PREFACE

M.I.A. Robotics is a **30-person** company of interdisciplinary students from Alexandria University which was originally founded in 2011. This being its **11th** year at the MATE ROV Competition, the company has accumulated expertise in the field of underwater robotics and Machine learning over the years. With a steady pace towards improving the performance of each ROV.

There are numerous phyla that encompass a wide range of marine organisms. These phyla include diverse groups of animals, plants, and other organisms that inhabit marine ecosystems. Our mission to enhance the way we interpret deep ocean and gather more information by utilizing **AI** such as object detection.

1- INTRODCTION

The oceans cover more than 70% of our planet's surface, yet much of their depths remain a mystery to us. Ocean expeditions provide us with a unique opportunity to dive into this enigmatic realm and unravel its secrets. Scientists, researchers, and explorers from all corners of the globe have dedicated their lives to understanding the complex interplay between the ocean and the planet's ecosystems.

At the heart of ocean expeditions lies the pursuit of knowledge and understanding. From mapping the seafloor and studying marine life to investigating climate patterns and discovering ancient shipwrecks, these voyages offer a wealth of opportunities to expand our knowledge of the ocean and its countless inhabitants.

One of the important tools needed for these expeditions is the classifying the underwater species to gain more information about this classification we need to localize this classified species in the captured frame, this enable us to know more about the scene underwater by applying more advanced algorithms such as tracking to know about the species behavior.

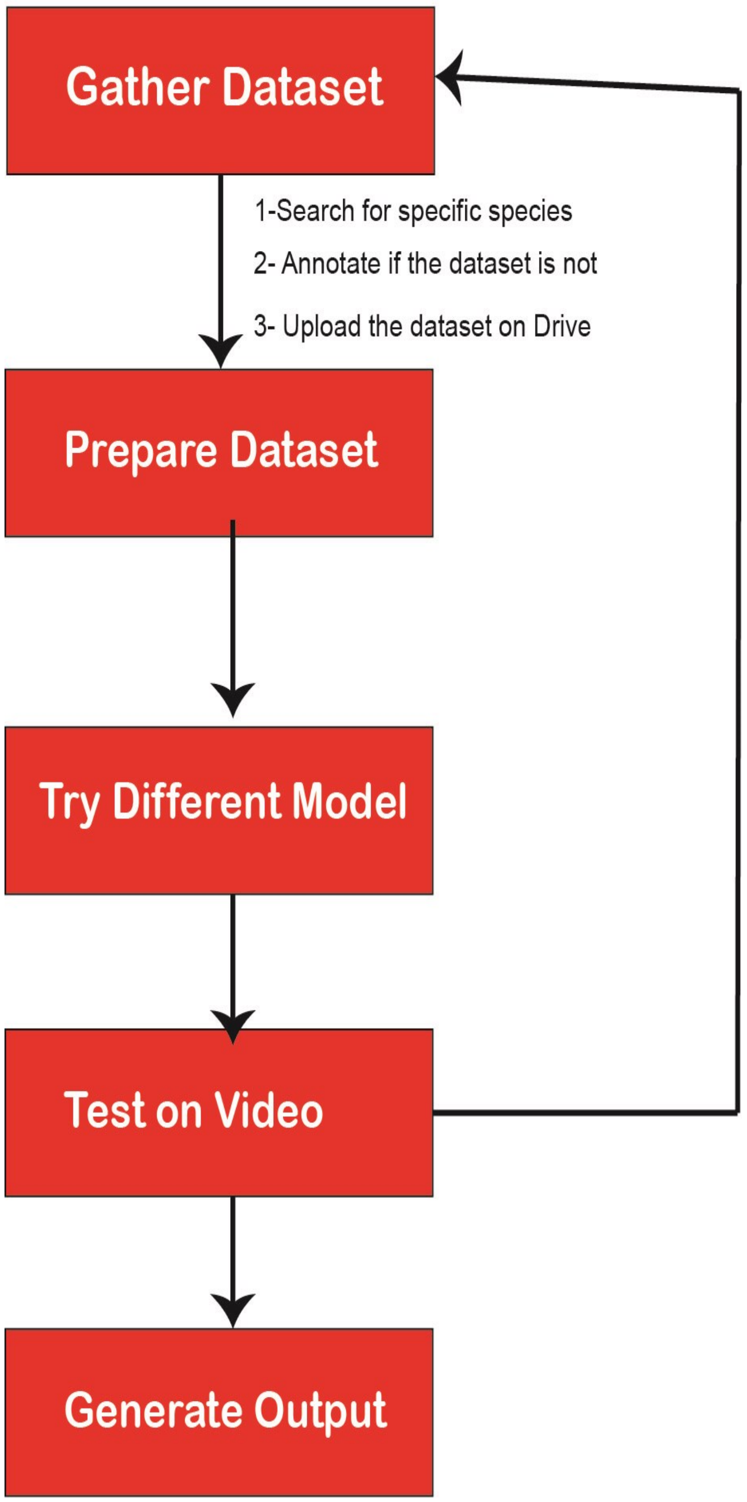
Our Object Detection model classifies underwater species by their unique physical characteristics, such as body shape, movement patterns, and distinguishing features like arms or tentacles. We have trained the model on a diverse dataset of marine life, ensuring it can accurately identify species like brittle stars (Ophiuroidea), fish, and other echinoderms. By focusing on key visual cues such as the central disk shape and the flexibility of the arms, our model can differentiate brittle stars from similar species.

2-PROCESS FLOW

Iterative Approach

The **iterative process**, also known as the agile process, emphasizes *flexibility*, *adaptability*, and *continuous improvement* throughout the project. This give as :

- **Flexibility:** The iterative process allows for changes and adjustments to be made throughout the project based on feedback, new insights, or evolving requirements.
- **Continuous feedback:** feedback is incorporated regularly, enabling the project team to address issues, refine models, and make improvements incrementally.
- **Rapid prototyping:** The iterative process often involves building and refining prototypes or minimum viable products (MVPs) to gather feedback early on and validate model in early steps.
- **Collaboration:** Collaboration and communication among team members, are essential in the iterative process. Regular meetings and interactions facilitate transparency and alignment.
- **Quick response to changes:** The iterative process allows for quick adaptation to changes in requirements, emerging technologies, or unexpected challenges. This agility helps avoid the risk of being locked into a suboptimal solution.


Gather Dataset
1-Search for specific species
2- