GREAT BARRIER REEF - OBJECT DETECTION MODEL

Data Mining M Project
October 2023
De Cecco Jasmin

INTRODUCTION

Australia's beautiful Great Barrier Reef is the world's largest coral reef with:

- 1,500 species of fish
- 400 species of corals
- 130 species of sharks
- Rays

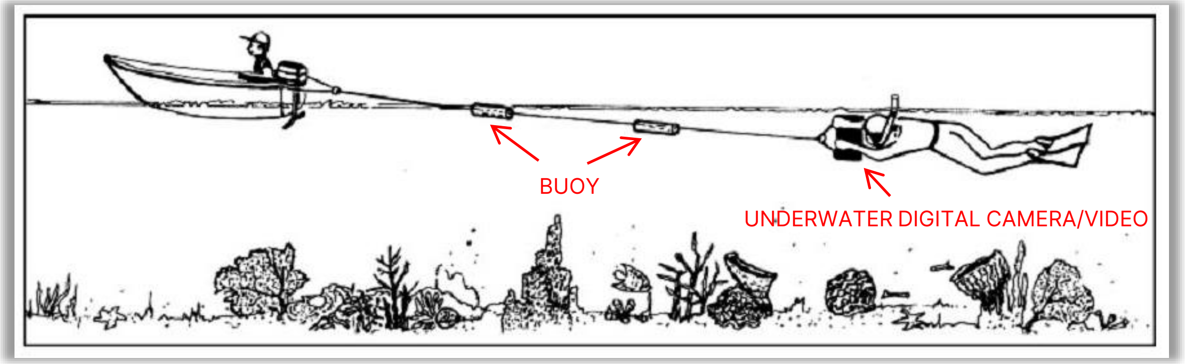
and a massive variety of other sea life.

Unfortunately, the reef is under threat, in part because of the overpopulation of one particular starfish: the **coral-eating crown-of-thorns starfish** (or COTS for short).

Scientists, tourism operators and reef managers established a large-scale intervention program to control COTS outbreaks to ecologically sustainable levels.

GOAL & REASONS

"Manta Tow“: traditional reef survey method performed by a snorkel diver to map COTS.



Effective but with some limitations:

- operational scalability
- data resolution
- Reliability
- traceability

The Great Barrier Reef Foundation established an innovation program to improve COTS Control exploiting AI strengths.

Australia’s national science agency, CSIRO has teamed up with Google to develop innovative machine learning technology.

Goal: object detection model model trained on underwater videos of coral reefs in order to identify starfish in real-time .

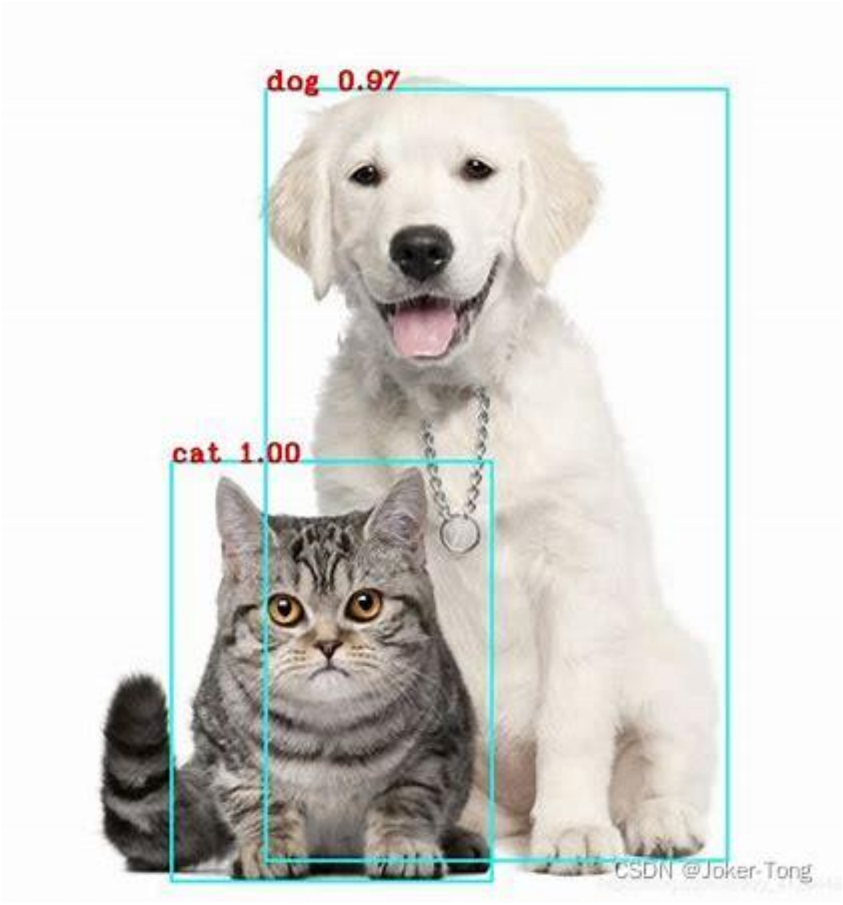
With a system that can analyse large image datasets accurately, efficiently, and in near real-time researchers will improve their work.

