Mine-RCNN

Or how we were able to recognize pigs et. familia in Minecraft

SAPIENZA
UNIVERSITÀ DI ROMA

Dario Loi, Davide Marincione,
Benjamin Barda

Bachelor's degree in
Applied Computer Science and Artificial Intelligence
Sapienza, University of Rome

A. Y. 2021 - 2022

Breaking the ice

Figure: Us.

Faster RCNN

Developed in 2015 by Facebook's researches, Faster-RCNN is still today an industry standard thanks to it's accuracy and performance, getting a step closer to real time object detection

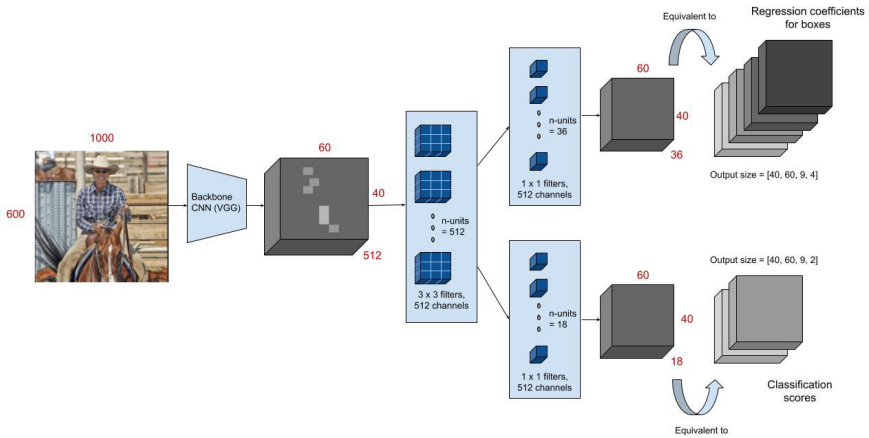

Figure: Faster-RCNN architecture.

Why minecraft?

Minecraft has several desirable qualities:

- Simple graphics.
- Sandbox.
- Available to every team member.
- Distinguishable entity silhouettes.

Figure: A Minecraft promotional image.

Behold, data!

4000 images spread across 40 videos!
How did we collect these videos?

- 1 minute long (circa).
- As many biomes as possible.
- One mob per video (except test).

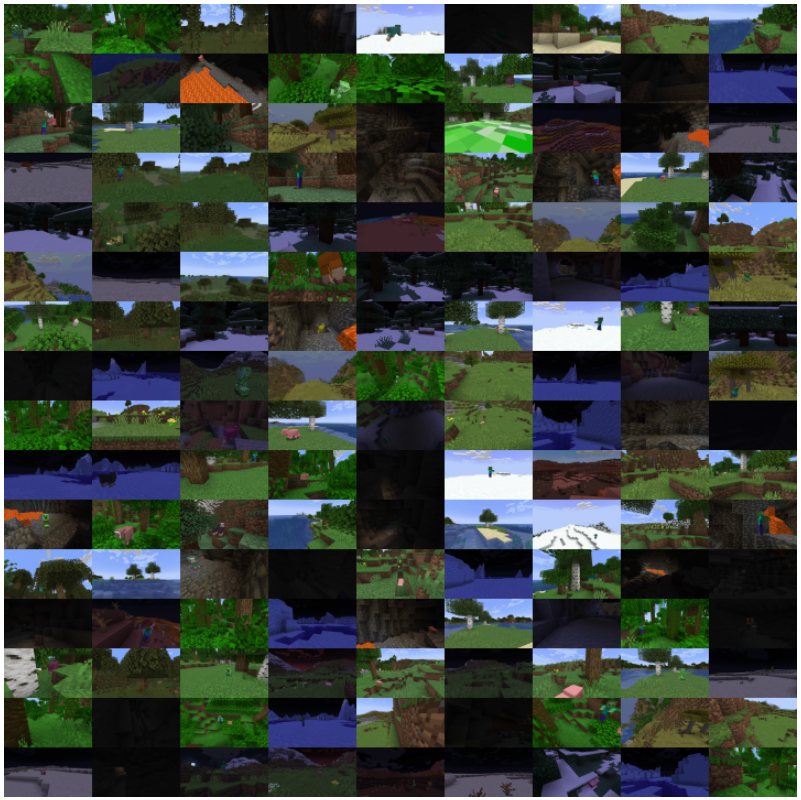

Figure: A representative chunk of our dataset

Augmentation Techniques

In order to prevent overfitting and increase the amount of information available, we employed various data augmentation techniques, such as:

- Rotation and Reflections.
- Adjustments to Contrast, Brightness and Saturation.
- Sharpening and Blurring the image.

