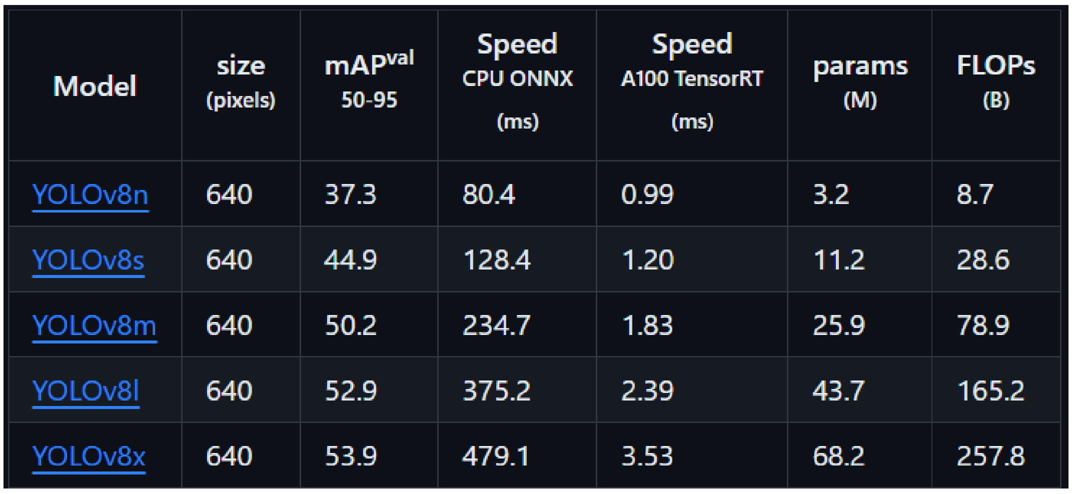
`

duration in milliseconds required to process each individual picture (ms/img). A decrease in latency results in an increase in the speed at which the model can process images.

The graph illustrates a positive correlation between the size of the model and the number of parameters it possesses, as well as a negative correlation between the size of the model and its speed. This is because larger models require a greater number of calculations in order to process an image. Nevertheless, the graph indicates that YOLOv8 models exhibit higher speed compared to YOLOv7 models, despite having a higher number of parameters. YOLOv8 incorporates some architectural enhancements that enhance its efficiency. Here, The COCO (Common Objects in Context) dataset is used which is a vast compilation of annotated images and movies used for tasks such as object detection, segmentation, and instance tracking.

The graph also displays the COCO MAP50-95 Val values, which serve as a metric for evaluating the object detection precision of the models. A model's accuracy increases as the COCO MAP50-95 Val score rises. The graph provides a clear trade-off between accuracy and latency. Greater in size, larger models exhibit enhanced accuracy, but at the cost of reduced speed. In general, the graph indicates that YOLOv8 outperforms prior iterations of YOLO in terms of both speed and accuracy in object recognition. Despite having more parameters, it is more efficient than YOLOv7.

The analysis of YOLO models reveals that the v8 models demonstrate a notable enhancement in mAP (mean Average Precision) ranging from +4 to +9 in contrast to the v5 models. Remarkably, this gain is achieved while keeping a comparable runtime. It is worth noting that v8m and v8l exhibit superior performance in terms of both mean average precision (mAP) and speed when compared to v5l and v5x (YOLO V5) Furthermore, the v8n model emerges as the highest-performing lightweight variant, demonstrating exceptional proficiency in terms of both accuracy and speed when compared to the other YOLO versions under consideration.



Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 6: YOLOV8 Version Comparison (source: Home - Ultralytics YOLOv8 Docs (2023))

The Yolov8m model was selected for research testing due to its high speed and suitability for detecting a limited number of objects.