OBJECT DETECTION MODEL

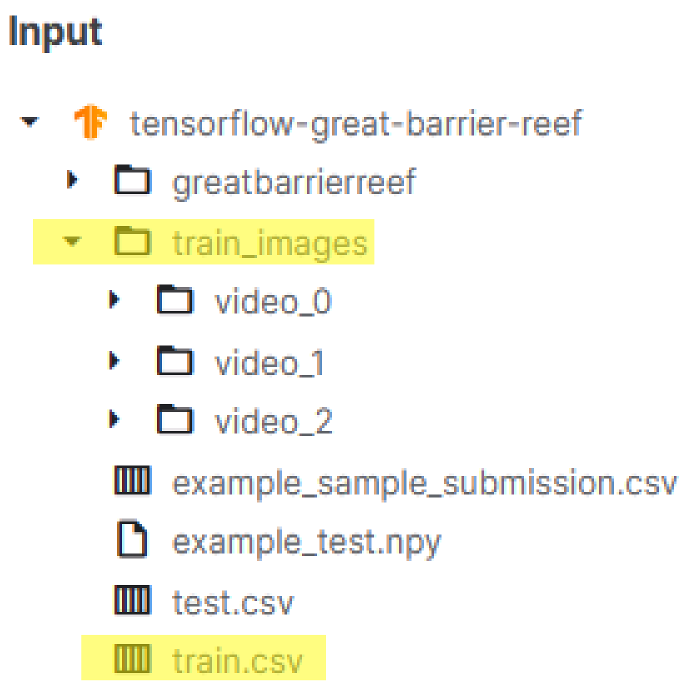
Input:
sequences of underwater images taken at various times and locations around the Great Barrier Reef

Output:
prediction of presence and position of crown-of-thorns starfish as bounding box with a confidence score for each identified starfish.

NOTE
The model won't be applied on a test set since it is part of a competition that uses a hidden test set available only once the notebook is submitted and scored.



INPUT DATA



train_images/
Folder containing training set photos of the form video_{video_id}/{video_frame_number}.jpg.

train.csv , test.csv
Metadata for the images. As with other test files, most of the test metadata data is only available to your notebook upon submission

example_sample_submission.csv
A sample submission file in the correct format. The actual sample submission will be provided by the API; this is only provided to illustrate how to properly format predictions.

example_test.npy
Sample data that will be served by the example API.

greatbarrierreef
The image delivery API that will serve the test set pixel arrays. You may need Python 3.7 and a Linux environment to run the example offline without errors.

EXPLORING DATASET: train.csv

- **video_id** - ID number of the video the image was part of. The video ids are not meaningfully ordered.
- **sequence** - ID of a gap-free subset of a given video. The sequence ids are not meaningfully ordered.
- **video_frame** - The frame number of the image within the video, there could be occasional gaps in the frame numbers.
- **sequence_frame** - The frame number within a given sequence.
- **image_id** - ID code for the image, in the format '{video_id}-{video_frame}'
- **annotations** - The bounding boxes of any starfish detections in a string format that can be evaluated directly with Python. A bounding box is described by the pixel coordinate (x_min, y_min) of its upper left corner within the image together with its width and height in pixels.

```
train.head()
```

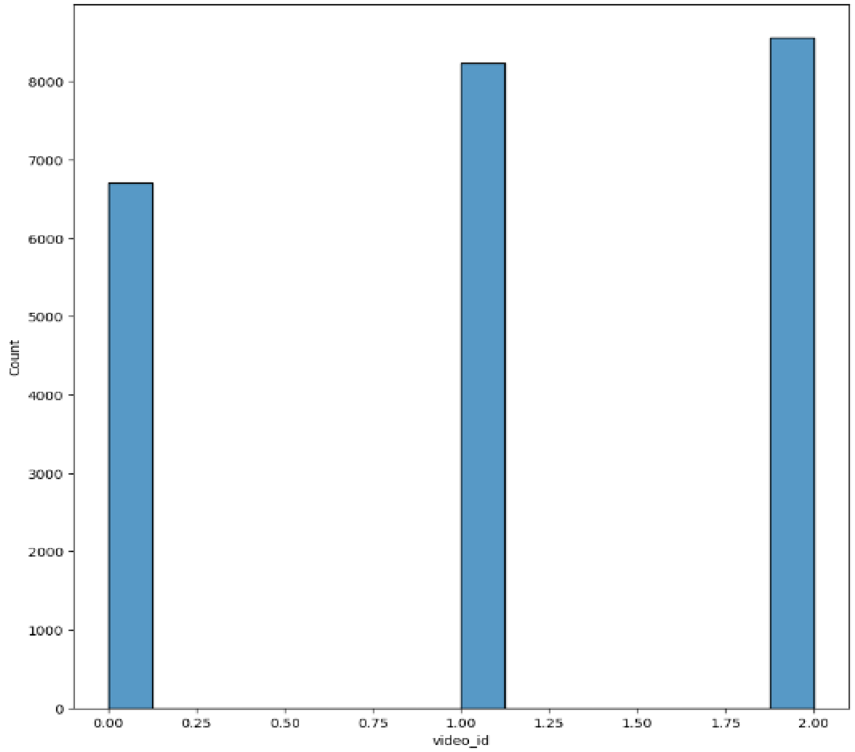
	video_id	sequence	video_frame	sequence_frame	image_id	annotations
0	0	40258	0	0	0-0	[]
1	0	40258	1	1	0-1	[]
2	0	40258	2	2	0-2	[]
3	0	40258	3	3	0-3	[]
4	0	40258	4	4	0-4	[]

```
train.tail()
```

	video_id	sequence	video_frame	sequence_frame	image_id	annotations
23496	2	29859	10755	2983	2-10755	[]
23497	2	29859	10756	2984	2-10756	[]
23498	2	29859	10757	2985	2-10757	[]
23499	2	29859	10758	2986	2-10758	[]
23500	2	29859	10759	2987	2-10759	[]

