CSGO Aimbot Detector

Sudarshan Kulkarni

1 Dataset

I went through 4 videos on youtube of (legit) CS2 gameplay, manually marked the timestamps when the kills took place, and then through libraries like ‘pytubefix‘ and ‘moviepy‘ I downloaded and split the videos into 3s long clips @ 30fps and 360p resolution.

For gameplay of aimbot, as unfortunately I couldn’t download the game on my laptop (due to wifi constraints), I got clips for about 30 kills from my friend Vidit, who is also participating in this track of IG.

2 What I tried

My general outline was, using a pretrained CNN to get a feature representation for each frame of a clip. Then pass the representations into a sequence model (GRU/LSTM) and then take their final output and pass it through a (Variational) Autoencoder, and training it with reconstruction (+ KLD loss if VAE)

During test time, I’d pass the clip through the model, and if the loss was above a certain threshold, it was classified as aimbot, otherwise as real gameplay.

2.1 LSTM + VAE

Using just first 3 gameplay videos for clips (86 clips for training), it correctly classified 26/30 aimbot clips as aimbot. In this case, my test set for actual games was having only 7 clips, and it correctly classified 6/7 of them as no-aimbot.

But when I included the gameplay from the 4th video, which contained about 55 kills, it firstly performed poorly on original model, and moreover decreased overall accuracy to 68% on a model trained with data of all 4 videos. The former can be partially explained because the map was different in new clips, but latter is more difficult to explain.

Nevertheless, I had to fix this. So I decided to try some regularization. Weight decay of 10^{-4} didn’t help much. It just increased the training and test loss overall, but didn’t help with accuracy.

2.2 2.1 + Denoising VAE

Unfortunately, didn't work either, this gave a lower accuracy of 68%. I couldn't get time to go through entire clips of 4th video, but I decided to drop it and go with original 3 only, which were from 2 (similar) maps. I believe there is a bigger problem with the data apart from just the map.

2.3 2.2 without video4

This also failed, giving only 61% accuracy , which made me wonder why it had originally worked in section 2.1. So I decided to try to go back and retry that.

2.4 2.1 with AE and VAE

I tried the part 1 of 2.1 again, with AE, and by reducing the network size to a single hidden layer of 256 dim and latent space 64. This brought accuracy upto 70% but not to original level. This now mostly leads to me to believe that the initial success was a fluke due to a small test set of only 7 clips

2.5 Final Thoughts

I believe that adding the extra data from new map wasn't the actual issue, it was that my first testing consisted of too less examples and was a fluke. If had time, I would use GRUs with denoised sprase encoders (I haven't implemented sparse AE in code, so no time for it now), along with weight decay.

One more thing which I SHOULD have done, was pre-computing the feature vectors for all the clips through resnet. This would've massively boosted training speed and helped with trying out more models quickly. I will try these stuff again as I find time after CSOC

I started a bit later as I was busy with some personal stuff. This was a very nice experience as I learnt about AEs and also implemented a AEs and LSTMs for the first time. Also very thankful to seniors for this as I otherwise wouldn't have learnt about AE so easily 🙃