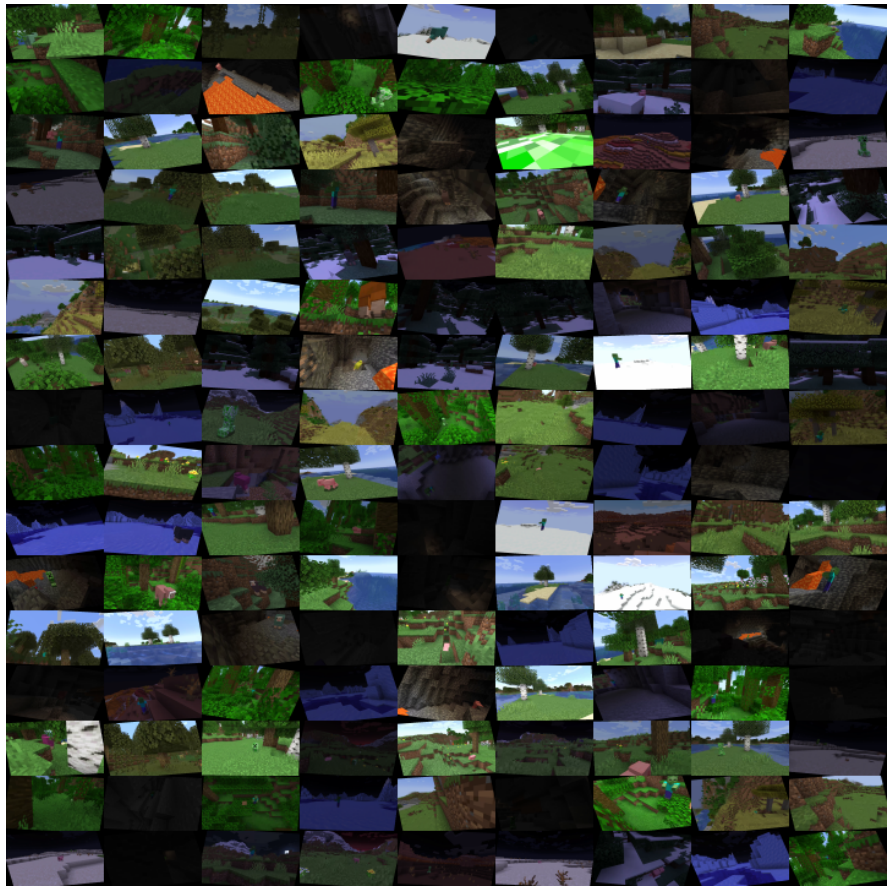

Figure: Our Dataset, Augmented.

Tool

How to label 4000 images?