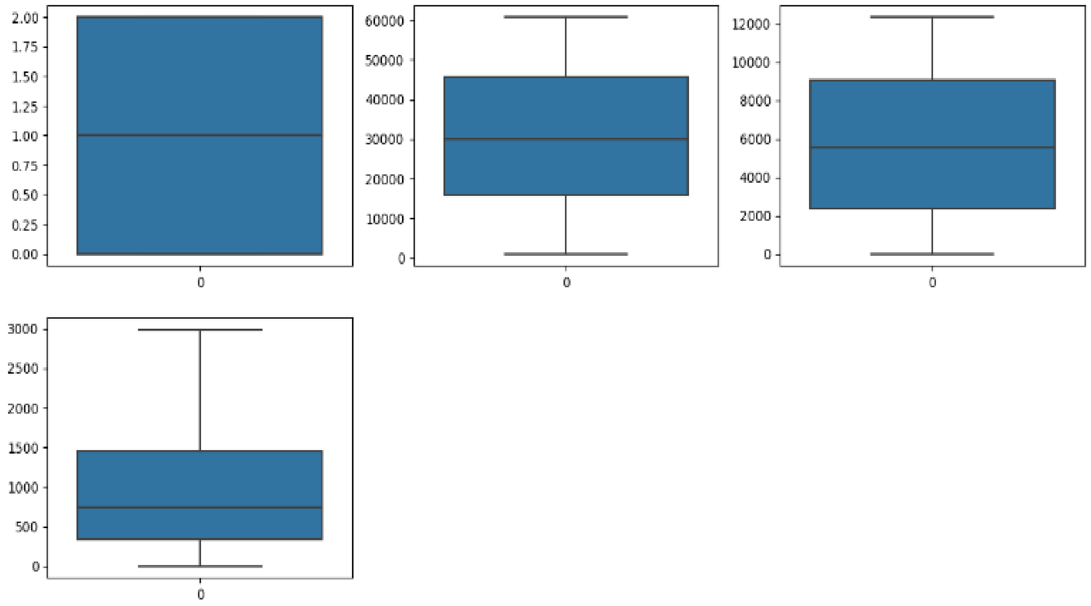
EXPLORING DATASET: distribution

Image distribution per video



video 0 video 1 **video 2**

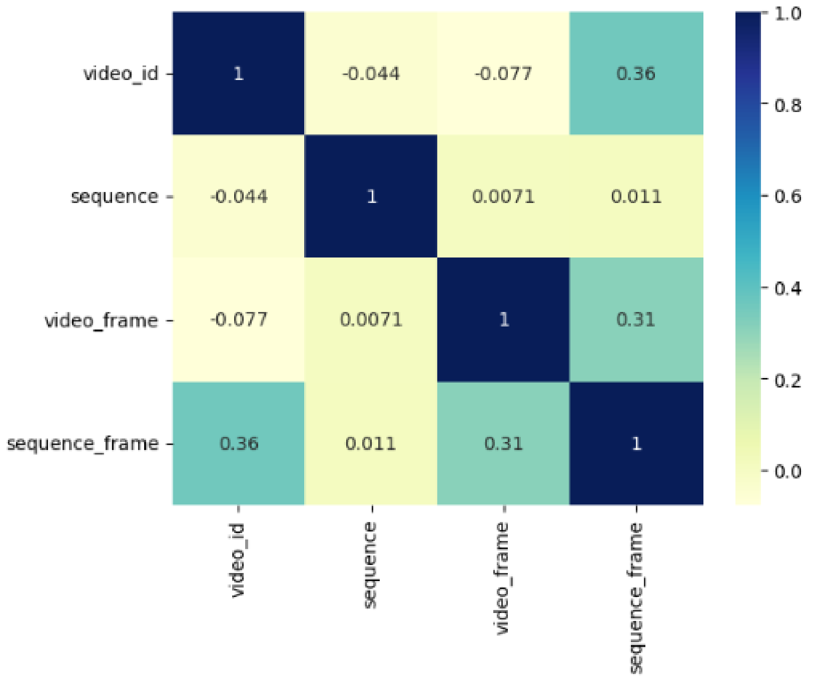
Boxplot



There are no outliers that will fit poorly in the model: no anomaly will badly affect the model

