Aimbot Detection in CS:GO

Vidit Kumar

June 2025

1 Introduction

In the modern gaming landscape, cheating tools such as aimbots are increasingly undermining competitive integrity. This report presents an approach for detecting aimbot behavior in CS:GO gameplay using a Long Short-Term Memory (LSTM) based neural network model. We extract gameplay features from user-submitted video clips, perform sequential learning, and train a binary classifier to predict whether the gameplay is AI-assisted or not.

2 Dataset Generation

1 minute of 720p @60 encoded by H.264 encoding is used for feature extraction. With using aim-bot and without using aim-bot.

3 Feature Extraction

Features were extracted from raw CS:GO gameplay videos. The extraction process involves:

- Capturing optical flow between consecutive frames to estimate mouse movement.
- Detecting firing events via frame brightness spikes.
- Used YOLO-based person detection to estimate enemy positions relative to the crosshair.

The feature vector for each frame is:

$$[x_{pos}, y_{pos}, dx, dy, \text{is\_firing}]$$

where (x_{pos}, y_{pos}) is the vector to the nearest enemy, (dx, dy) is the approximate mouse motion from optical flow, and *is_firing* is a binary indicator for firing of weapon.

4 Model Architecture

The neural network used is a sequential LSTM model followed by fully connected layers:

- LSTM1: input size = 5, hidden size = 256, 2 layers

- LSTM2: input size = 256, hidden size = 128, 2 layers
- FC1: $128 \rightarrow 32$
- FC2: $32 \rightarrow 2$ (binary classification)

All intermediate layers use ReLU activations and dropout with probability 0.2.

5 Training Setup

- Loss Function: CrossEntropyLoss
- Optimizer: Adam with learning rate 0.001
- Sequence length: 600 frames
- Batch size: 1 (due to GPU memory constraints)
- Training duration: 20 epochs

6 Evaluation

The model was evaluated on a held-out test set using classification accuracy. The final model achieved near-perfect training accuracy, although care was taken to avoid overfitting by using dropout and shuffling batches.

7 Results and Discussion

Despite simple features, the LSTM model was able to learn temporal dependencies that distinguish human vs AI input behavior. Test set accuracy is 80 % (although test set was really small) The major bottlenecks were:

- Speed of feature extraction
- Lack of Data

So to overcome training bottleneck this type of model structure is used to train the model and it is also requires less compute.

8 Conclusion

This work demonstrates the viability of using sequential models for aimbot detection using gameplay video. While improvements are possible by enhancing feature representation and adding enemy detection accuracy, the current model shows promise for real-time applications.

9 Sources

- This whole report is AI generated and little bit code is also written by AI.
- YOLOv5: https://github.com/ultralytics/yolov5
- Aimbot used for training purpose: https://github.com/SunOner/sunone_aimbot