NOVA course on deep learning in remote sensing

Home exercise

Jaime Candelas Bielza

1. OBJECTIVE

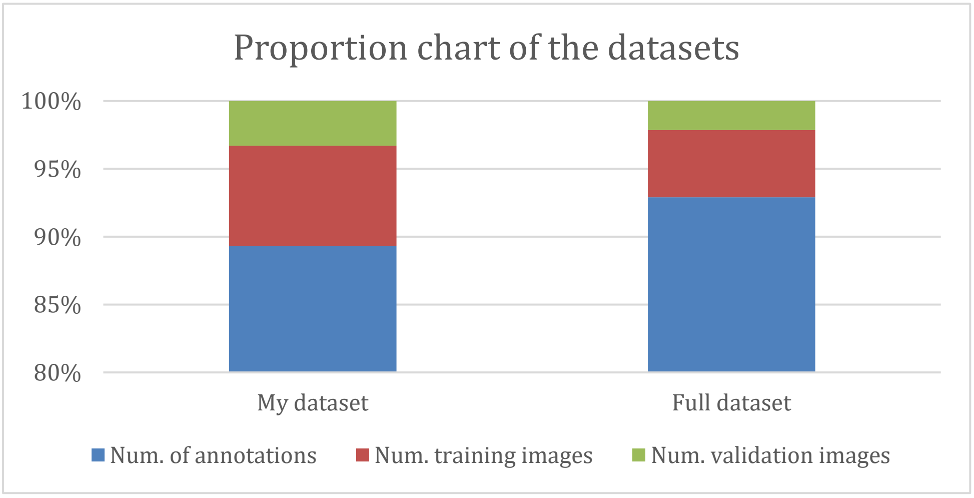
Compare the effect of training a seedling detector on your own annotated dataset Vs full dataset (all annotations merged) on the detector's performance.

2. DATA

Each classmate got an individual set of images for the training of their models and at the end of the course, all annotations and images were shared to compare performances.

Dataset	Num. of annotations	Num. training images	Num. validation images
My dataset	569	47	21
Full dataset	5074	270	117

In deep learning models such as YOLO, the number of annotations is crucial and it is important to have as many as possible. However, the quality of the annotations also affects the performance of the models.



In this case, the full dataset has more annotations relative to the number of images than my dataset. It will be interesting to see how much they differ and why.

3. METHODS

With these datasets, two set of models were made and trained, one per dataset.

3.1. Model Training

For each dataset, nine YOLO v8 were made and trained based on the possible combinations of 3 YOLO sizes (nano, medium and large) and 3 image sizes (416, 608 and 800). The rest of hyperparameters were set as default (epochs = 300 among them).

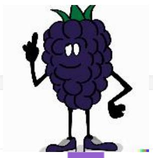
YOLO_size	**Image_size**
nano	416
nano	608
nano	800
medium	416
medium	608
medium	800
large	416
large	608
large	800