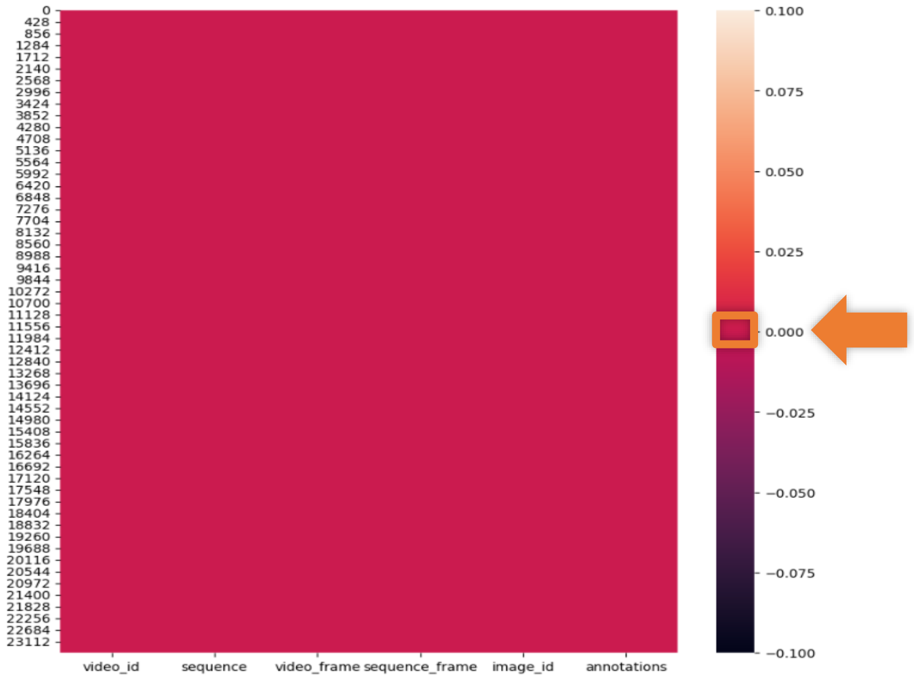
EXPLORING DATASET: cleaning

Correlation Matrix



No noticeable correlation detected:
no columns to drop

Null Values



There is any null value detected

EXPLORING ANNOTATIONS

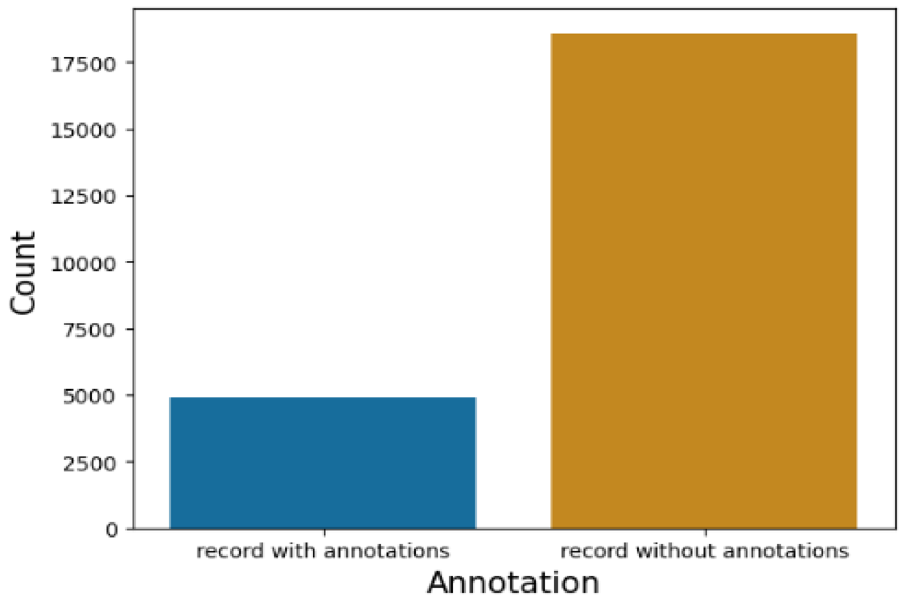
- Annotations have the following format:

[{'x': 645, 'y': 182, 'width': 41, 'height': 45}]

	video_id	sequence	video_frame	sequence_frame	image_id	annotations
16	0	40258	16	16	0-16	[{'x': 559, 'y': 213, 'width': 50, 'height': 32}]
17	0	40258	17	17	0-17	[{'x': 558, 'y': 213, 'width': 50, 'height': 32}]
18	0	40258	18	18	0-18	[{'x': 557, 'y': 213, 'width': 50, 'height': 32}]
19	0	40258	19	19	0-19	[{'x': 556, 'y': 214, 'width': 50, 'height': 32}]
20	0	40258	20	20	0-20	[{'x': 555, 'y': 214, 'width': 50, 'height': 32}]

- There can be multiple annotations for an image
- Not all records contain annotations
- Empty annotations are represented just by ‘**[]**’
- They are important for bounding boxes and train the model

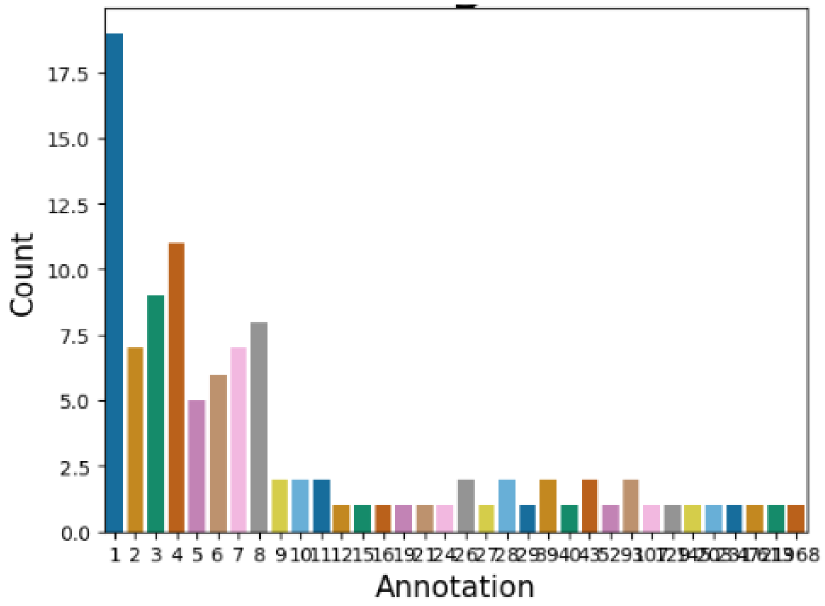
Records Annotation Distribution



There are 18582 records without annotations and 4919 records with annotations.