1. Load image
2. Create box / purge
3. Next

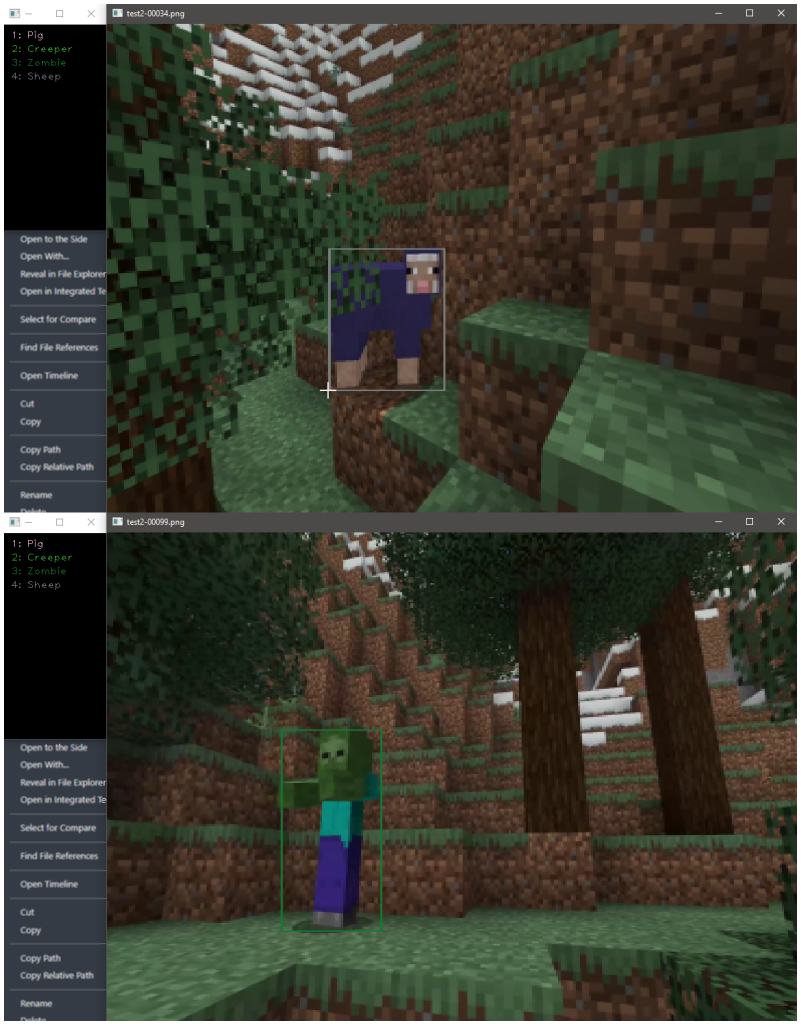

Figure: BBoxing in our tool

Our Backbone

The backbone is the convolutional *heart* of our model, it is:

- Blazingly fast.
- Adaptable to any resolution.

While also offering:

- A 92% accuracy when used as a Classifier.
- A mean training time of $\approx 2h$.

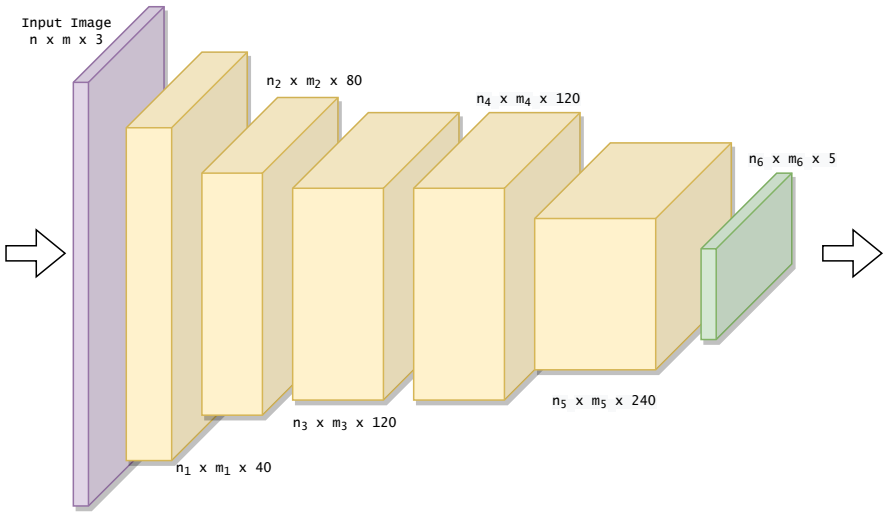

Figure: Our backbone.

Our RPN

Our RPN network extends our Backbone and is composed mainly of two twin layers:

1. A Classification layer.
2. A Regression layer.

Before feeding data into those, it also performs some pre-processing:

- Anchor Splashing.
- Base convolution.
- Flattening (how do we get to fully connected otherwise?)