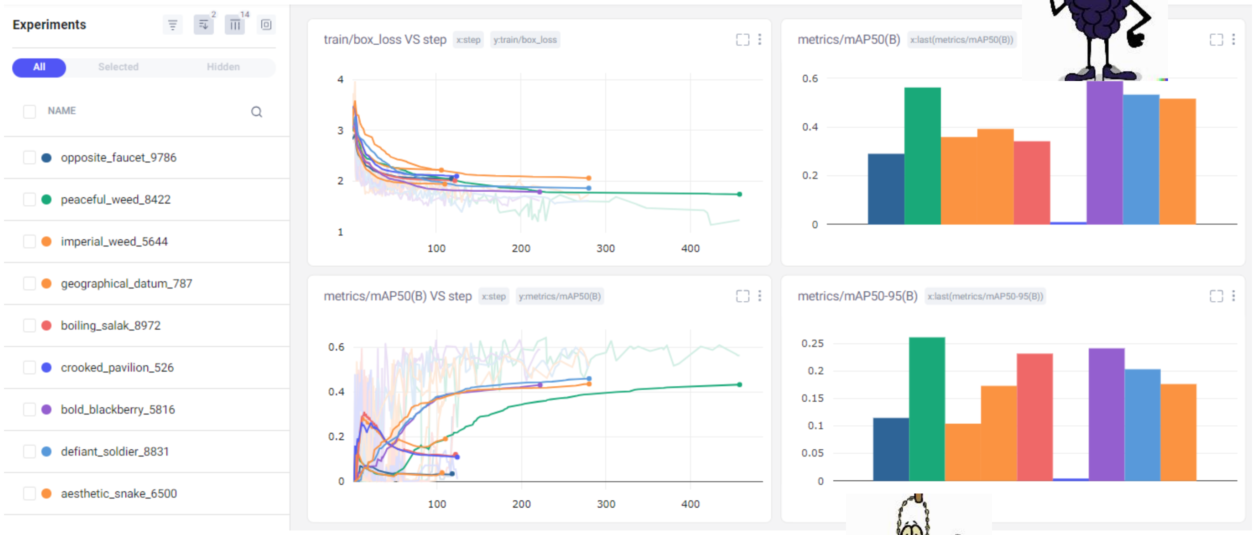
3.2. Model Evaluation

After training the models, their performances were assessed in COMET with the help of the graphs.

3.2.1. Models from my dataset

From the models trained with my dataset, the best ones according to mAP50(B) were peaceful_weed_8422 and bald_balackberry_5816, which corresponds to a YOLO Large and image size of 608 and a YOLO Nano with image size of 800, respectively.





● opposite_faucet_9786
● peaceful_weed_8422
● imperial_weed_5644
● geographical_datum_787
● boiling_salak_8972
● crooked_pavilion_526
● bold_blackberry_5816
● defiant_soldier_8831
● aesthetic_snake_6500

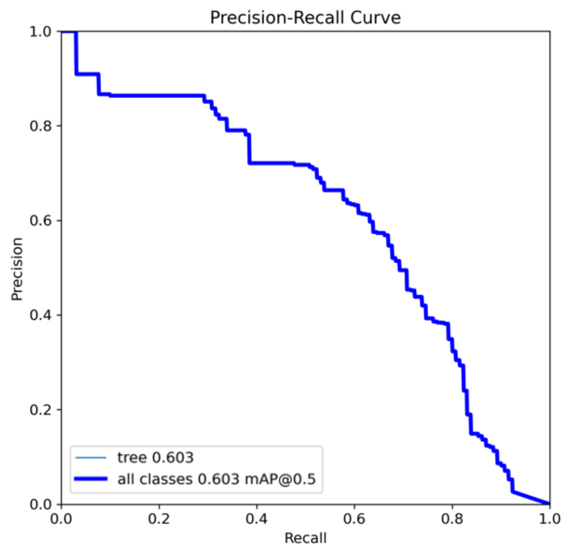
Is also worth noting that the YOLO Large went further along the Steps axis, which means that the model has been trained for a longer duration or has undergone more training iterations. More complex models require more iterations to achieve optimal performance, thus, more usage of resources.

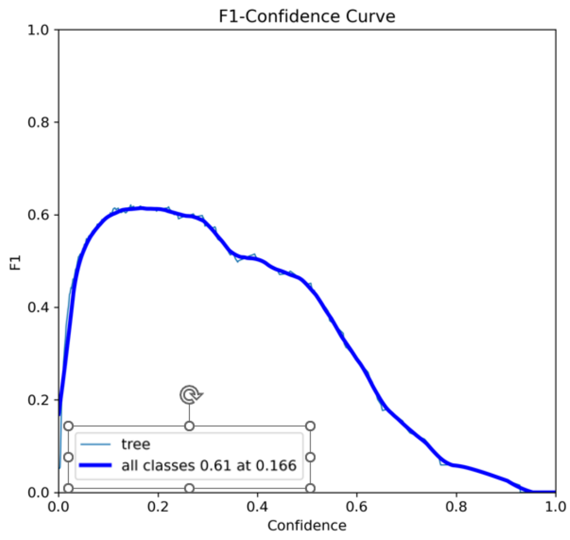
This outcome is interesting because it indicates that the smaller and faster YOLO Nano model with larger images (800) is able to achieve performance similar to the larger YOLO Large model with smaller images (608), which suggests that the larger input size compensates for the smaller model architecture.