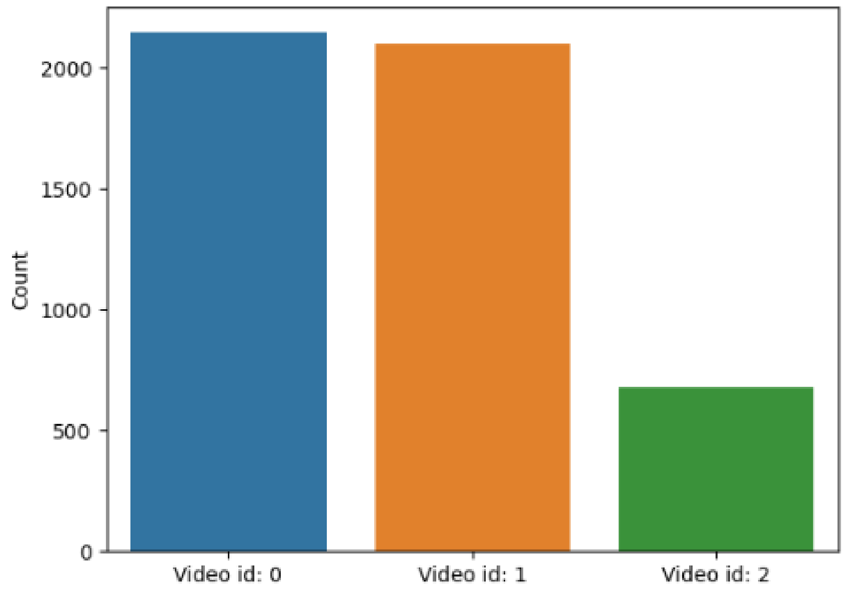
EXPLORING ANNOTATIONS: distribution

Annotation Length Distribution



- Most records have one or few annotations
- The maximum number of annotation for a record is 18

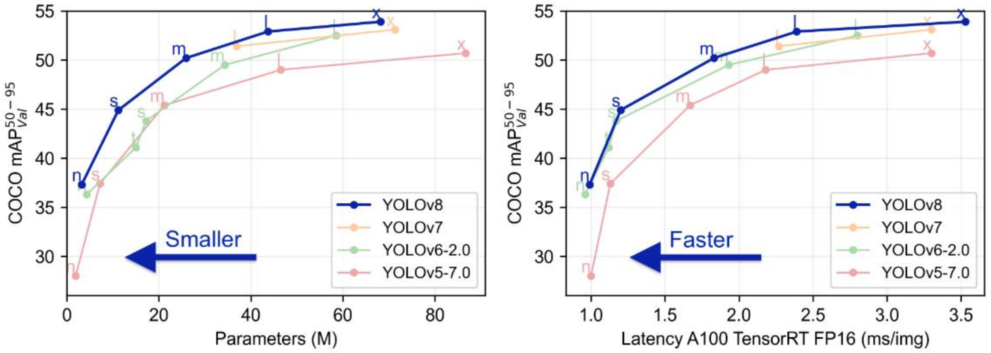
Annotation Distribution per video

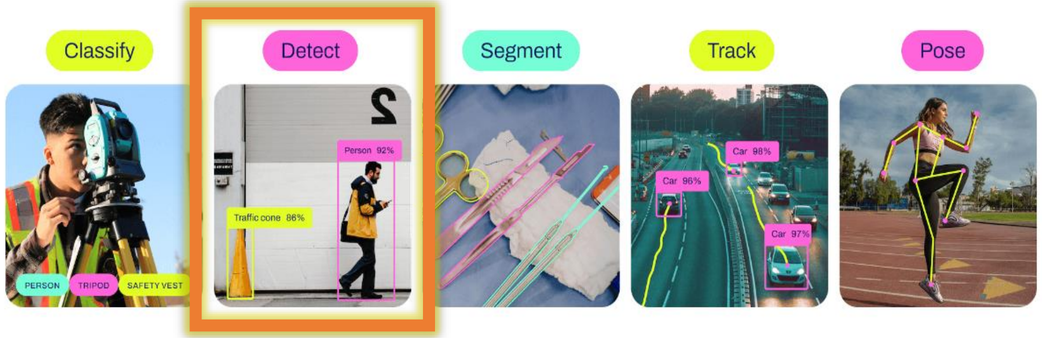


Even if video_2 has the biggest number of images, it shows to be the one with less annotations

OBJECT DETECTION MODEL: Yolov8

- YOLO (You Only Look Once), a popular object detection and image segmentation model first launched in 2015
- YOLOv8 is the latest version of YOLO by Ultralytics: new features, improvements for enhanced performance, flexibility, and efficiency.
- YOLOv8 supports multiple computer vision tasks: **detection**, **segmentation**, **classification**, and **pose estimation**.
- Ultralytics YOLOv8 supports several modes:
  - **Train**: training a YOLOv8 model on a custom dataset.
  - **Val**: validating a YOLOv8 model after it has been trained.
  - **Predict**: predictions with trained YOLOv8 model on new images/videos.
  - **Export**: exporting a YOLOv8 model to a specific format for deployment.
  - **Track**: tracking objects in real-time using a YOLOv8 model.
  - **Benchmark**: benchmarking YOLOv8 exports speed and accuracy.





Yolov8: data preparation

To use the model on custom data it needs **annotated data** with bounding boxes and following setting:

- Yolo format annotations

bounding box is described by the pixel coordinate from its upper left corner (x, y, w, h)

(object-class-ID, X center, Y center, Box width, Box height)

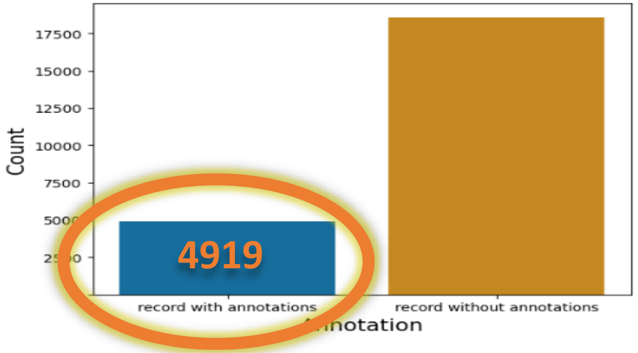
- Input directories called «**images**» and «**labels**»

  - **images**: *name_of_image_with_annotation*.jpg

create directory ***local_path*/images/train** → select only data with annotations → transfer from **/train_images** to the new directory