`

4. Design Specifications

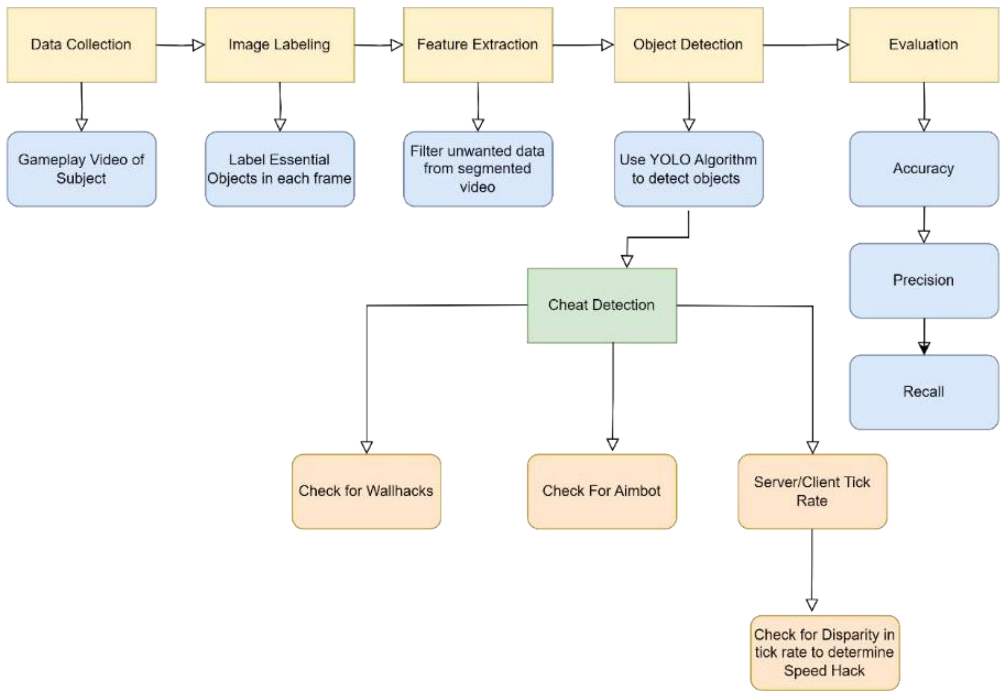

Data Collection
Image Labeling
Feature Extraction
Object Detection
Evaluation
Gameplay Video of Subject
Label Essential Objects in each frame
Filter unwanted data from segmented video
Use YOLO Algorithm to detect objects
Accuracy
Precision
Recall
Cheat Detection
Check for Wallhacks
Check For Aimbot
Server/Client Tick Rate
Check for Disparity in tick rate to determine Speed Hack

Figure 7: Implementation Architecture

The figure above illustrates the system architecture for the research based on anomaly detection in first-person shooter games. Commencing with the gathering of data from publicly accessible sources on the internet, the subsequent process of labelling images guarantees the construction of a thoroughly annotated dataset, which serves as an essential base for training and evaluation. The feature extraction phase selectively removes extraneous data, refining the dataset to emphasise essential features for cheat detection. Utilising the YOLO (You Only Look Once) method in the object detection phase improves effectiveness by allowing immediate detection of cheating techniques such as wallhacks and aimbots inside individual frames. In addition, the inclusion of server/client tick rate monitoring introduces a level of complexity, primarily aimed at detecting speed hacks by examining irregularities in gaming interactions between players and the game. The evaluation measures, including accuracy, precision, and recall, are crucial for assessing the effectiveness of the model.

5. Implementation

5.1. Dataset Creation

During the preliminary stage of the implementation, the dataset was carefully selected and organized to enhance the training process of the YOLO algorithm. The dataset was produced by selecting 200 photos from publicly accessible gameplay footage of the game. To ensure enough vital data within the dataset, the unnecessary images from the data were filtered out. The images were subsequently labeled using the open-source application makesense.ai, effectively identifying the specific areas of interest such as player, player gun, enemy, and walls for the purpose of anomaly identification.

`

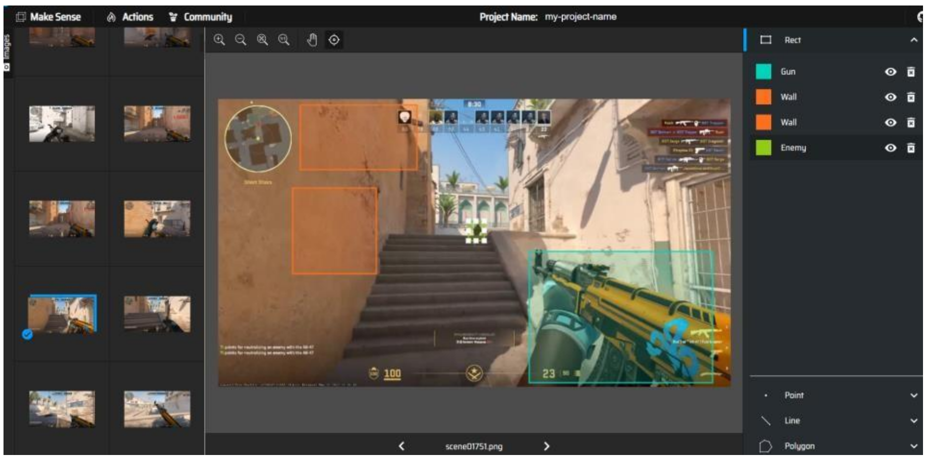
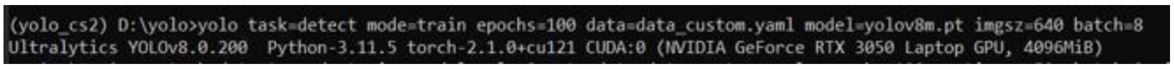


Figure 8: Image Labelling

5.2. Model Training

The YOLO method for anomaly detection was put into effect in an Anaconda environment. The purpose of selecting this environment was to speed up training and development and the ease of an integrated terminal. After training for about 100 epochs, the YOLO model was able to iteratively improve its comprehension of the distinct qualities and features seen in the game and the objects that were labelled were being detected effectively.

