DeepCheat: A Convolutional, Adaptive, and Generalizable Video Game Anti-Cheat

Jón B. Salenger[1]

[1]McDonogh School, 8600 McDonogh Rd, Owings Mills, MD 21117

***Abstract*—The video game market is one of the largest in the world, and it is growing. Trailing this growth in the video game industry has been E-Sports. E-Sports have struggled to be recognized as a legitimate form of competition, but the amount of money swimming around professional video-gaming has started to turn E-Sports mainstream. The rise in E-sport's legitimacy has manufactured a need for a system to prevent cheating in online communities. This paper proposes a workflow for an automatically adaptive and generalizable video-game anti-cheat that trains Convolutional Neural Networks (CNNs) to remove cheating players from game servers. The system described in this paper can use human moderation to quickly train a CNN to recognize new forms of cheating as they arise. DeepCheat was trained on a custom dataset of 168 images that contained only two values, player position and the location of an attack. After training, DeepCheat achieved an accuracy rate of 82% and demonstrated the ability to learn over time. These results suggest that DeepCheat is a viable, data-efficient, generalizable, adaptable, and cost-effective solution to curb cheating in online games.**

I. Introduction

E-sports have become a legitimate form of competition over the past decade. Although they do not garner the same viewership numbers as traditional sporting events, the video-gaming market totals over $94 billion in the US alone.[1] Furthermore, E-sports are important to foreign relations as they become leading culture exports from countries like Japan and South Korea.[2] E-sports may not yet be viewed as real "sports" by the mainstream, but the industry is large and growing. As stakes and prize-pools rise, people have begun to develop increasingly complex ways of cheating in digital competition. This has necessitated the development of evermore accurate, powerful, and intrusive systems for ensuring fair competition between those who complete online. Traditional anti-cheat systems often increase in cost and development time as they become progressively more complex. DeepCheat is not a stand-in for these traditional systems but can help mitigate the harm caused by cheaters in the time between the creation of a new form of cheating and the publication of a patch to remove the cheaters.

Normally, a game's anti-cheat must be updated manually as a reaction to the creation and dissemination of a new form of cheating. This system requires a team of people who can be summoned at any point to patch out a cheating method before it ruins a game's competitive queues (many games maintain an online ranking of top-players decided by their performance in so-called "competitive queues"). Moreover, game companies are required to hire and maintain a plethora of moderators who can manually punish players who cheat and are not detected by the game's auto-moderation system. This process is expensive and slow, requires hiring numerous people and modifying a game's source code when the need arises. Additionally, once a patch is made the development team must decide to thoroughly test the new changes (increasing turn-around time and allowing more players to cheat) or hastily release the new patch, possibly breaking the game for many people who are not cheating, an expensive mistake.

As a potential solution, this paper proposes that the moderators can be incorporated in a training loop for a Convolutional Neural Network (CNN), which processes generated images of player performance. The CNN can then learn adaptively as moderators identify new forms of cheating. This approach provides numerous advantages over requiring the moderating team to describe a new exploit to a development team and then waiting for the development team to finish a patch. The main advantages are adaptability and cost as it is no longer required for game companies to hold many software engineers on call.

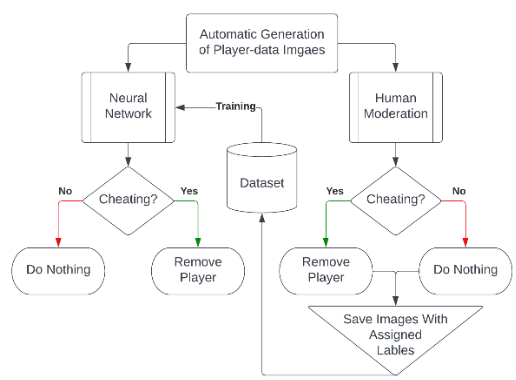

Figure 1. A diagram of the proposed DeepCheat workflow.

II. Methods

168 images (referred to as a "player report") were compiled to form a custom dataset of player reports created from the gameplay of cheating and valid players. These images were assigned the labels "dirty" and "clean" respectively. Four different players helped to create this dataset. Two of the players were given cheating tools and the other two were not. They were paired so in each scenario a cheater was fighting a normal player, and data was collected on their fights. Cheating players were given the ability to run faster, hit farther, and automatically aim their cursor. The player data was collected

using a custom written plug-in for the game Minecraft, which used packets sent to a game server to analyze the actions of a player and compose a corresponding image. This approach requires no code to be downloaded or executed on the client side.