  - **labels**: *name_of_image_with_annotation*.txt

create directory ***local_path*/labels/train** → write each annotation in a .txt → save in the new directory




```
Image real path: /kaggle/input/tensorflow-great-barrier-reef/train_images/video_0/100.jpg
Image yolo path: /kaggle/working/images/train/100.jpg
Annotations: [{'x': 276, 'y': 631, 'width': 116, 'height': 88}]

Label yolo path: /kaggle/working/labels/train/100.txt
Annotations yolo format: 0 0.260937 0.937500 0.090625 0.122222
```

TRAINING THE MODEL

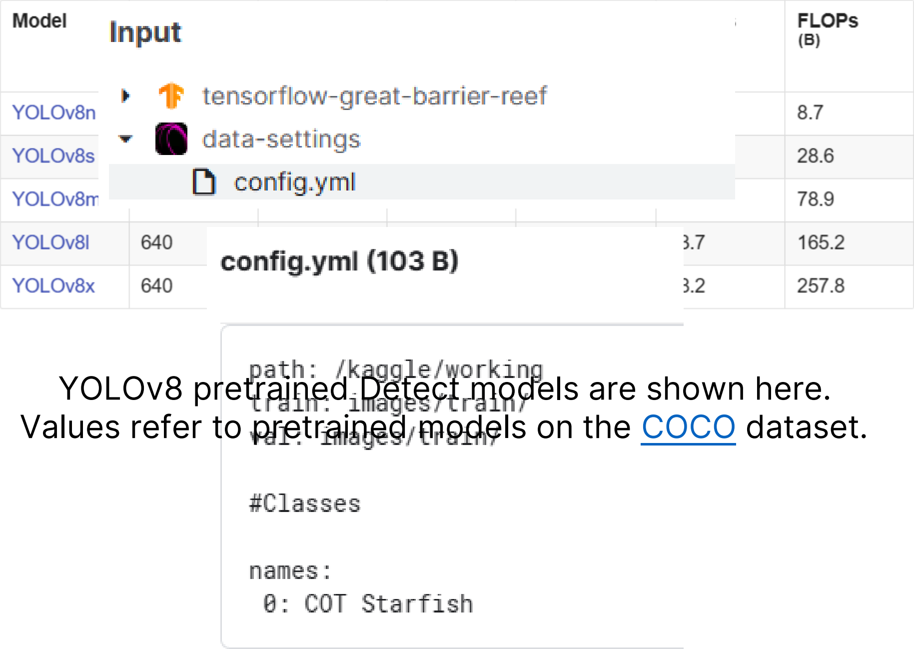
1. Install Ultralytics

```
!pip install ultralytics
import ultralytics
ultralytics.checks()
```

2. Load and train the model on custom data

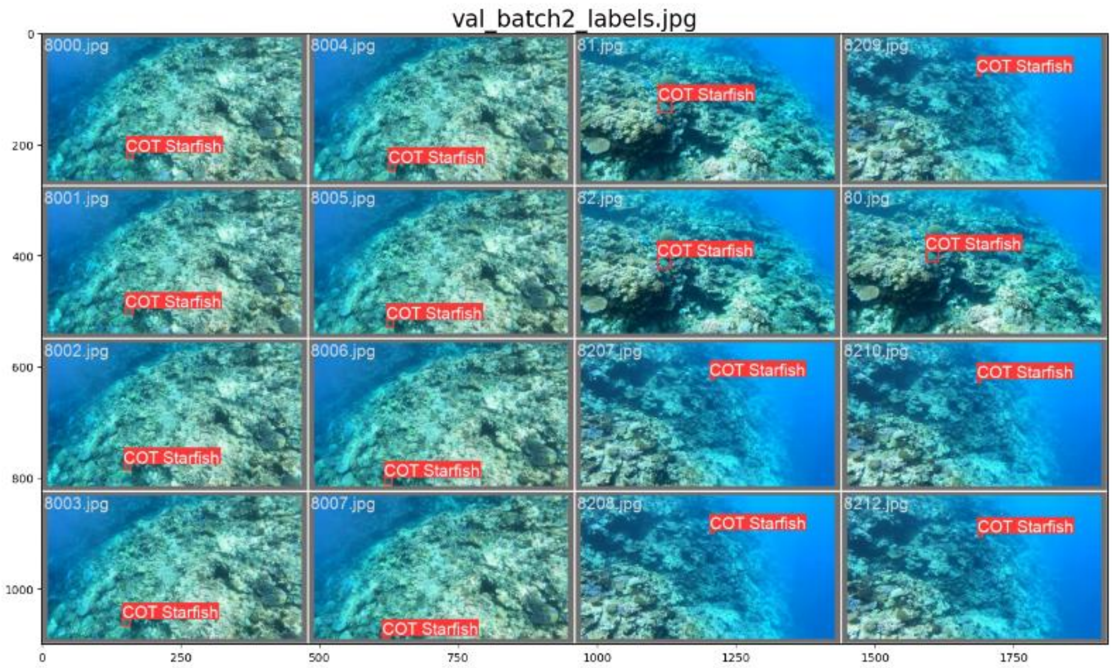
```
from ultralytics import YOLO

#Load model
model = YOLO("yolov8n.yaml") #build a new model from scratch

#Use the model
results = model.train(data = "/kaggle/input/data-settings/config.yml" epochs=100) #train the model
```

YOLO .yaml file allows to define the dataset root directory (**path**), the relative paths to training/validation/testing image directories and a dictionary of class names (**ID: class_name**)