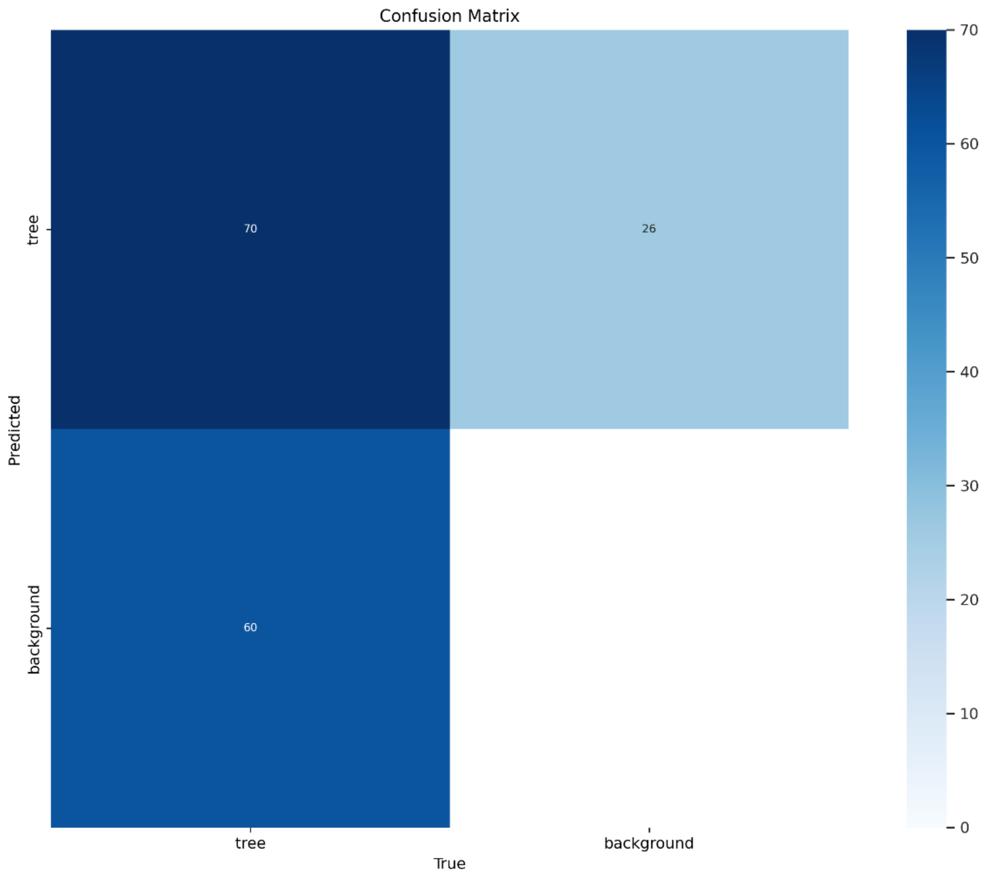
This finding highlights the potential trade-off between model size/complexity and image size in object detection tasks. It suggests that using a smaller model with larger images can be an efficient approach to achieve similar performance while reducing computational requirements.

Since both models achieved similar mAP50(B) values, YOLO Nano less resource constraint, which allows to mount it in smaller devices and could be used in real-time scenarios, I am going to select the **YOLO Nano with an image size of 800** for further analysis.

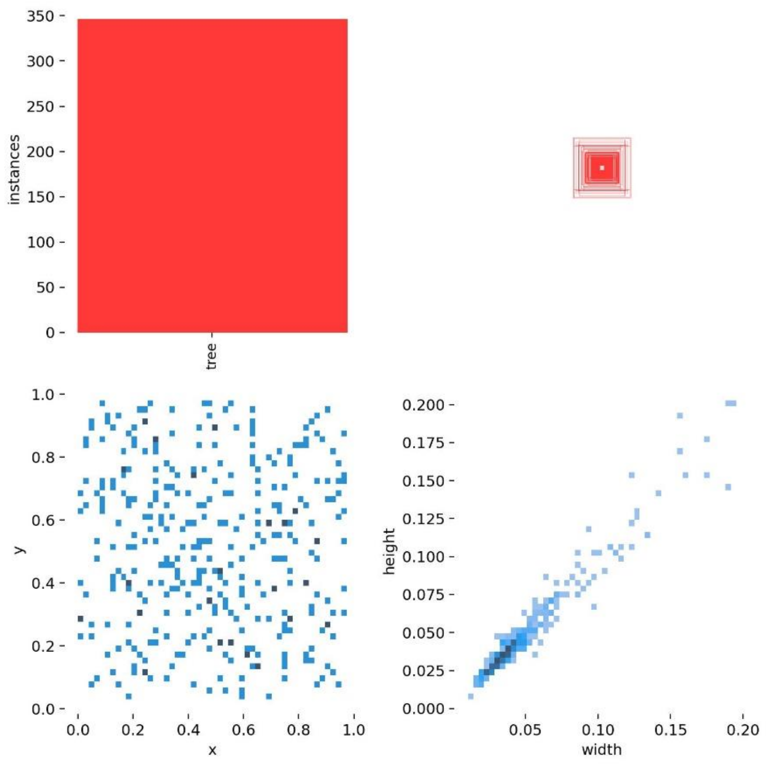
Precision-Recall and F1-Confidence curves don’t show irregularities and shows what could be a normal case.







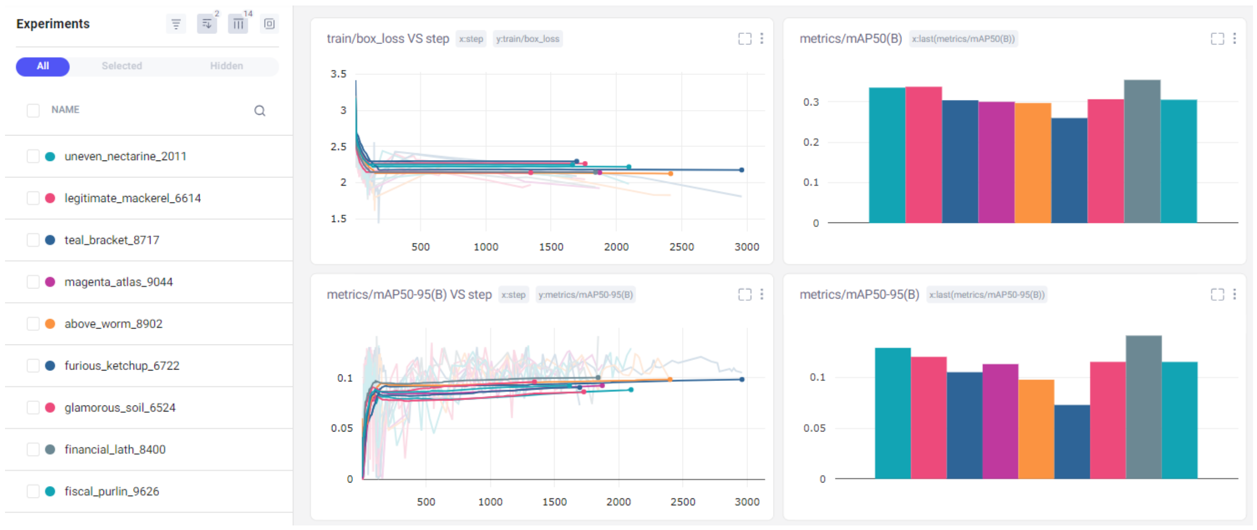
Confusion Matrix
70
26
60
tree
background
Predicted
tree
background
True
70
60
50
40
30
20
10
0



350
300
250
200
150
100
50
0
instances
tree
1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
height
0.05
0.10
0.15
0.20
width

3.2.2. Models from the full dataset

From the models trained with the full dataset, the best one according to mAP50(B) was financial_lath_8400, which corresponds to a **YOLO Nano and image size of 608**.



