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 Introduction

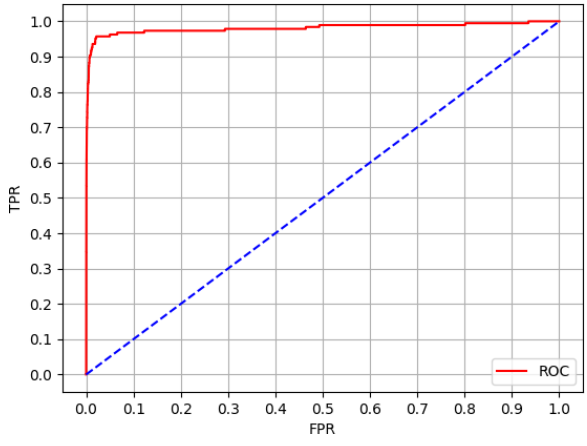
2 Method

2.1 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.

3 Results

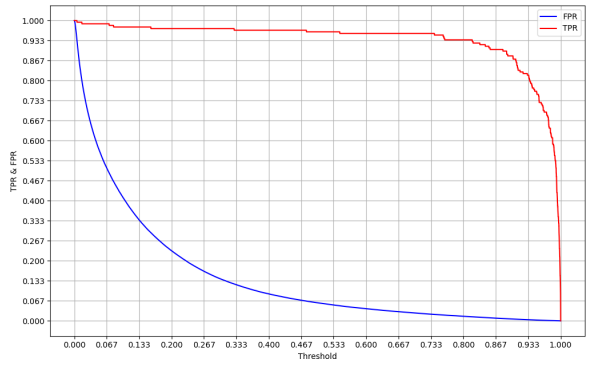
Given the inexperience, the difficulty of the task and the (inadequate) hardware at hand, we think to have reached positive results, the system shows signs of being able to recognize traits of the mobs we've trained it on, even though sometimes they are just cases of pareidolia. Before giving some manner of statistics over its capabilities we would like to point out an important decision: that of the threshold for deciding whether, given a score, the anchor for which it is related to is actually a positive one or not. To decide this fundamental hyperparameter we resolved in sampling five hundred (500) images from our training dataset and, after letting the system apply non-maximum suppression, recovering the scores for all the remaining anchors and pairing them to their true labels. Once this preprocessing was done we plotted the ROC:



And a graph, showing the decrease of the *true positive ratio* (TPR) and of the *false positive ratio* (FPR) as the threshold increased, as following:

From both these graphs we notice our network is indeed able to separate positive and negative anchors with very clear-cut definition. Furthermore, we decided that a good compromise between (TPR) and (FPR) could be achieved by choosing a threshold between 0.8 and 0.9: since we didn't want to remove positive anchors too much we put ours at 0.81. We posit that, to avoid false positives as much as possible, 0.9 should work fine too.

After this, in our opinion, fundamental decision was made: we resolved to open Pandora's box and



create a test set of around 250 images with multiple mobs in the same shot. Unfortunately, as said previously, we didn't develop our network up to classification of the boxes, and thus a full confusion matrix of those is out of the question. At any rate we show the misclassification table for the anchors (after non-maximum-suppression) scored by our network over the whole test set:

	Pos. label	Neg. label
Pos. score	62	3545
Neg. score	17	127789

Telling us that, indeed, our threshold works, since the TPR is 0.78 and the FPR is 0.03. We would like to point out that, since the labelling of the anchors is itself hyperparameter driven (given the choice of anchors and that of the IoU thresholds used to label them) and that we think to have chosen a combination of these parameters that biases the labels towards the non-object side and, furthermore, given this label by itself already dominates the distribution: it is only normal for the number of positive labels to be so low in the set. If we look at the actual images with the positive region proposals added, we'll appreciate much more positive-looking (that may not actually be labelled as positive) proposals than the anchor labelling would actually suggest.

Let's now give a look at some of the proposals created by our network:

CUE A SLEW OF IMAGES WITH COMMENTS

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