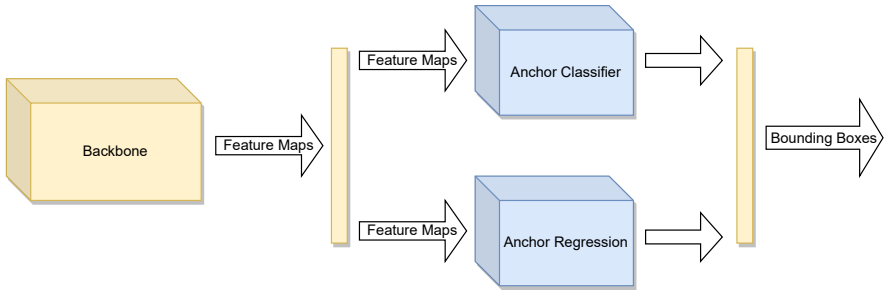

Figure: Our network's proposal layer.

Our Tricks

In order to allow our Backbone and RPN to *punch above their weight*, that is, learn a complex task with fewer parameters or with a smaller dataset, we employed two popular regularization techniques:

- Batchnorm. A Layer that computes a running μ and σ^2, standardizing the input at inference time, this increases capacity.
- Dropout. A Layer that randomly zeroes out outputs of the preceding layer, adding bagging[1] to our network and making it more robust.

[1] Essentially, simulating an ensemble without performance costs.

Examples (1/2)

Even if many proposal are presented, the network realizes which is the objects to focus on, and which to discard
It is not always that easy. . .

Figure: A creeper in it's natural environment

Examples (2/2)

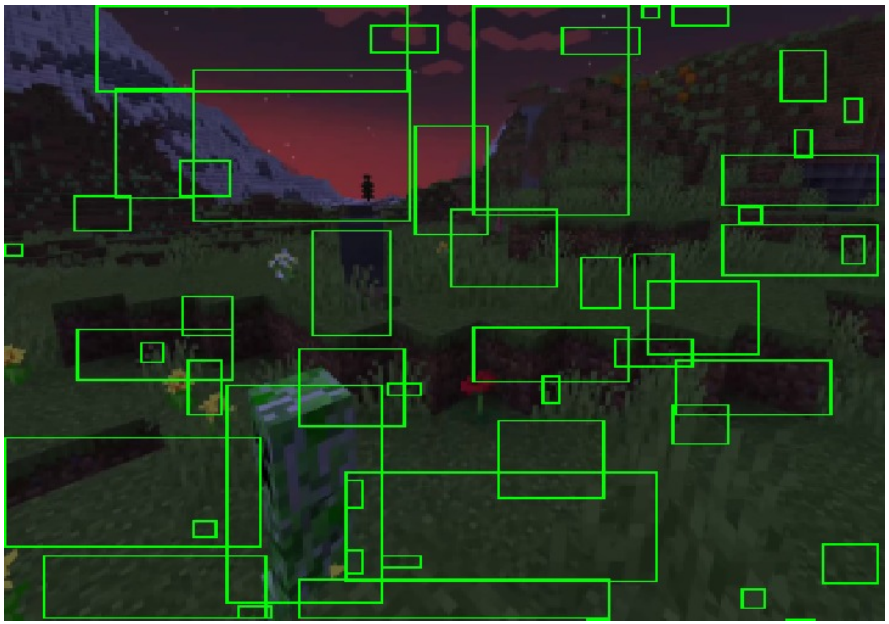

Figure: A very confusing sunset[a]