Experiments

All Selected Hidden

NAME

● uneven_nectarine_2011

● legitimate_mackerel_6614

● teal_bracket_8717

● magenta_atlas_9044

● above_worm_8902

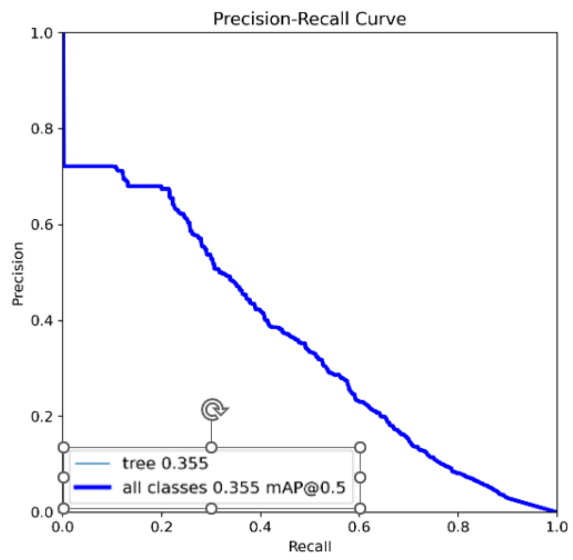
● furious_ketchup_6722

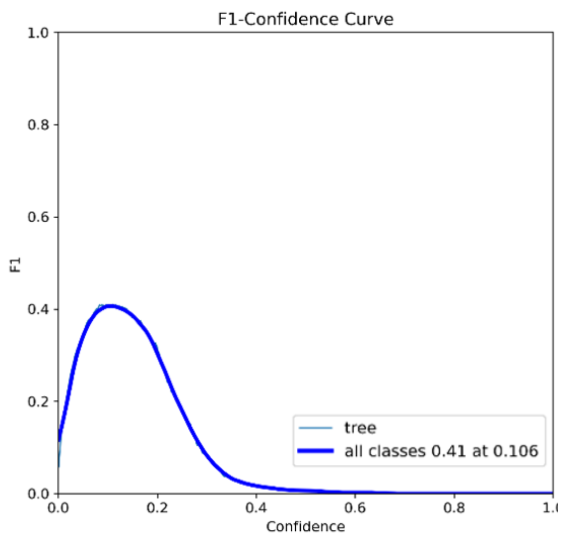
● glamorous_soil_6524

● financial_lath_8400

● fiscal_purlin_9626

While F1-Confidence curves looks good, the Precision-Recall curve does not.



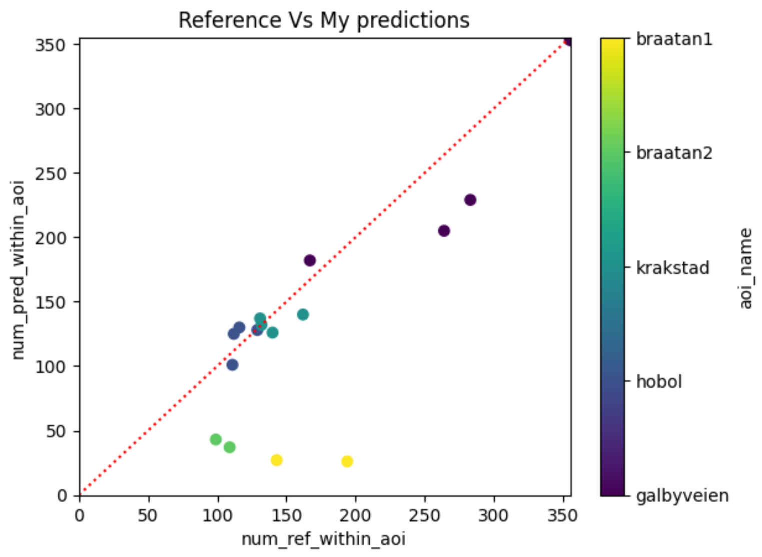


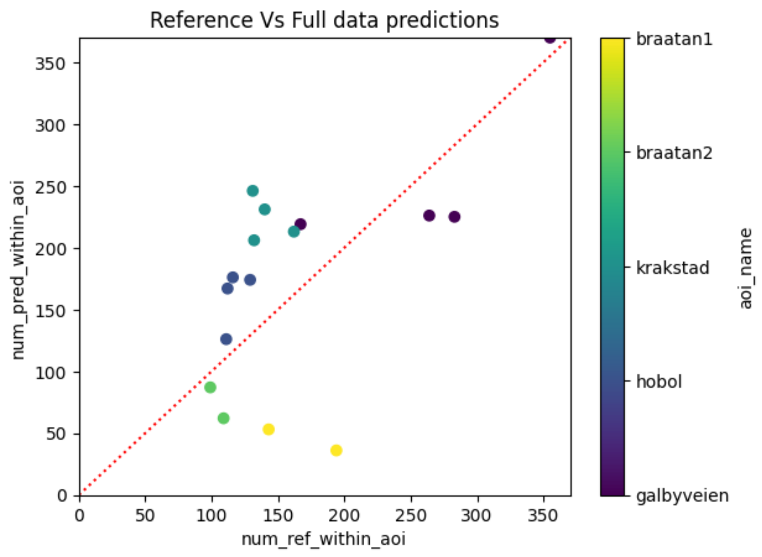
- Comparison of the two detectors trained:

Dataset	**Model**	**Max mAP@5**
My dataset	YOLO Nano and image size of 800	0.589
Full dataset	YOLO Nano and image size of 608	0.354

- Domain-specific (RMSE, bias, RMSE%, bias%, scatterplot of reference Vs predicted):

Dataset	RMSE	bias	RMSE/ha	RMSE %	Bias/ha	Bias
My dataset	60.1	-32.8	641.9	43.6	-338.7	-23.0
Full dataset	71.4	10.6	715.1	48.6	65.5	4.4





4. DISCUSSION

- Comment on the impact of different hyperparameters (model training) on the detectors' performance and inference speed (some COMET plots would be greatly helpful to explain the experiment 😀).