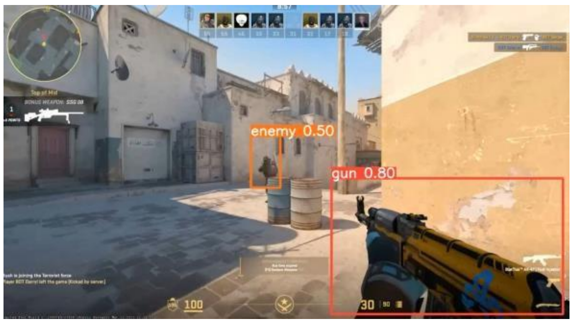


```
(yolo_cs2) D:\yolo>yolo task=detect mode=train epochs=100 data=data_custom.yaml model=yolov8m.pt imgsz=640 batch=8
Ultralytics YOLOv8.0.200  Python-3.11.5 torch-2.1.0+cu121 CUDA:0 (NVIDIA GeForce RTX 3050 Laptop GPU, 4096MiB)
```

Figure 9: YOLO Execution Command

5.3. Object Detection

The YOLO model's trained weights were used in conjunction with the Anaconda environment and Ultralytics, a framework created by YOLO, to achieve the smooth capture of in-game objects. This assisted in the identification of important in-game elements like the Player, the Player's gun, Enemies, and Walls. The model successfully identifies enemies, walls, and player guns as anticipated.



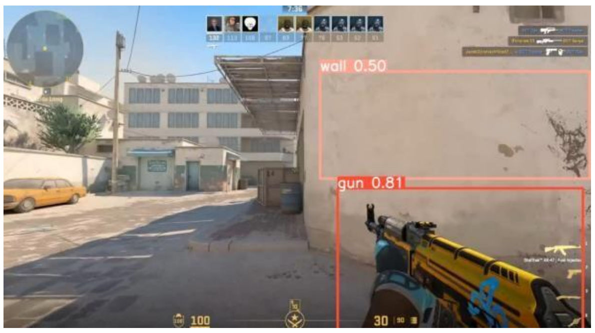


Figure 10: Object Detection (Enemy and Gun)

Figure 11: Object Detection (Wall and Gun)

`

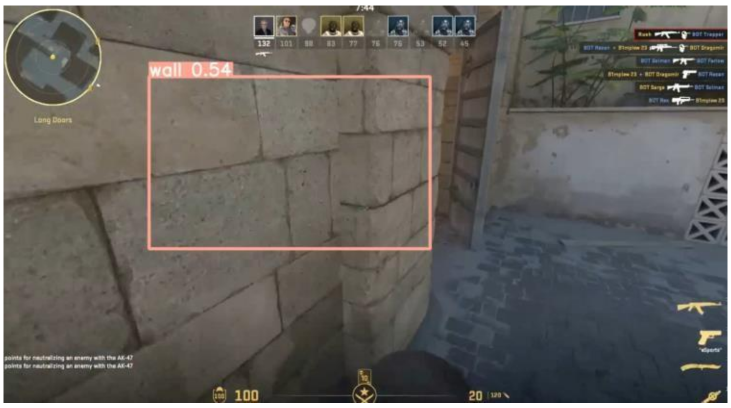


Figure 12: Object Detection (Wall)

5.4. Proposed Solution for Aimbot Detection

To determine the accurate pitch and yaw required to target an opponent, the AimBot generates vectors for both the player and the enemy, thereby establishing a difference vector between them. Trigonometry is used to compute the pitch angle by taking into account the distance between players and the disparity in their heights. The yaw angle is determined by calculating the disparities in x and y displacements. Maintaining these angles within specific limits is essential to prevent detection by anti-cheat programs.

The aimbot detection system's primary component is the evaluation of in-game parameters. The technique is concerned with two angles in particular: pitch and yaw. The game mentions[9] that yaw (i.e., side-to-side angles) is between $-180 \leq \text{yaw} \leq 180$, while pitch is defined as down-to-up angles limited between -89 and 89 degrees. Any deviation from these bounds suggests that the targeting behavior may be automated or artificial or in conclusion the player is using an aimbot.

Players' motions are evaluated by the aimbot detection code, which considers their pitch and yaw angles as input. When a player crosses the pitch and yaw boundaries, it is suspected that aimbots are being used. It is possible for the system to precisely identify and detect such abnormalities by collecting and monitoring player data. The Following Application Takes Pitch and Yaw data as input and determines whether a player is using an Aimbot or not.

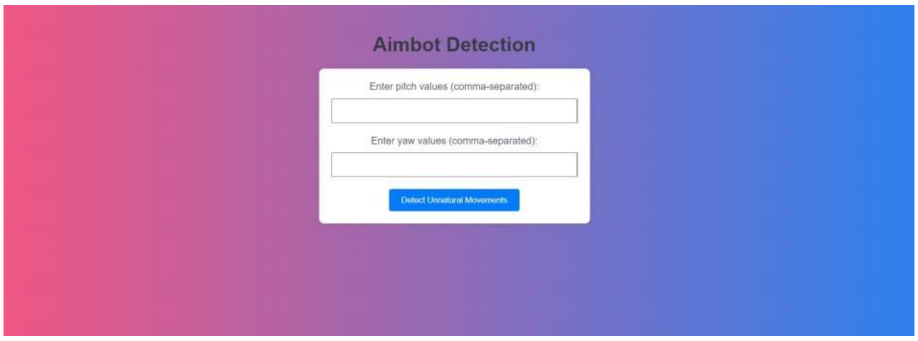

Aimbot Detection
Enter pitch values (comma-separated):
Enter yaw values (comma-separated):
Detect Unnatural Movements