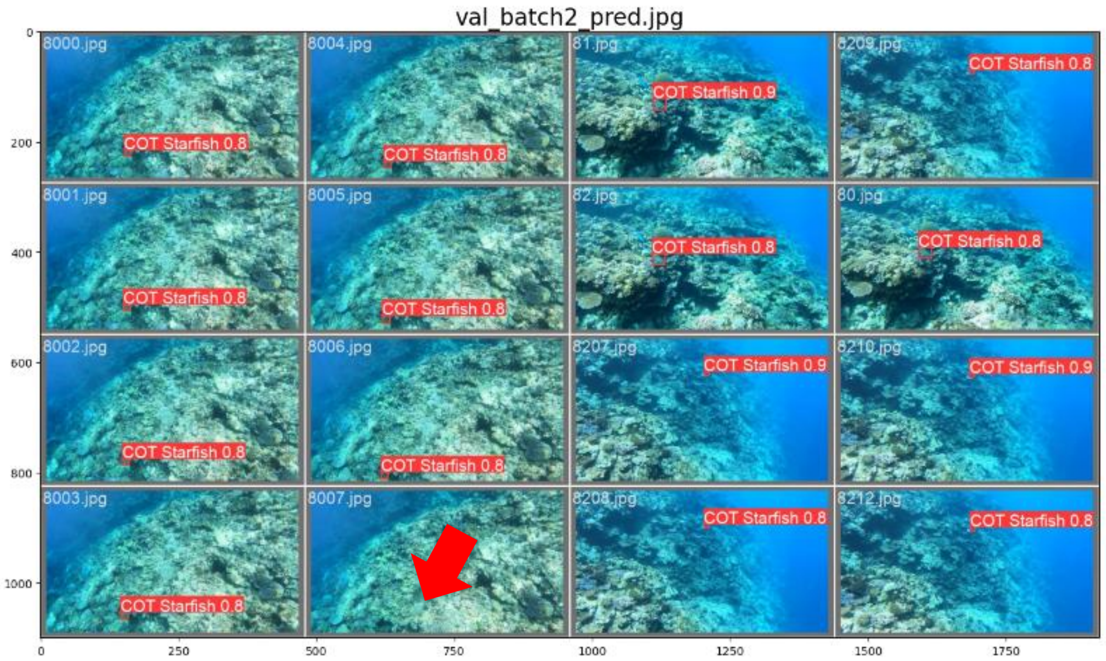


RESULTS: images batch

Specifics:
Kaggle platform - yolov8n model - no agumentation - training for 100 epochs