  As comented above, the more complex is the YOLO structure used, the more steps (itarations) it needs to become stable.

- Is the model trained on more data better than the one trained on your own dataset? Discuss what could be the underlying causes.

  The model trained with my annotations performed better than the one from the whole class (RMSE = 60.1 and RMSE = 71.4) although it underestimated the number of seedling while the whole class' model overestimated. This can be because I missed seedlings while annotating, while my classmates were making annotations to other objects than seedlings (other herbaceous plants for example).

- Provide at least three example screenshots of cases where the detector performed poorly and discuss what could be a good strategy to mitigate such issues.

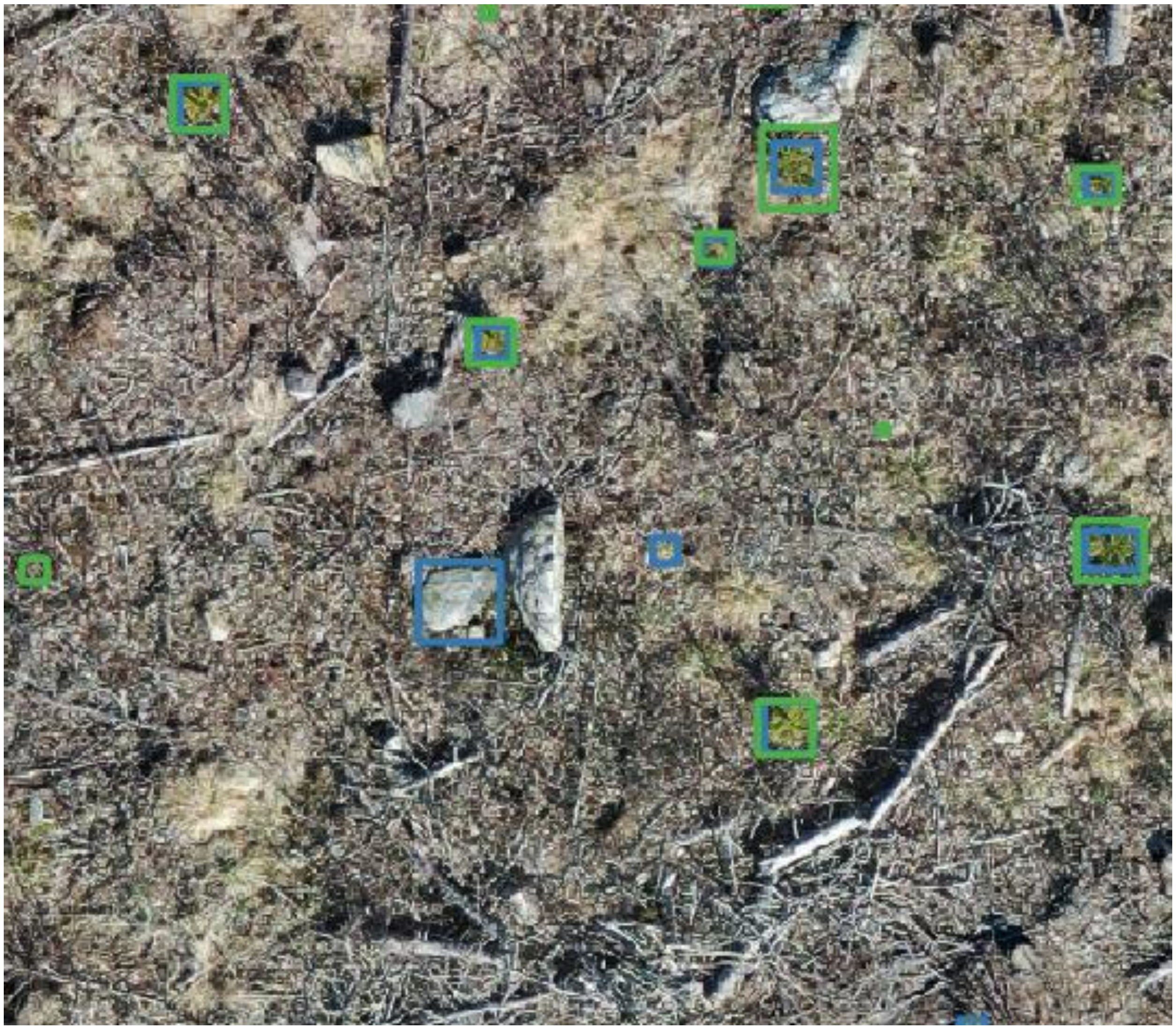


Figure 1. Screenshot where a rock was predicted as a tree (blue is predicted and green is reference data).

