Creating an Aimbot Using YOLOv8 to Understand and Prevent in Games

Joshua Richman

University of Michigan

richmajo@umich.edu

Abstract

Aimbots are bad for the gaming industry, since they take away the enjoyment of the other players. This paper will focus on creating an aimbot which uses YOLOv8 to train a weight based on public data from roboflow on Aim Labs Gridshot Ultimate gamemode. This aimbot is being created to understand how these types of aimbots are made so as to understand what needs to be looked at when creating anti-cheat software.

1. Introduction

The use of aimbots has been around for almost as long as games that could use them existed. Aimbots are bad for the gaming industry because if they become too prominent, players will stop playing and buying games. Many aimbots use the game client data to find the location to aim at, but there are also aimbots that use the visual output to aim at enemies [5].

I will be focusing on creating an aimbot that uses visual output to score as many points as possible in Aim Labs Gridshot Ultimate, to understand how they work, and the things game developers and anti-cheat developers should be looking for to prevent these aimbots.

Aim labs is a training game for other games such as Valorant or CSGO. While there is a scoreboard, all testing was done in an offline environment, so as not to tamper with any legitimate scores, since this is meant to understand how this kind of cheating works.

To do this, I will be training an object detection algorithm using YOLOv8. You Only Look Once, or YOLO, is a real time object detection system known for being able to quickly classify objects. I decided on YOLOv8 because it is one of the newest and fastest YOLO versions [2], and contains documentation that is easy to understand.

2. Related Work

2.1. Call of Duty Zombies Aimbot using YOLOv5

Satori Digital implemented YOLOv5 to train weights on a dataset of different classifications of parts of a Call of Duty Zombies game, such as zombies, barriers, and weapons [4]. These weights were used to find the location on the screen of the zombie or other classes, and change where the player aimed through a distance calculation between the current mouse location and the center of the target. To make this function simpler, after each shot the mouse would revert back to where it started before the function ran. This showed a way to use object detection in the gaming industry.

2.2. Aim Labs Gridshot Ultimate Aimbot using Haar Cascades

Emeep developed an aimbot for Aim Labs Gridshot Ultimate using Haar Cascade, an object detection algorithm that only identifies one type of object at a time. Emeep achieved a score of 181,006 in Gridshot Ultimate, which is one of the highest scores of all time[1]. To achieve this score however, he changed the settings to remove distractions for the algorithm, such as the gun being in frame, or the background not being a solid color.

3. Methodology

In this section, I first introduce the training data collected from roboflow. Then, I will explain the different weights chosen from the YOLOv8 models.

Lastly, I present the different

3.1. Training Data

I used a public dataset from roboflow titled aimlabs which consisted of 187 images of Aim Labs Gridshot Ultimate with the targets labeled and split into 135 training images, 33 validation images, and 19 test images [7].

3.2. YOLOv8 models

YOLOv8 is a new model that builds upon the success of previous YOLO versions and introduces improvements to further boost performance and flexibility [2]. It has 5 different sized models which can be seen in figure 1, ranging from n to x, where n is the smallest and fastest object detector and x is the largest but slowest object detector. Since my goal is to create the highest scoring aimbot, my main goal is detecting objects as quickly as possible. Because of this I decided to to train my custom weights with the three smallest YOLOv8 models, yolov8n, yolov8s, and yolov8m. I trained yolov8n and yolov8s weights with 25 and 50 epochs, while just training a yolov8m weight with 25 epochs[3].

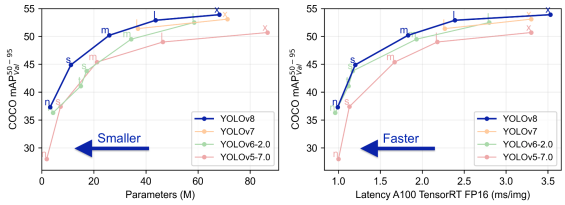

Figure 1: YOLOv8 models size and speed compared to other YOLO versions

4. Experiment

4.1. Singular Image Test

To decide which weights to use, I compared the inference time and confidence percentage of the test image shown in figure 2. This test image was chosen because it used a different background and color gun than the normal set up, which hopefully means that the classifications are not overfitting to just my own game and settings.

Figure 2: Image used for testing detection confidence and inference speed [6]

From this test I found that each of the different weight model types performed similarly to other weights of the same model type, which can be seen in figure 3. I also found that the base pretrained models are able to classify the targets as sports balls, which is class 32 for all of the pretrained weights. With this information, you can just search for sports balls with the pretrained weights and it will identify the targets similar to the custom weights. From this, we see that yolov8s appears to detect objects in a single image at the highest confidence in the most efficient amount of time, compared to yolov8n which does it slightly faster but with less confidence, and yolov8m which does it much slower and with similar confidence.

