**NOVA course on deep learning in remote sensing**

**Home exercise**

# **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.

# **Methods**

## Statistics

|  |  |  |
| --- | --- | --- |
|  | **Own** | **Full** |
| **n images** | 22 | 136 |
| **n trees** | 166 | 1131 |

## **Model Training**

I have done the models training by setting different hyperparameters to find the best performing model. I was playing around with following parameters

* Image size: 640, 1028, 1280, 1824
* Epoch: 50, 100, 200, 300
* Batch size: 2, 4, 8, 16 (The CUDA memory capacity is the limiting factor here, so I must reduce the batch size to keep the model training).
* Model: I played with all YOLO models from nano to extra-large, all have different computational power and different pre-trained weights. Again, CUDA was limiting factor here, we must either reduce image size, batch size or model to accommodate available memory space.

## **Model Evaluation**

Thank you for introducing us to Comet.com as it provides the detailed graphical representation of model training in real time. The detailed information about train/loss, val/loss, Accuracy, Recall, mAP, metrics, box/loss against each step/epoch can be easily visualized on experiment panel. The comparative panel provides detailed and comparative information about all related experiments with different hyperparameters.

The following graphs explains the difference between models’ performance using own and all data set.

![A screenshot of a computer

Description automatically generated with low confidence]()

# **Discussion**

The model hyperparameters are the most variable and important factors for successfully training a model to detect the object of our interest. The combination of different parameters were applied to customize the training of our models and then analyzed through Comet and other graphical windows for different number of factors including training losses, validation losses, precision, accuracy, inference time and overall prediction capabilities of the trained model.

Here is the Comet output for model training on whole dataset with different hyperparameters:

![A screenshot of a computer

Description automatically generated with low confidence]()

Unexpectedly, the model performed better on my own dataset compared to the combined one. The best training results were obtained using the own dataset as shown below:

|  |  |
| --- | --- |
| A picture containing flower, plant, mosaic, screenshot  Description automatically generated  Labelled batch | A picture containing screenshot, map, text, mosaic  Description automatically generated  Predicted batch |
| A picture containing text, diagram, font, number  Description automatically generated | A picture containing text, diagram, line, plot  Description automatically generated |
| A picture containing text, diagram, line, screenshot  Description automatically generated | A picture containing text, diagram, screenshot, design  Description automatically generated |

I have analyzed the results for both datasets using different but comparable hyperparameters. The underlying cause for different training capabilities could be in different annotations style by different students attended the course. Even though the number of pictures and tree is far greater in the combined dataset compared to my own small dataset but different way of looking at pictures and unique annotation style is confusing the model training with whole dataset comprising small different datasets.

The relationship between different models training and hyperparameters can be summarized as:

**Training speed:** The small image size, small no. of epoch and smaller models like nano and small version of YOLOv8 results in efficient and faster model training compared to models with higher architecture and bigger image sizes.

**Model training:** The model training primarily depends on annotation quality, image size and choice of model. If the object is easy to detect then small and efficient models do their job very well. We have to try different parameters to optimize the model training with best possible results.

**Memory utilization:** As we are working on Google Colab with free but variable GPU access, so model training is highly influenced by availability of resources, If GPU is not available then it takes forever for CPU to train the model. Setting higher hyper parameters to improve the model performance also results in CUDA ran out of memory that stops the training process.

**Augmentation:** Data augmentation plays an important role in multiplying the number of training data without the need of annotating a greater number of pictures. Different augmentation including flip, rotate, hue and exposure results in multiple strides of the same picture that results in more efficient model training.

**GITHUB repository**

# The scripts to a github repo

[aqi147/ProjectNOVA (github.com)](https://github.com/aqi147/ProjectNOVA)