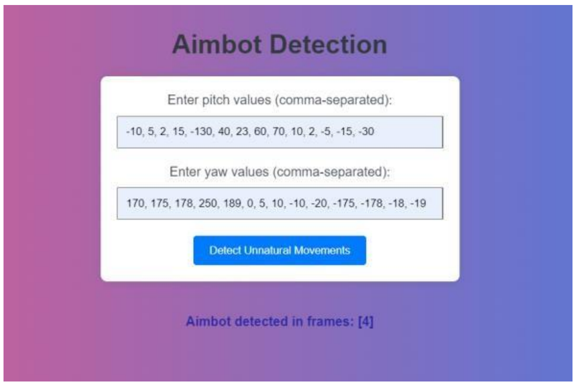
Figure 13: Application for Aimbot Detection

The given solution use z-scores to detect the usage of aimbot by analysing pitch and yaw data. It identifies frames when the z-scores for these variables exceed a predetermined threshold. Z-

[9] https://developer.valvesoftware.com/wiki/WisePYR

`

scores, which represent departures from the average, are employed to identify abnormal fluctuations.


Aimbot Detection

Enter pitch values (comma-separated):

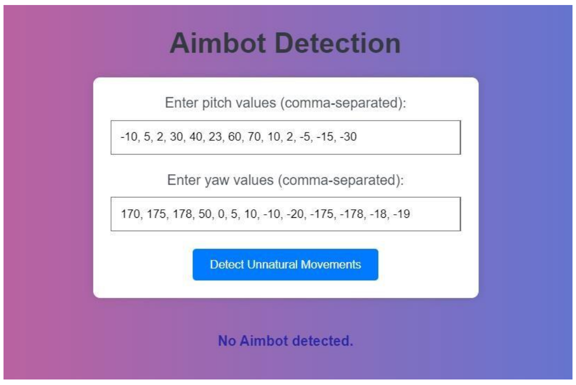
-10, 5, 2, 15, -130, 40, 23, 60, 70, 10, 2, -5, -15, -30

Enter yaw values (comma-separated):

170, 175, 178, 250, 189, 0, 5, 10, -10, -20, -175, -178, -18, -19

Aimbot detected in frames: [4]

Figure 14: Aimbot Detected


Aimbot Detection

Enter pitch values (comma-separated):

-10, 5, 2, 30, 40, 23, 60, 70, 10, 2, -5, -15, -30

Enter yaw values (comma-separated):

170, 175, 178, 50, 0, 5, 10, -10, -20, -175, -178, -18, -19

No Aimbot detected.

Figure 15: Aimbot Not Detected.

If the z-score of a frame surpasses the zscore_threshold, it indicates an abnormal change in targeting. The code subsequently detects and displays frames containing identified anomalous movements in a given frame. If The Z-scores predict that data is normal with no unnatural or abnormal spikes in each frame data while also considering the set Pitch/Yaw data, the application would determine that the aimbot is not being used.

The code improves the detection of the aimbot by conducting statistical analysis on the pitch and yaw data. By utilizing z-scores and implementing a dynamic threshold, it detects frames exhibiting abnormal movements, offering a precise and transparent approach. This technique provides flexibility, assisting in the ongoing enhancement of anti-cheat mechanisms by identifying aimbot behaviour precisely and enabling thorough study. Continuous cooperation within the anti-cheat community is essential for maintaining an advantage against ever-changing cheating techniques.

5.5. Proposed Solution for Speed Hack Detection

In online first-person shooter (FPS) games, both the client and the server run on distinct time intervals because of network latency. Suppose the server is currently at tick 120, while the client is lagging at tick 116 due to delays in transmitting data. The latency between the server and client in this example is 4 ticks. To enhance the fluidity of the gaming experience, first-person shooter (FPS) games employ a method known as client-side prediction. This functionality enables the user to anticipate and react to actions on their device without having to wait for validation from the server, hence diminishing the apparent delay for participants.

This proposal presents a comprehensive strategy to strengthen the game's anti-cheat system by incorporating tick rate analysis to identify and prevent speed hacks. The solution entails incorporating a tick rate monitoring mechanism into the game server architecture to record the frequencies at which updates occur between clients and servers. Games frequently need rendering 60 frames per second, which necessitates using the rendering function every 16.6 milliseconds.

`

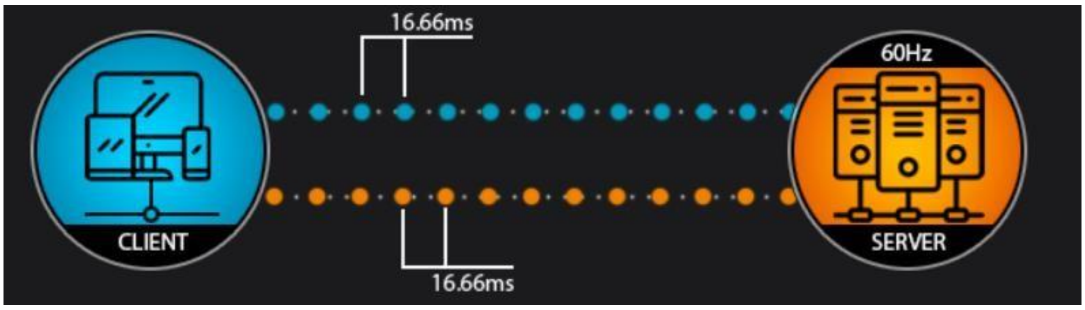


Figure 16: Client/Server Tick Rate (source: Ticks and update rates | Unity Multiplayer Networking)