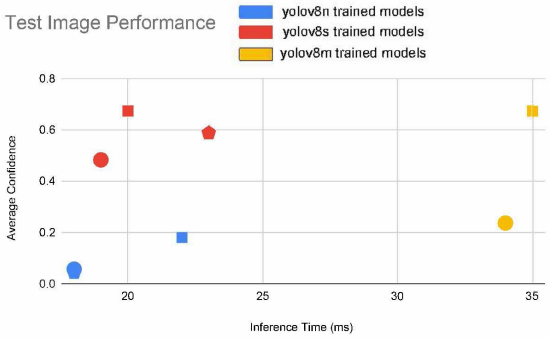

Figure 3: Scatterplot of average classification confidence and inference time of the weights, differentiated by color

4.2. Implementing the Aimbot

Using these object detections, I calculate the distance from the current point the mouse is at to the center of each of the classified targets. To get these center values, I take the average of the minimum and maximum x and y coordinates of the detected objects. I then multiply this value by a constant value (x = 1.75, y = 1.6) to factor the sensitivity in game. Then we choose the closest target, and move our mouse in the x and y direction by implementing the python library win32api to hopefully aim at the center of the target. If there are no targets detected, the mouse is

moved down and to the right 30 pixels to give the algorithm a different angle to view the targets. I found moving the aim towards where the gun is being held allows for the best chance to classify targets that may have been missed due to being behind the gun.

4.3. Results

After running each weight through the Gridshot Ultimate gamemode, the scores and accuracies were found for each of the different weights, which can be seen in figure 4. These were run minutes after each other in similar conditions to make sure the computer wasn't slowing down the later runs.

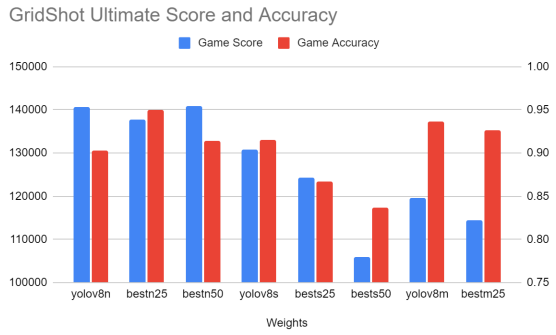

Figure 4: Gridshot Ultimate Score and Accuracy for each weight

During the testing of each weight in the game, I found that the scores were generally close, with the highest score during this single test being 140,569 points using the bestn50 weight and the highest accuracy being 95% using the bestn25 weight. Since all of them were above 100,000 points, which is better than most humans playing the game, I believe that most of the difference in scores was due to the computer being used to run the game, model, and google chrome tabs all at the same time. This can be seen in figure 5, which shows the score over time during the game. The score consistently increases until around 35 seconds in, where the inference time starts to become much larger than it normally was, and the score over time slope decreases a bit, before returning to a similar slope as the beginning around the 40 second mark. Another reason for differences in scores is that the targets show up randomly, so there may be a situation where each target ends up being next to every other target and there is minimal movement of the mouse needed to keep hitting the targets. Lastly, game accuracy may not be a very good indicator of high performance, since the way I wrote the shooting portion of the code may lead to unnecessary shots if the object detector cannot find any objects.

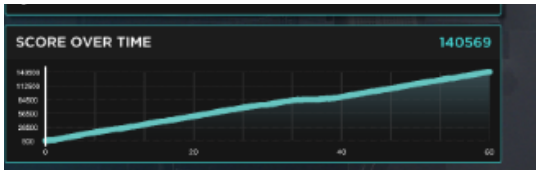

Figure 5: Score over time from running with yolov8n.pt

5. Conclusion

In conclusion, aimbots that use visual outputs to aim at enemies can be created without accessing the game client data with object detection algorithms. While there is no way to stop someone from training an object detection algorithm for a game, there are ways to check for non-human inputs, such as the use of pyautogui, to detect these types of aimbots. In the future, I would like to see how well classifications would work in games with multiple objects that would need classifications, to see if having many classifications slows down the inference process enough to make this not as viable. I would also like to figure out how to make the code work in a more user-friendly way, since running the program and quickly switching over to the game before it started moving the mouse was sometimes complicated. Lastly, I would like to run this program in an environment that can handle the detection and game at the same time, to see the full extent of this aimbot.

References

[1] Emeep, "Dominating Aimlab with an AI Aimbot," www.youtube.com. [Online]. Available: https://www.youtube.com/watch?v=g66uu8eGkAE&t=1s (accessed Dec. 14, 2023).

[2] G. Jocher, A. Chaurasia, and J. Qiu, "YOLO by Ultralytics," version 8.0.0, Jan. 2023. [Online]. Available: \url{https://github.com/ultralytics/ultralytics}

[3] P. Skalski, M. Read, "Train YOLOv8 on a Custom Dataset," Roboflow Blog, Jan. 10, 2023. [Online]. Available: https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/

[4] Satori Digital, "AI Aimbot | YOLOv5 Tutorial | Tech Breakdown # 2," www.youtube.com. [Online]. Available: https://www.youtube.com/watch?v=ilsn-TvryyA

[5] V. is T. N. Old, "How does aimbot work in COD?," Nov. 09, 2023. [Online]. Available: https://www.vintageisthenewold.com/game-pedia/how-does-aimbot-work-in-cod#:~:text=How%20do%20aimbots%20detect%20players (accessed Dec. 14, 2023).

[6] Image, "Aimlabs on Steam," store.steampowered.com. [Online]. Available: https://store.steampowered.com/app/714010/Aimlabs/

[7] Project, "aimlabs Dataset," Roboflow Universe, Roboflow, Open Source Dataset, June 2022. Available: \url{https://universe.roboflow.com/project-kvfn6/aimlabs}. Accessed: December 14, 2023