Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230589

Machine learning anti-cheating algorithm and a test against computer vision aimbot

Gefei Yang[1, 5], Yuqi Zhang[2], Feihong Liu[3], Zishuo Gao[4]

[1]College of Engineering, School of Computing, University of Utah, Salt Lake City, 84112, USA
[2]Champaign Central High School, Chapaign, 61822, USA
[3]Crean Lutheran High School, Irvine, 92618, USA
[4]Computer science, University of Bristol, Bristol, BS8 1QU, UK
Gefei Yang, Yuqi Zhang, Feihong Liu, and Zishuo Gao contributed equally to this work and should be considered co-first authors.

[5]u1271767@utah.edu

Abstract. In this article, we introduced a recurrent neural network (RNN) serves as a cheating detection method in CS: GO against traditional and AI cheating methods. Cheating methods today are growingly stronger, making traditional anti-cheating methods less and less helpful. The CV AimBot, which does not include any illegal operation but only reading the game footage, has made all traditional anti-cheating methods de facto useless. We developed a CV AimBot from scratch and used it to generate the sample data. We gathered datasets from traditional AimBot and newly invented CV AimBot and run it through an RNN. Our RNN shows high accuracy against both traditional and CV AimBot. It also shows that introducing data from CV AimBot with significantly lower the accuracy of the result from RNN.

Keywords: Machine Learning, Anti-Cheating, Aimbot, RNN.

1. Introduction

As first-person shooter games become more popular and competitive, an increasing number of players are using cheating methods to win in games, with even some of the most advanced or professional players using these methods to gain advantages over their rivals.

One of the most popular games in the world of first-person shooter games is Counter Strike: Global Offense (CS: GO). However, CS: GO has been troubled by the issue of cheating for quite some time. Arguably the most popular cheat in first person shooter games such as CSGO is the aim bot, which controls a player's aim commonly by using injectors into a game's memory while running [1]. To combat this, traditional anti-cheating methods consist of monitoring the use of injectors and blocking such loopholes, or using human supervision to review the replays of games where a player is reported as cheating [1].

In the last few years, a new type of AimBot has shown up. This new method is using a computer vision model to detect the human body, or even certain body parts in the games' image. Being fundamentally different from the traditional methods of cheating, this new method only requires the

DOI: 10.54254/2755-2721/6/20230589

access to the games' image, or in actual running, the games' streaming videos. The access to game information this method requires is essentially the same as streaming or recording software such as Open Broadcaster Software (OBS) or NVIDIA ShadowPlay, which means they are not detectable by any traditional anti-cheating software or algorithm.

However, many of these cheating detection tasks, against traditional and newly emerged, could be done by machine learning, sometimes with better accuracy in the results [2, 3]. While some advanced players could achieve the same aiming in 150 milliseconds of reaction time, the speed and consistency of the AimBot still have an edge over the human players, thus those are the factors we are considering when we extract datasets and train the Machine Learning model [4].

Chapter two will review the variables required in analyzing and determining cheating activities, chapter three will introduce our approach to train the machine learning model,

chapter four will generalize the CV AimBot and our implementation, and lastly chapter five will conclude our paper with possible future improvements.

2. Variables in determining cheating activities

Data from every game is stored as ticks on servers with most commonly 64 or 128 ticks per seconds [4], which is the servers' tick rate. However, we will only pick out the ticks that are right before a kill has been made. To gather the desired data, we have to scrape through a demo file and localize the ticks and information we need. The position, movement, and aiming angle of the players before a kill has been made has been shown to be good determining factors. After noticing an enemy, human players have a reaction time that is usually around 120ms~200ms, which could be clearly reflected in the ticks. In contrast, the cheating software would either aim right when the opponent appears in sight, or even aim at the opponent that is behind an object or still yet to be in sight. Because the cheating software manipulates the local gaming memory, it is very easy for them to know the exact position of other players. The specific variables that we collected from the ticks are *pitch, yaw*, the position of the player in x, y, z, and velocity vector with respect to each direction as vec0, vec1, vec2.[4] To avoid any single event that could result in false determination, we also used the scraper to trace the number of kills, deaths, assists, headshots, wallbangs (shots that hit opponents behind a wall), and total shots. These variables were used to calculate the percentage of headshots and wall bangs in order to make a comparison between the consistency between human players and cheating software [4].