In this case a rock was predicted as a tree. This could be due to poor annotation.

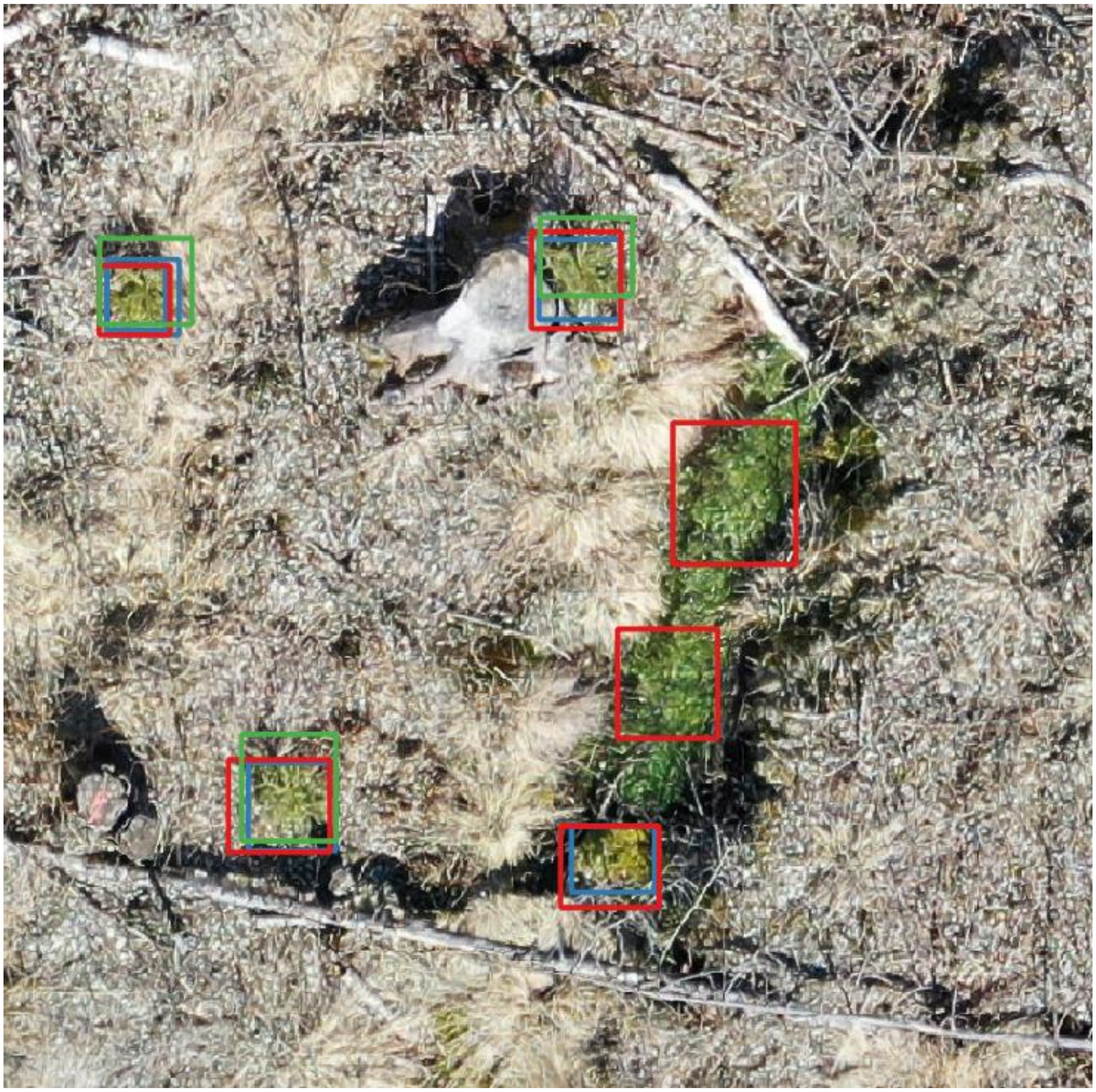


Figure 2. Screenshot where other vegetation was predicted as a seedling (blue is my prediction, red is the class' prediction and green is reference data).

