Mine-RCNN

Loi Dario (1940849), Marincione Davide (1927757), Barda Benjamin (1805213)

Abstract—Real time object detection has recently been made possible due to steady state-of-the-art advancements in the field [1], [2], these methods propose the use of a Region Proposal Network to identify Regions of Interest (RoIs) in the image and correctly classify them, we aim to reproduce the architecture proposed by [2] applied to a novel environment, that of the popular sandbox Minecraft, both for the ease-of-collection of the required data and for a number of graphical properties possesed by the game that make such a complex problem more approachable in terms of computational resources, moreover, due to the novelty of the environment, we also train the entirety of the network from the ground up, having no pre-trained backbone at our disposal.

Index Terms—Object Detection, Convolutional Neural Network, Sandbox, Region Proposal, Real Time Detection

1 Method

1.1 The Dataset

We started first by building the dataset. We recorded one-minutes long videos of minecraft using commercial screen captures softwares. We then loaded those shorts into python, using the OpenCV library, in order to sample one frame per second as we beleived would have given enough time for the next sampled frame to have significative diffrences with respect to the previous ones. We then downsampled the images in order to compress the size of the dataset in order to be able to share it with ease. To further reduce the problem we adopted for the final image a one-to-scale ratio, thus making the image squared. At this time we opted to limit ourselves to only five classes we were aiming to classify: Zombies, Creepers, Pigs, Sheeps, Nothing. The next step was labeling each sampled frames. We developed a simple but effective tool that allowed us to draw boundng boxes(BBox), and assign to each one of them a label corresponding to a class mentioned above. During this process we pruned images that we considered unfit to be part of the dataset (e.g. frames inside the game menu or outside of the game). After standardizing the coordinates of BBoxes we saved them into JSONs files. Having our JSONS files ready we group them into a single .dtst file for better integration with the PyTorch library, which is the one we decided to use for this project. From the sampling of the images we collected 3920 valid frames.

2 Results

Given the inexperience, the difficulty of the task and the (inadequate) hardware at hand, we think to have reached positive results, the system shows signs of being able to recognize traits of the mobs we've trained it on, even though sometimes they are just cases of pareidolia: to give some numbers, the final model achieves a staggering 94% accuracy on the validation set when recognizing whether an anchor contains an object or not. Is this a nice result? It depends, the real problem with this value is that, essentially, non-object anchors dominate the dataset (that's normal, unless you happen to be trying to classify every possible thing in an image), this

References

[1] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[2] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497