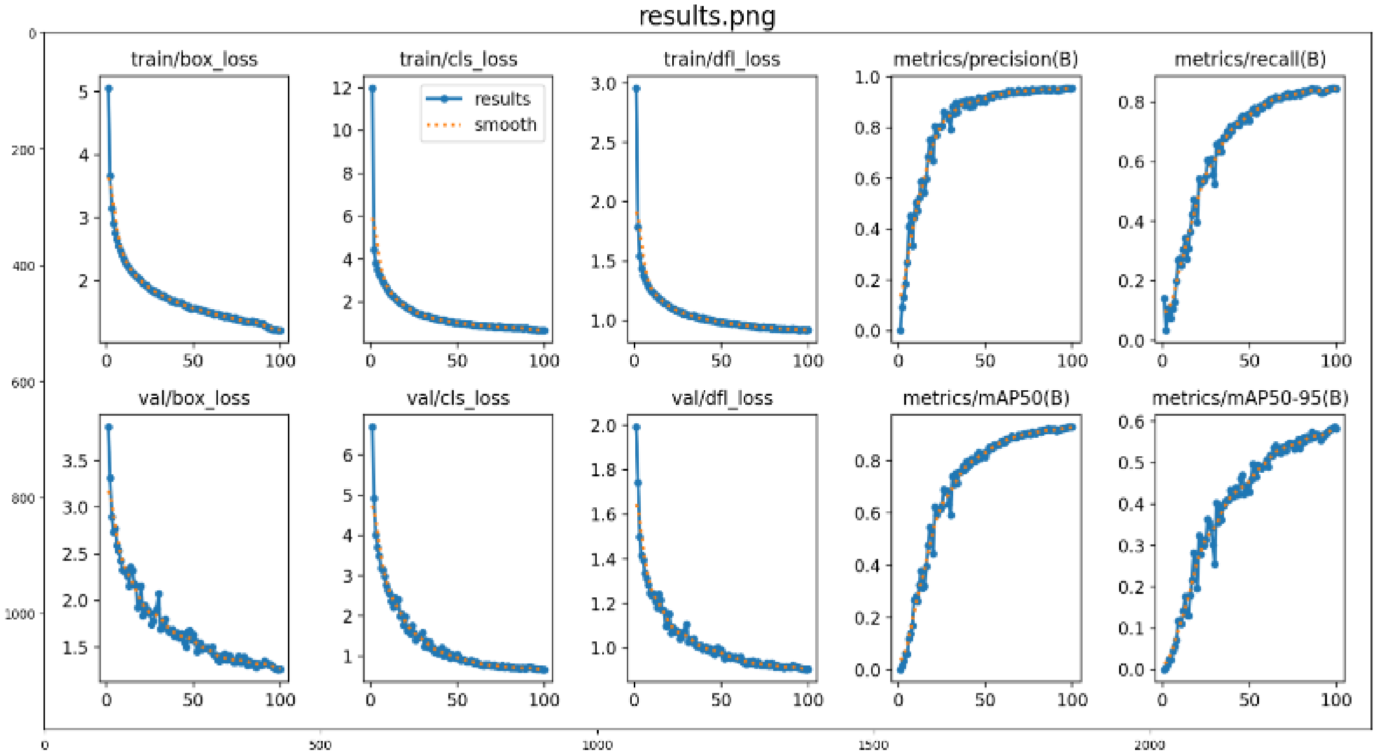


labels



predictions

RESULTS: loss, precision, recall, mAP50



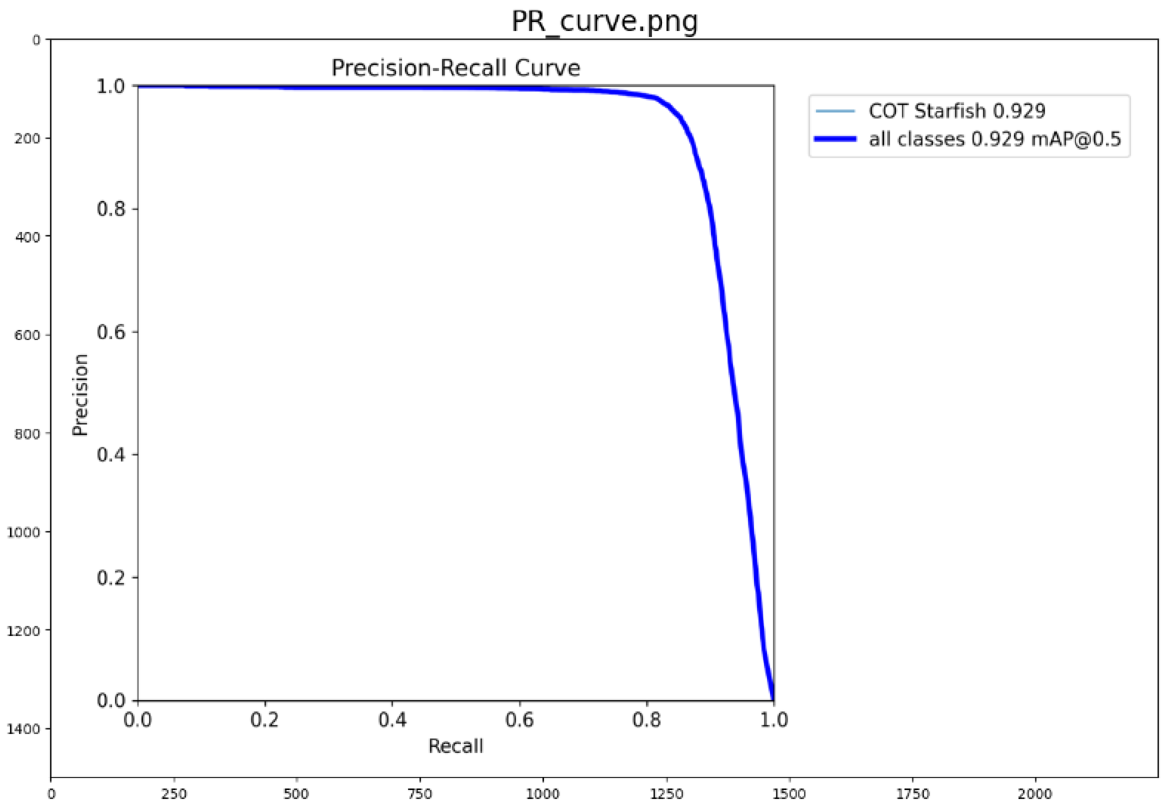
mAP50-95:

The mAP50–95 is the mean average precision computed with an IoU of 50%, 55%, 60%, 65%, 70%, 75%, 80%, 85%, 90%, 95% and then averaged.

Less good than the previous one: **0.58**

This mean that the model is pretty good at detecting all the COTs in our validation images (mAP50 = 0.93) but is not so good at finding the perfect bounding box (mAP50–95 = 0.58).

RESULTS: precision-recall curve



The **precision** and **recall curve** depicts the tradeoff between precision and recalls for various thresholds:

- A high area under the curve indicates both high recall and high precision.
- If the scores are high for both, it indicates that the classifier is yielding accurate findings (high accuracy) and a majority of all positive results (high recall).
- A perfect model is shown at the point (1, 1), indicating perfect scores for both precision and recall.

Good model: mAP of **0.929**
It correspond to the area under the curve

COMMENTS

No actual evaluation of the model:

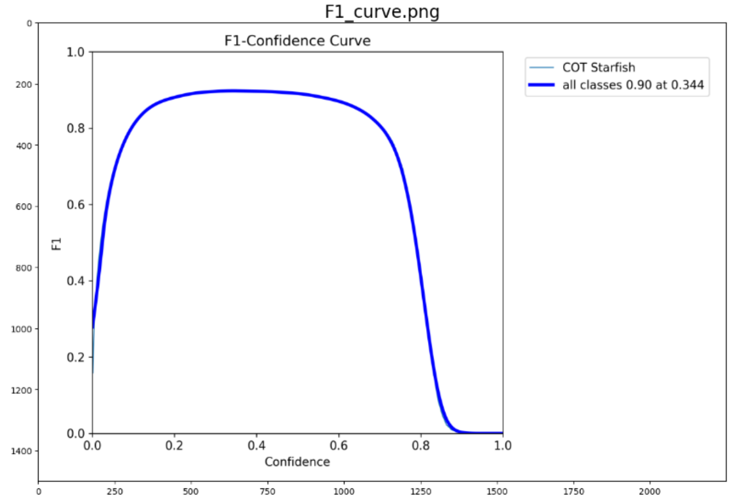
- No validation set
- No test set

To improve the model many things can be tried:

- training with other Yolov8 versions
- augumentaton (using Yolo settings or self made functions)
- setting confidence threshold by evaluating F1 curve

Name	Type	Default	Description
`source`	`str`	`'ultralytics/assets'`	source directory for images or videos
`conf`	`float`	`0.25`	object confidence threshold for detection

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8



USEFUL LINKS

- Yolov8 information: [Home - Ultralytics YOLOv8 Docs]
- Kaggle project: [Great Barrier Reef Object Detection | Kaggle]
- Github project: [jasmindc/DeCecco-ObjectDetection-Project (github.com)]


THANK YOU!