The system's objective is to detect anomalies in player movements by establishing a standard tick rate and creating algorithms to estimate expected movement speeds. Pattern recognition techniques in conjunction with object detection and server-side checks will analyse these patterns in more depth, and predefined thresholds will activate flags to prompt additional research. Additionally, Speedhacks can be visually distinguished as well without the need for having external anti-cheat factors, however, sometimes the enhanced speed might also be due to ping disparity which might wrongly accuse a fair player of using speedhacks. Hence, such methods with the amalgamation of server-client side checks along with machine learning techniques are essential for distinguishing between a valid speed hack cheat and a false alert.

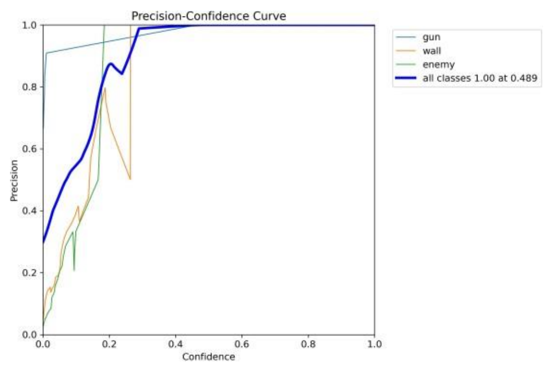
6. Evaluation

The model was trained using 220 photos with three labels, which were cleaned for a satisfactory outcome, with 200 images in the Train folder and 20 in the Val folder. Every output from the same data set is shown below.

6.1. Model Evaluation

Accuracy is chosen as the performance metric together with precision, recall, and f1-score for each label class to calculate the performance of this model.

6.1.1. P Curve & R Curve



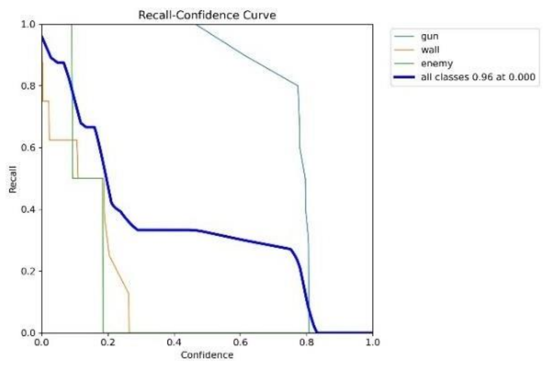


Figure 17: P Curve

Figure 18: R Curve

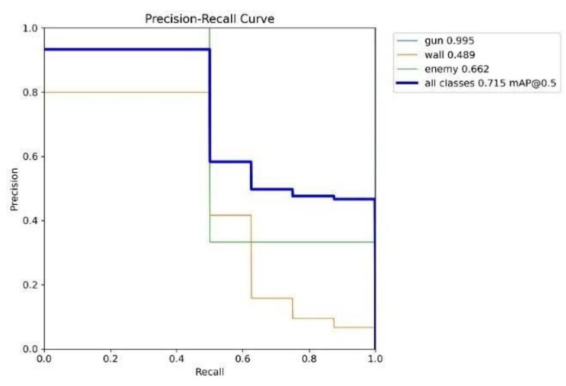
The accuracy of a model's positive predictions is determined by its precision. It is the proportion of actual positive results to all positive predictions. On the contrary, recall measures a model's capacity to catch each relevant occurrence of a positive class. It is the proportion of real positives to all genuine positives.

`

The Values achieved in the P curve & R curve graphs can be said to be between 0.5- 1.0 which pertains to the model achieving a reasonable balance between precision and recall considering the scale of a small dataset.

6.1.2. PR Curve & F1 Curve

The Precision-Recall (PR) curve is a visual depiction that shows the shifting relationship between precision and recall at various decision thresholds in a classification based on a binary model. Precision is a measure of the accuracy of positive predictions. It counts the proportion of projected positive instances that are positive, with a focus on minimizing false positives. Recall measures the model's capacity to accurately identify true positive instances, emphasising its sensitivity to false negatives. The numbers displayed on this curve hover around 71%, signifying a commendable equilibrium between precision and recall. Enemy accuracy is low due to the small size of its occurrence in each frame, while wall accuracy is low due to the detection dependence on the player purposefully pointing towards the wall, which was less in the dataset that was taken to perform this test and training.