The choice of using CS: GO is not made arbitrarily. One problem this method encountered while executing is the scarcity of available data sources. In fact, CS: GO is the only popular game which enables a third party to get direct access to the in-game statistics. This could be a result of game companies' protection on their data and privacy, but it also makes it difficult for outside researchers to study upon those issues. CS: GO has a built-in tool to download a convertible *.dem* file which is called the *demo file*. The demo file contains every server information that is useful to reproduce a single game exactly. It can be downloaded directly from the game and can be shared by any file transferring method. CS: GO also offers a partially usable official tool: *demoinfogo* to convert a demo file into a txt file. The tool is no longer supported by Valve officially and will not compile with up-to-date libraries. Certain modifications might be required depending on the machine and compiler.

3. Training the Machine Learning models

After collecting 124 datasets from regular gameplay and 110 datasets from a cheater's gameplay, we are able to train our machine learning model. RNNs, in our situation, offer an edge compared to other types of deep learning models because it offers a means to store past information. This is intuitive because a single frame cannot determine if a player is cheating. Rather, if several frames are considered together, we are able to get more information about the speed and behavior of the player within a range of time. The addition of continuity allows us to better compare between the behavior of a genuine player and the behavior of a cheater.

Below is an example of the data we scraped from the demo files using the *demoinfogo*. This format is then scraped and converted into numerical form using a tool named Demo_scrape.py from the *Cheat*

DOI: 10.54254/2755-2721/6/20230589

Detection using Machine Learning within Counter-Strike: Global Offensive. The tool was kindly provided by Mr. Harry Dunham, the author of the paper mentioned earlier, and later modified to meet our requirements. A typical collection of information in the demo file of a certain entity in one server tick is shown as the following image.

```
Entity Delta update: name:EOiAøEE¶iA§¥„Eü„E≥„Eà„Eº, id:2, class:40, serial:781 {
Table: DT_CSPlayer
Field: 0, m_flSimulationTime = 64
Field: 1, m_nTickBase = 15165
Field: 21, m_angEyeAngles[1] = 76.948242
Field: 72, m_flCycle = 0.434143
Field: 93, m_flCycle = 0.075256
Field: 1331, m_flThirdpersonRecoil = -0.000000
}
Entity Delta update: name:ÁÅ´„ÅÆÁ•ûÁ•ûÊ•Q_ÂÜÜËàû, id:3, class:40, serial:839 {
Table: DT_CSPlayer
Field: 0, m_flSimulationTime = 63
Field: 1, m_nTickBase = 15164
Field: 2, m_vecOrigin = 1443.528320, 2812.955078
Field: 3, m_vecOrigin[2] = 127.901382
Field: 4, m_vecVelocity[0] = 195.602127
Field: 5, m_vecVelocity[1] = 76.418655
Field: 7, m_vecOrigin = 1443.528320, 2812.955078
Field: 8, m_vecOrigin[2] = 127.901382
Field: 24, m_flGroundAccelLinearFracLastTime = 236.921875
Field: 58, m_flCycle = 0.968353
Field: 59, m_flPlaybackRate = 0.007386
Field: 93, m_flCycle = 0.918182
Field: 103, m_flWeight = 0.282353
}
```

Figure 1. Extracted Demo File.

Although we want our RNN to consider new information, as well as information that's already seen, this can run the risk of the vanishing gradient problem. Essentially, the greater the number of layers, the more untrainable the network becomes. Thus, LSTM is ultimately decided to be the best model of RNN, because it can be thought of as "as a network that has a parallel conveyor belt that layers and nodes can output values to in order to access at a later point during the learning process." [5].

Following the LSTM layer, a simple dense layer is implemented with a sigmoid activation function to output the confidence/probability of a player cheating. To do gradient descent, the ADAM algorithm is used, which employs a general stochastic gradient descent.[6] To ensure the generalizability of our model, dropout is implemented to combat overfitting, which is when a model has high accuracy with the data it is trained on, but performs poorly with new data.

For the method of training and testing datasets, we used the cross training and testing module from the Keras Tensorflow Framework, which it randomly splits the dataset into big group of testing data and small group of testing data, and go for certain rounds and calculated accuracy for each round and the final accuracy, which is the average of the accuracy from each round of the training.

4. Visual cheats

With the development of algorithms and AI technology, there are more and more ways of cheating that break away from previous methods.