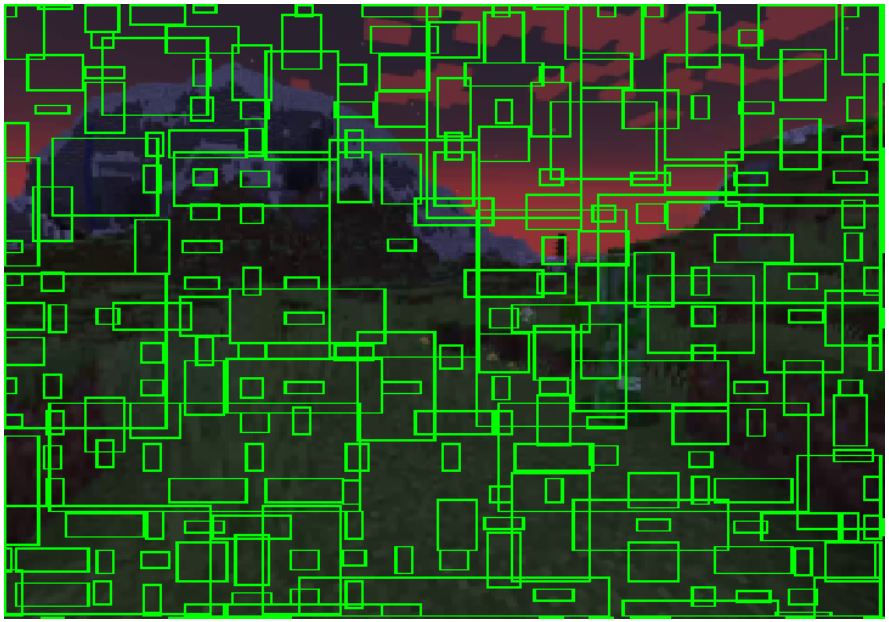

Figure: The network struggling

[a]The issue is talked about in-depth in our paper.

In Conclusion...

We believe that, even if the model has its hiccups when presented with weird scenarios, the mere fact of being able to deploy such an architecture on a novel dataset, having trained it completely from scratch, is quite a good result.

Most importantly, the model is able to identify objects at inference time at an impressive speed, and, thanks to optimization techniques such as *model freezing* and *quantization*, it could be used as a tool for real-time object detection.

The End.

Question Time!

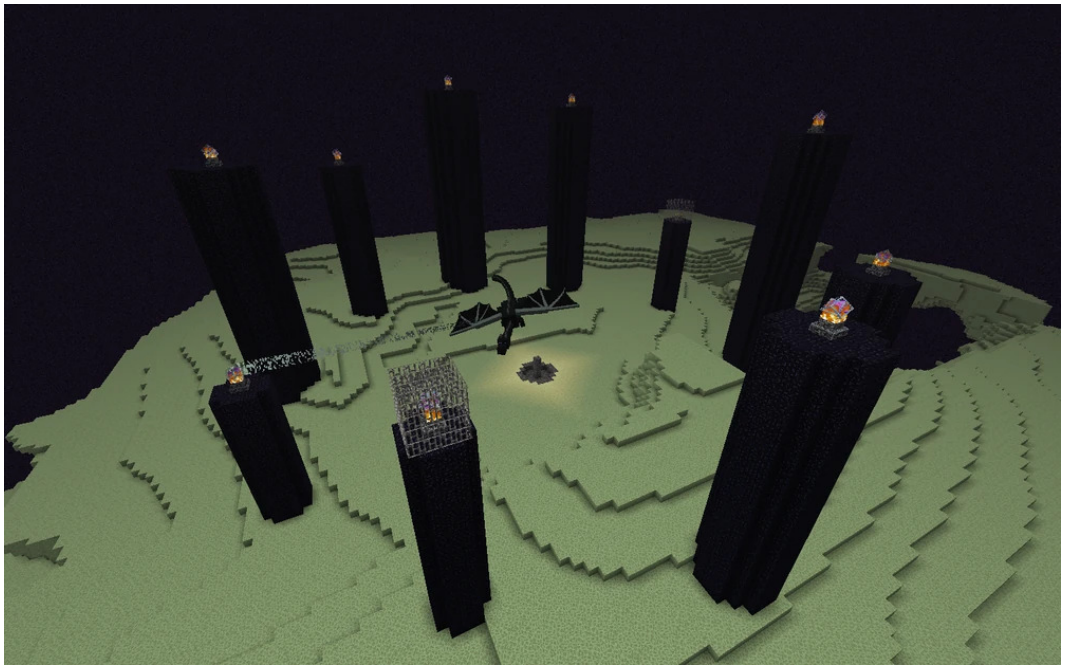

Figure: The End.