Player reports were constructed by polling player positions from the server side every 100ms and then plotting a line between the last known position and the current one. To feed the model extra information, a red dot was drawn in the location of every attack by the player. This continues for 30 seconds until a 98x98 image is drawn detailing changes in a player's position and the location of every attack that player made. The amount of data within the image was intentionally kept small, in order to demonstrate that this approach is generalizable and data efficient.

The choice of Minecraft as the test game does not reduce DeepCheat's generalizability as all major game engines use a cartesian coordinate system, Minecraft's game design just emphasizes this feature. Minecraft was also chosen for its third-party API, which allows players to interface directly with the game server, a necessity for creating this dataset. Minecraft is a game assembled by blocks of 1 cubic unit of volume making it easy to map player location on a 2D plane to an image. If the players are competing in a flat virtual "arena" of X by Y then the resultant image can be of size X by Y. In this case, the arena used to train the model from player data was 98 blocks by 98 blocks, so the resultant image was 98 pixels by 98 pixels. This means that image size, format, and resolution are all task-dependent even within the same game.

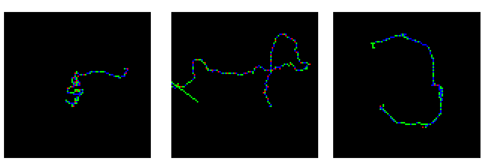

Figure 2. Examples of generated player reports, left and right are "clean" while the center image is "dirty."

Transfer learning techniques and the FastAI library were used to train a CNN based on a 34-layer residual network with pretrained weights. The standard 34-layer residual network (resnet-34) proved to be an extremely accurate and generalizable architecture for classifying images of all types.[3] All layers of the model were trained for 5 epochs, and only the last layer was trained for an additional 2 epochs. This helped the model learn the general format of what a DeepCheat image would look like on all the layers but prevented overfitting by keeping the first n-1 layers more general. The model was trained using a discriminative LR starting at 1e-3 and the Adam optimizer. We made use of the binary cross-entropy loss (also referred to as log-loss) function defined as:

$$-(y*\log(p)+(1-y)*\log(1-p)) \tag{1}$$

where y was a binary indicator, either 1 or 0, that reflects the correct label, and p was the predicted probability of the input being class y. This loss function penalized the model for being over or under confident by including the predicted probability in the equation. By optimizing the predicted probability to be as close to the correct label as possible the likelihood of costly false positives decreases.

III. Results and Discussion

After training, the model achieved a training loss of approximately .17 and a validation loss of approximately .47. The model's final accuracy was 82% (rounded to the nearest whole number). However, in practice this would mean that 82% of the time the model would successfully identify a cheater within *30 seconds*. As each successive player report is generated, the model will have a new chance to predict whether the player is cheating or not. DeepCheat's effectiveness in practice, therefore, would be much higher as the game continues.

The ability of DeepCheat to learn and improve based on the created dataset is shown by the decrease in the model's training and validation loss over time. The decrease in validation loss in conjunction with training loss implies that this task was learnable, and results were not the symptom of model memorization, a common critique of deep learning algorithms Ultimately, these results further suggest that the fundamental idea behind DeepCheat is viable. CNNs are able to learn on images composed of abstract renderings of player data.

IV. Conclusion

The goal of this project is to demonstrate that the DeepCheat workflow is both generalizable and accurate. The efficacy of DeepCheat as a Minecraft specific anti-cheat system could be improved by increasing the resolution of the image (by further subdividing each block into 4, 9, or n^2 others), displaying a wider variety of data to the model (e.g., player head position or player hit reach), or by decreasing the time between player position checks. However, as more data is made available to the model the less generalizable it becomes. For example, in some top-down style games, such as League of Legends, player head position is not a piece of data that would aid the model in classifying cheaters. Through the use of task-agnostic data and computationally efficient methods of data generation this paper demonstrates that the DeepCheat workflow is both generalizable and accurate.

Acknowledgment

I would like to acknowledge the research done by Gleb Esman at Splunk, a cybersecurity company, which gave me the idea to build DeepCheat.

References

[1] Clement, J. (2022, May 24). Video games industry in the U.S. 2022. Statista. Retrieved October 7, 2022, from https://www.statista.com/statistics/246892/value-of-the-video-game-market-in-the-us/

[2] Rea, S. C. (2020, May 25). Crafting stars: South Korean e-sports and the emergence of a digital gaming culture. Association for Asian Studies. Retrieved October 7, 2022, from https://www.asianstudies.org/publications/eaa/archives/crafting-stars-south-korean-e-sports-and-the-emergence-of-a-digital-gaming-culture/

[3] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. Procedings of the British Machine Vision Conference 2016, 1–1. https://doi.org/10.5244/c.30.87