**YOLO-NAS OBJECT DETECTION**

La Sala Carlo - 1956560 Lubrano Valerio - 1935801 Scuderi Matteo - 1937090

# **0. Folder Structure**

### **Datasets** (Contains the three datasets we used for training)

* GreenDetection.zip
* Hituav-a-Highaltitude-Infrared-Thermal-Dataset.zip
* SkyFusion-YOLO.zip

### **Jupyter\_Notebooks** (Contains all the scripts we used for both training and testing)

* LabelsArrangement.ipynb
* Satellite object detection(Green).ipynb
* Satellite object detection(Sky).ipynb
* Satellite object detection(Thermal).ipynb

### **Trained\_Models** (Contains all the scripts we used for both training and testing)

* yolo\_NAS(M)\_Green(10EP).pth
* yolo\_NAS(M)\_Sky(20EP).pth
* yolo\_NAS(M)\_Thermal(20EP).pth

### **Videos** (Contains all videos used, before and after inference)

* Green\_Miami.mp4
* Green\_Miami\_predicted.mp4
* Green\_Ostia.mp4
* Green\_Ostia\_predicted.mp4
* Sky\_Hawaii.mp4
* Sky\_Hawaii\_predicted.mp4
* Sky\_Fiumicino.mp4
* Sky\_Fiumicino\_predicted.mp4
* Thermal\_Drone.mp4
* Thermal\_Drone\_predicted.mp4

# **1. Intro**

For this project, we first explored the world of object detection algorithms. This branch or computer vision is fastly advancing, thanks to new Deep Learning techniques and algorithms. We discovered that due to its fast inference time and high accuracy, the most used algorithm is exactly the YOLO one.

However, from 2015 more than 15 official versions of YOLO have been released, proving that the object detection field is widely researched and investigated all over the world.

## **1.1 Implementation Choice**

Even though we could opt for the state-of-the-art, brand new YOLO-v9 developed by Ultranalytics, or for some “simpler” older versions, we made our choice based on the purpose of this project.

Since we knew we had to deal with satellite object detection, we knew that our target classes would mostly be small objects. YOLO-NAS has been proved to be extremely precise in small object detection and the results we found just confirm such a claim.

Therefore, we picked the **YOLO-NAS implementation** offered by **Supergradients**, which allows to run inference on videos with no particular effort and is optimized for CUDA devices.

## **1.2 Datasets Choice**

We picked three very different training Datasets that we found interesting:

1. **Thermal Infrared UAV Images:**

Dataset of thermal images taken in a urban setting, created for detecting people, bicycles and vehicles

1. **Satellite Vehicles Images (Sky-Fusion):**

Dataset of Satellite and aerial images containing land vehicles, aircrafts and boats

1. **Tree Urban Detection (GreenDetection):**

Dataset of Satellite and aerial images for trees and palms recognition.

We think our choice has been the best compromise between what we believe could have real life useful applications and what our limited computational resources would allow us to obtain.

For instance, using larger datasets was never an option, due to GPU and RAM saturation in our machines. Despite that, we’re very happy with the results obtained and stand by their validity, knowing that a better training on more computationally performant devices could greatly benefit our models.

## **1.3 Model Testing**

We have trained one model for each Dataset, all available in the folder “Trained\_Models”. We selected a small test set for running inference in each Jupyter Notebook, but any custom image or video can be used for running “prediction()” (more on that later). We also added 5 videos to perform object detection on. In the “Videos” folder, there’s also the prediction to those same input videos.

The outputs of the model are post-processed by default, with the predicted bounding boxes already added to the image/video.

# **2. Code Structure**

All Jupyter Notebooks have the same structure, that is here summarized:

1. **Imports from Libraries:**

Run all imports that are useful in the scripts, including installing Supergradients.

1. **Google Drive Mounting:**

Used to quickly retrieve datasets and pre-trained models from this Google Drive Folder

1. **Preparing Dataset:**

Defining hyperparameters and dataset folders before training

1. **Training Model:**

Train the selected YOLO-NAS model (S, M, L)

1. **Saving Trained Model:**

Export to Google Drive a trained model for further importing it

1. **Importing pre-trained model:**

Import a pre-trained model that is ready for testing.

1. **Image Testing:**

using model.predict(...), takes a folder with images as input and produces images with bounding boxes as output

1. **Video Testing;**

using model.predict(...), takes a video as input and produces a video with bounding boxes as output

1. **Auxiliary Functions:**

Functions not strictly related to the Object detection task.

## **WARNINGS:**

1. All paths to folders defined on Jupyter Notebooks are **relative to the Google Drive folder “MyDrive”**. Therefore, it is highly recommended to right click on the folder “Project\_Intelligent” and **add a shortcut to “MyDrive”**. Otherwise all paths must be changed to run the scripts.
2. When running the first command, which is **!pip install super-gradients**, the Google Colab environment might require to **reboot the session** before going on with the rest of the code. This is due to some conflictual dependencies . Once that cell is executed, follow the instruction on the screen, reboot the session as indicated, then proceed executing the rest of the code. **Ignore the error at the end of the execution of the cell**, since it's factually irrelevant.
3. All models available to import are CUDA-optimized. Therefore, they **won’t run on non-CUDA devices without proper conversion**.
4. Training large models, especially on large datasets, is computationally expensive. Beware of RAM and GPU capabilities on the device in use.
5. By default, the model used for testing is the imported one. Therefore, prediction is run as model\_imp.predict(...). When training your own model, change that line of code to model.predict(...).

# **3. Conclusions**

We believe that the results obtained from the models are quite promising. Remember that, due to computational limitations, we could only train the YOLO-NAS models on small datasets and only for a limited number of epochs, due to RAM and GPU saturation on Colab’s runtime environment.

All things considered, object detection is quite accurate and fast for all the classes defined, with higher accuracy in classes such as cars that find more examples in the training set.

That’s why we believe that a Dataset change or extension could be extremely beneficial in terms of performance.

However, it’s fair to highlight the weakest points in our model as well. First of all, models struggle to identify larger objects, both for the algorithm implementation and for the nature of the training images.

Secondly, due to “limitations” in the datasets, the models struggle to detect some particular class of objects. For instance, docked boats are very hard to detect for the Sky-Fusion model, since the training images only contain offshore boats.