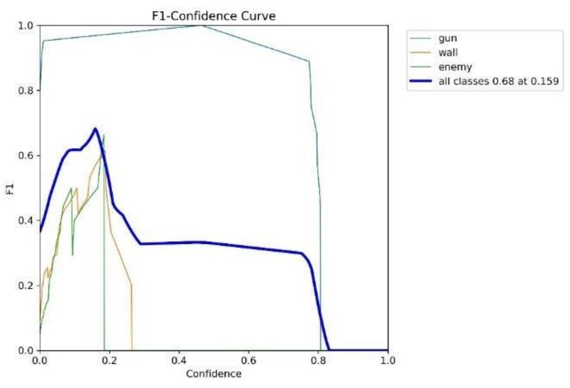


Figure 19: PR Curve

Figure 20: F1 Curve

The F1 score is a unified measure that combines precision and recall, offering an optimal balance between the two. The F1 score is a condensed statistic that combines both the precision and recall metrics, similar to the PR curve. The F1 curve can be observed to have most values above 50% with the highest value being 68% at a confidence level of 0.159 indicate that the model is operating well in general.

6.1.3. Confusion Matrix

The confusion matrix below aids in visualizing the algorithm's performance or accuracy. The confusion matrix provides a concise summary of the algorithm's performance.

`

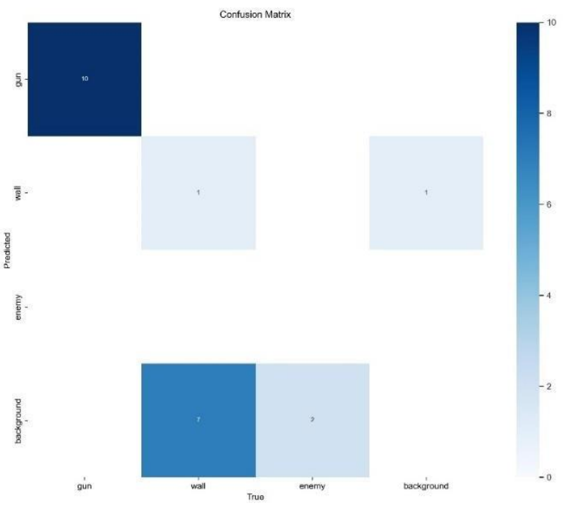


Figure 21: Confusion Matrix

It displays distinct values for each of the three labels. The matrix validates the recurring trend that has already been observed - exact values for each category, hence strengthening the idea that the algorithm consistently performs adequately.

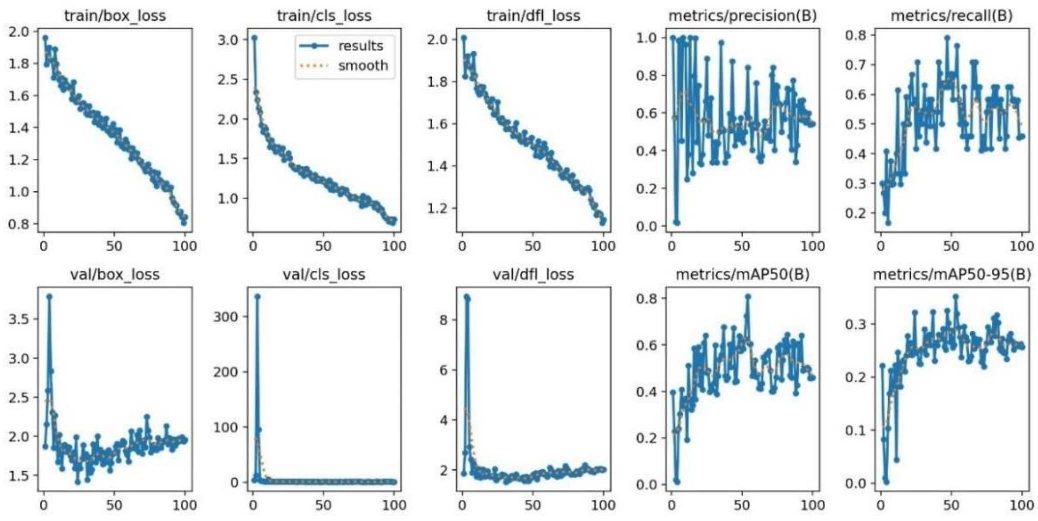


Figure 22:Results of Training

6.2. Discussion

The training results provided below clearly demonstrate the AI's ability to reliably detect various labels. Although the model is often accurate, it can occasionally produce incorrect positive results.