This can be seen, for example, in online chess tournaments. Because of the advent of artificial intelligence, a chess player can play against a human on a computer while running an AI program on his phone. The AI imitates his opponent's move and provides moves that will be used on the computer. Obviously, there is no such thing as a kernel-level countermeasure for this type of cheating (as this is effectively the equivalent of having a personal support move in the back). Although chess is not a traditionally online game, it demonstrates that there exists methods that bypass past forms of anticheats. This is true for first person shooter games, and there is now a class of plug-ins that can effectively bypass existing forms of detection [7].

Visual cheats are a different type of cheat from other types on the market, as they do not alter or read the memory data of the game itself, nor the kernel, and the usual anti-cheat detection methods are in general incapable of againsting them. The CV AimBot, for example, is essentially a deep learning-based aid, which will detect human body or body parts such as head in certain areas on screen and moves the

DOI: 10.54254/2755-2721/6/20230589

cursor to a specific body part or at least helps to keep the cursor stays on the body [7]. Because it does not modify the game files or data, it is very difficult to detect thoroughly. Some aggressive anti-cheating methods will read through essencial files and processes while running the game. However, this is not only controversial, but the plug-in can be connected to another computer via a capture card, which is a common strategy for streamers to prevent performance loss caused by capturing and streaming, making detection even harder.

It is also difficult for third party researchers and players to tell whether game companies have developed methods against it. Game companies' anti-cheating strategies are usually cautious to prevent any false conviction because it can be potentially more damaging.

Valve Anti Cheat (VAC) for example, is used in CS: GO but is known for its high accuracy and low efficiency [8]. A typical cheater will only be convicted after about one month. Even if the anti-cheating methods are effective, they tend to not publish it to prevent future advancement on cheating methods.

The first step is capturing live game footage. With the help of OBS or a video capture card, the user can easily record the live output of the game and immediately send it to a locally running program or another computer. Once the footage has been captured, the pre-trained CV algorithms on the computer such as YOLOv3 can be used for fast humanoid target detection [9].

Once the AI identifies an enemy on screen, it can quickly calculate the relative position between the cursor and any body part as required such as head or torso, which in most FPS will take the most damage, then move the cursor to it with or without the need of players' intervention. In addition to fast aiming, the AI can also adjust the cursor by pulling back the mouse just like a real person against the recoil patterns of certain weapons. If designed properly, it can also conduct automatic firing to compensate for the possibly slow reacting time of players.

Since those kinds of behaviors such as auto headshot hack are not achieved by modifying system memory or files, but by collecting on-screen information like the human eye or recording softwares, all movements of the visual AI can be mistakenly recognized by

today's traditional anti-cheat detection systems as being performed by a real person, therefore can evade all monitoring [7].

Figure 2. Custom CV AimBot.

We deployed a CV AimBot using a self-customized program with multiple options for CV config and weight such as YOLOv3 and YOLOv4. This CV AimBotis able to accurately identify and extract

DOI: 10.54254/2755-2721/6/20230589

information of characters in the game image within milliseconds and give the exact pixel coordinates of the center of body or body parts. It can then move the cursor to that pixel with optional offsets. After an afternoon of testing and optimizing, the AI was able to, in a competition, outperform as high as a Gold Nova I Ranking player without using any gadget but only shooting, which means its performance is better than at least 40% percent of all players, without triggering any of the game's anti-cheating methods. After using it for a month, none of the accounts we tested on has been banned by VAC.

One fact to notice is that our CV AimBot is only using CV models which are open and free for downloading. With customized models trained by in-game images or pose estimation models, better results have been developed and validated [7].

5. Results and future improvements

As from this result of cross training and testing, the RNN for cheating analysis is having up to 67.08% and averagely 60.98% accuracy of catching cheaters in the testing files. It is also obvious to observe that after training, there is significant growth in accuracy and drop in loss. The result shows that with limited computing power and data sets, this RNN is still capable of figuring out the cheater with high accuracy. However, as mentioned before, this result is not accurate enough for any game company to give a conviction. There exist other neural networks with similar structure but better accuracy. It is possible that our RNN's relatively low accuracy is the result of randomly including demo files from using CV AimBot. Since the actions from CV AimBot are more similar to human players', their patterns are harder for neural networks to learn and detect. However, the growing accuracy shows that with larger data sets and more computing power to support more complex structures of neural networks, higher and even commercially acceptable accuracy is achievable.