In this case where other vegetation was predicted as a seedling the reason could be also because poor annotation.

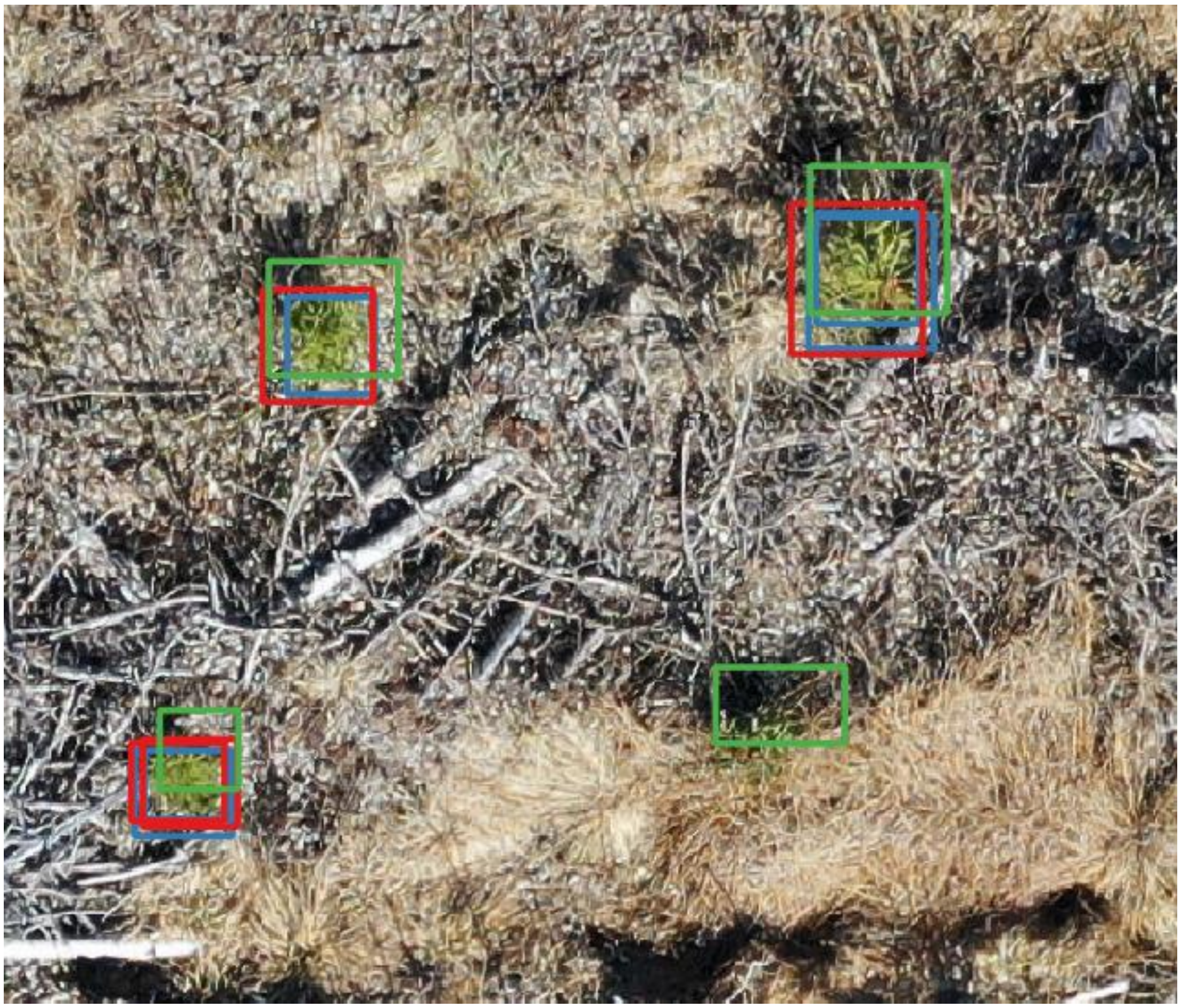


Figure 3. Screenshot where a seedling was not detected (blue is my prediction, red is the class' prediction and green is reference data).

In this case where a seedling was not detected by any model could be also by poor annotation.

Mainly, the annotation stage is the most important. What we see is what the models will see too so if we don't invest enough time at this stage, the rest of the modeling will be at a steak.

5. Github repository

Following Nico's tutorial, create a github repo and upload your code there and paste the link to your github repo here.