`

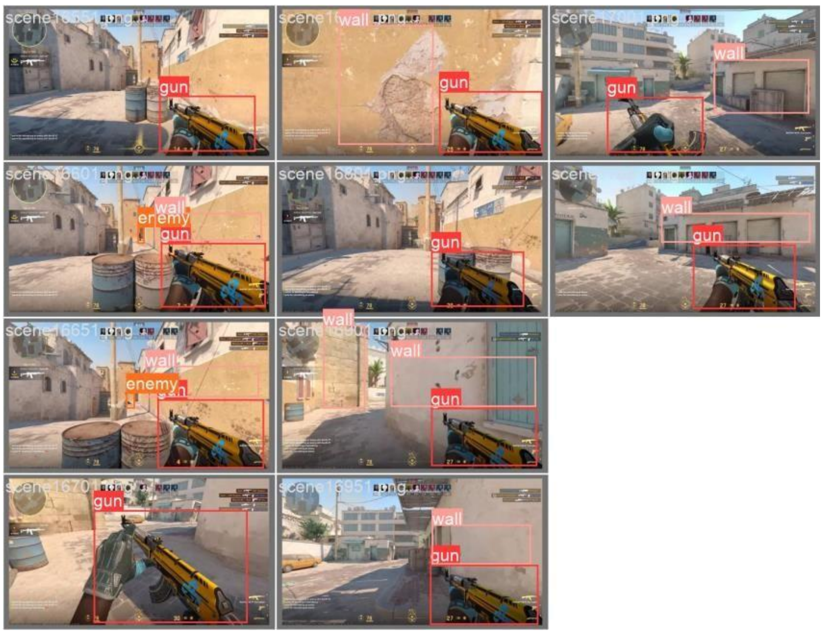


Figure 23: Post Training Object Detection

6.2.1. Comparison with Existing Models

Unlike traditional anti-cheat solutions that depend on rule-based and signature-based approaches, our cheat detection system utilises machine learning with YOLOv8, providing clear technical benefits. This system, which has been trained on a broad dataset from Counter-Strike 2, has good precision. The model's flexibility in detecting and adapting to changes in real-time makes it a proactive solution for evolving cheating strategies, surpassing rule-based systems that struggle to keep up.

6.2.2. Implications for Game Developers

The findings have significant implications for game creators, as they introduce an entirely new approach to cheat detection using machine learning. The model's effectiveness, combined with its commitment to protecting user privacy, makes it an excellent resource for developers seeking to enhance their security procedures. By harnessing the effectiveness of object detection algorithms, game developers have the potential to transform the gaming experience through the integration of sophisticated cheat detection technologies. This measure not only ensures the safety of the player community but also secures the financial interests of developers by promoting fair play and maintaining long-term player involvement.

6.2.3. User Acceptance and Adoption

When considering user approval, the level of technical clarity in the detection process becomes an essential factor. Users are more inclined to accept a cheat detection method that is transparent, simply understandable, and reduces instances of false positives. The complex details of this technique, specifically the utilization of z-scores to reduce the occurrence of incorrect positive results, enhance the transparency and sophistication of our system. It is crucial to find a middle ground between preventing cheating and maximizing user satisfaction to successfully use these advancements, which aligns with the user-centered approach of this research.

`

6.2.4. Practical Implication

When considering the practical implementation of this cheat detection system, an important decision arises regarding its architectural framework: if to implement it within the client-server architecture or completely transition to a server-based model, utilising cloud gaming to eliminate any latency. Nevertheless, this change is not devoid of its difficulties. Although technological developments and improvements in transmission speeds hold promise, effectively tackling these difficulties still needs further research and effort in the future. The choice between the client-server model and cloud gaming adds an important aspect to the implementation approach, requiring a delicate equilibrium between immediate responsiveness and the effectiveness of cheat detection.

7. Conclusion And Future Scope

The study introduces an effective method to distinguish cheaters from legitimate gamers by utilizing artificial intelligence. It specifically focuses on identifying common cheating techniques such as Wallhack, Aimbot, and Speed hacks. The study also proposes the use of behavioral analysis to detect cheaters or ensure fair competition by maintaining skill parity. Although there is currently limited information available on open-source anti-cheat engines, the study asserts that the integration of AI can eradicate cheaters and organizations involved in the sale of cheats.

The deployment of AI-driven anti-cheat systems encounters challenges such as the necessity for instant identification demands, the requirement for a cloud-centric framework, and the complexities of behavioural analysis. Despite these difficulties, AI-powered anti-cheat systems can be implemented on many gaming platforms and genres, providing a cohesive resolution to the widespread problem of cheating.

Future goals involve integrating AI data to detect cheating in real-time, migrating to a cloud-based infrastructure, and applying behavioural analysis to identify gamers exhibiting suspicious behaviour. Additionally, a dataset consisting of thousands of images can also be used to train any given object detection model for even sharper and more accurate results. The existing approach for detecting aimbots is highly proficient in analyzing statistical data related to pitch and yaw. However, its applications are limited due to proprietary constraints. Collaborations between anti-cheat developers and game publishers offer prospective possibilities for improvements in the future.

Furthermore, the paper does not exhibit the ability to identify speed hacks because of the limits imposed by the client-server architecture. To address this, the future vision entails incorporating this feature like the suggested aimbot detection system. Resolving unique issues will allow for immediate tracking of player behaviours, while continuous investigation could integrate machine learning for flexible identification in conjunction with support from game developers.

Eventually, this study emphasises the capacity of artificial intelligence (AI) to completely transform the field of anti-cheating measures. Additional research is required to tackle the issues and considerations mentioned before, but we hold a positive outlook on the development of anti-cheat technology.

`

8. References

Alayed, H., Frangoudes, F. and Neuman, C. (2013) ‘Behavioral-based cheating detection in online first person shooters using machine learning techniques’, *IEEE Conference on Computatonal Intelligence and Games, CIG* [Preprint]. Available at: https://doi.org/10.1109/CIG.2013.6633617 . (Accessed: 10 October 2023).