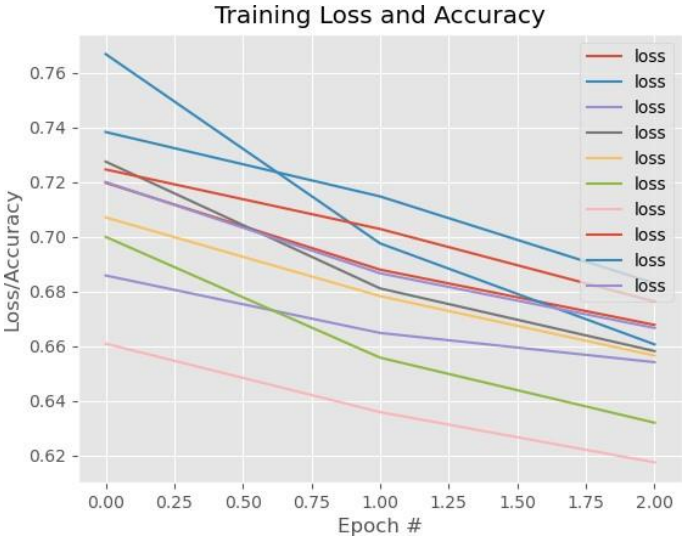

Figure 3. Loss across batches.

As seen in the results, we can see that those new types of cheating methods challenged the traditional anti-cheating methods, even our machine learning base anti-cheating. Thus, we have to analyze the behavior of those visual based cheating methods further, capture its key features that differ from human players, and add additional modules into our neural network that could analyze those characteristics. It is also promising that with larger data sets, these methods will perform better when detecting CV AimBot. At the same time, it is foreseeable that CV AimBot will evolve to disguise themselves to perform more similar to human players. This threat cannot be underestimated. However, machine learning based anti-cheat would have less limitations compared to traditional anti-cheat with its ability to analyze almost complete gaming data and learn the cheating behaviors of characters from it. As whenever a new cheating method comes out, it will have its own characteristics which most likely would

DOI: 10.54254/2755-2721/6/20230589

be able to be captured by the machine learning model through rigorous training. Thus, we believe that the future machine learning based anti-cheating is very promising, and we will keep improving this module.

Figure 4. Accuracy across batches.

6. Conclusion

Cheating in video games has been a problem for gamer communities, game companies, and anti-cheating researchers since it exists. Traditional anti-cheating methods are effective against cheating method in the past but have shown less and less capability against newly those which are newly invented. Our method of using RNN against traditional AimBot and CV AimBot shows high accuracy against both and can potentially reach higher accuracy if given more computing power and datasets. However, the threat of CV AimBot and other AI based cheating method cannot be underestimated. Our result indicates that introducing CV AimBot into the datasets will lower the accuracy of the result. The fact that most AI based cheating methods does not require illegal operation makes it the most important problem form online video games. The necessity of more powerful anti-cheating methods is more urgent than ever. Further investigations and research are needed to produce methods with higher accuracy to be deployed for commercial usage, but artificial neural networks have shown it capability outperforming traditional anti-cheating methods.

Acknowledgement

Gefei Yang, Yuqi Zhang, Feihong Liu, and Zishuo Gao contributed equally to this work and should be considered co-first authors.

References

[1] Panicos Karkallis, Jorge Blasco, Guillermo Suarez-Tangil, Sergio Pastrana,Detection video-game injectors exchanged in game cheating communities, 2021

[2] Nvidia Capture SDK. NVIDIA Developer. (2021, December 9). Retrieved September 24, 2022, from https://developer.nvidia.com/capture-sdk

[3] Aditya Jonnalagadda, Luri Frosio, Seth Schneider, Morgan McGuire, Joohwan Kim, Robust Vision -Based Cheat Detection in Competitive Gaming, 2021

[4] Harry Dunham, Cheat Detection using Machine Learning within Counter-Strike Global Offensive, 2020

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[6] Kingma, D. , & Ba, J. . (2014). Adam: a method for stochastic optimization. Computer Science.

DOI: 10.54254/2755-2721/6/20230589

[7] Kanervisto, A. , Kinnunen, T. , & Hautamki, V. . (2022). Gan-aimbots: using machine learning for cheating in first person shooters.

[8] Valve Anti-Cheat (VAC) system. Steam Support. (n.d.). Retrieved September 24, 2022, from https://help.steampowered.com/en/faqs/view/571A-97DA-70E9-FF74

[9] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.