Alkhalifa, S. (2016) *Machine Learning and Anti-Cheating in FPS Games*. Available at: https://www.researchgate.net/publication/308785899_Machine_Learning_and_Anti-Cheating_in_FPS_Games (Accessed: 12 October 2023).

Bakkes, S.C.J., Spronck, P.H.M. and van Lankveld, G. (2012) ‘Player behavioural modelling for video games’, *Entertainment Computing*, 3(3), pp. 71–79. Available at: https://doi.org/10.1016/J.ENTCOM.2011.12.001 . (Accessed: 13 October 2023).

Chambers, C. *et al.* (2005) ‘Mitigating Information Exposure to Cheaters in Real-Time Strategy Games’. (Accessed: 18 October 2023).

Chen, B. Di and Maheswaran, M. (2004) ‘A cheat controlled protocol for centralized online multiplayer games’, *Proceedings of the ACM SIGCOMM Workshop on Network and System Support for Games, NetGames’04*, pp. 139–143. Available at: https://doi.org/10.1145/1016540.1016554 . (Accessed: 18 October 2023).

Chen, K.T., Pao, H.K.K. and Chang, H.C. (2008) ‘Game bot identification based on manifold learning’, *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, NetGames’08*, pp. 21–26. Available at: https://doi.org/10.1145/1517494.1517498 . (Accessed: 28 September 2023).

Daniel Webb, S. and Soh, S. (2008) ‘A survey on network game cheats and P2P solutions’, *Australian Journal of Intelligent Information Processing Systems*, 9(4), pp. 34–43. Available at: http://find.curtin.edu.au/staff (Accessed: 17 October 2023).

Diwan, T., Anirudh, G. and Tembhurne, J. V. (2023) ‘Object detection using YOLO: challenges, architectural successors, datasets and applications’, *Multimedia Tools and Applications*, 82(6), pp. 9243–9275. Available at: https://doi.org/10.1007/S11042-022-13644-Y/TABLES/7 . (Accessed: 15 September 2023).

Dunham, H. (2020) ‘Cheat Detection using Machine Learning within Counter-Strike: Cheat Detection using Machine Learning within Counter-Strike: Global Offensive Global Offensive’. Available at: https://openworks.wooster.edu/independentstudy (Accessed: 30 July 2023).

Gorman, B. *et al.* (2006) ‘Bayesian imitation of human behavior in interactive computer games’, *Proceedings - International Conference on Pattern Recognition*, 1, pp. 1244–1247. Available at: https://doi.org/10.1109/ICPR.2006.317 . (Accessed: 15 September 2023).

Home - Ultralytics YOLOv8 Docs (2023) *Ultralytics YOLOv8 Docs*. Available at: https://docs.ultralytics.com/ (Accessed: 15 September 2023).

Jonnalagadda, A. *et al.* (2021) ‘Robust Vision-Based Cheat Detection in Competitive Gaming’, *Proc. ACM Comput. Graph. Interact. Tech*, 4(1), p. 18. Available at: https://doi.org/10.1145/3451259 . (Accessed: 15 September 2023).

`

Kang, A.R. *et al.* (2013) ‘Online game bot detection based on party-play log analysis’, *Computers and Mathematics with Applications*, 65(9), pp. 1384–1395. Available at: https://doi.org/10.1016/J.CAMWA.2012.01.034 . (Accessed: 10 September 2023).

Kotkov, D. *et al.* (2018) ‘Gaming Bot Detection: A Systematic Literature Review’. Available at: https://doi.org/10.1007/978-3-030-04648-4 . (Accessed: 16 October 2023).

Laurens, P. *et al.* (2007) ‘A Novel Approach to the Detection of Cheating in Multiplayer Online Games’. https://research.tees.ac.uk/ws/files/6438470/111786.pdf (Accessed: 02 September 2023).

McGraw, G. (2009) ‘Cheating massively distributed systems’. Available at: http://www.cigital.com (Accessed: 12 October 2023).

Mitterhofer, S. *et al.* (2009) ‘Server-side bot detection in massively multiplayer online games’, *IEEE Security and Privacy*, 7(3), pp. 29–36. Available at: https://doi.org/10.1109/MSP.2009.78 (Accessed: 12 September 2023).

Philbert, A. (2018) *Detecting Cheating in Computer Games using Data Mining Methods, iMedPub Journals*. Available at: https://www.imedpub.com/articles/detecting-cheating-in-computer-games-using-data-mining-methods.pdf (Accessed: 30 July 2023).

Tian, H., Brooke, P.J. and Bosser, A.-G. (2016) ‘Behaviour-based Cheat Detection in Multiplayer Games with Event-B’ : https://core.ac.uk/reader/322332700 (Accessed: 19 September 2023)

Willman, M. (2020) ‘Machine Learning to identify cheaters in online games’. Available at: https://umu.diva-portal.org/smash/get/diva2:1431282/FULLTEXT01.pdf (Accessed: 03 October 2023).

Zhang, Q. (2021) ‘Improvement of Online Game Anti-Cheat System based on Deep Learning’, *Proceedings - 2021 2nd International Conference on Information Science and Education, ICISE-IE 2021*, pp. 652–655. Available at: https://doi.org/10.1109/ICISE-IE53922.2021.00153 . (Accessed: 12 September 2023).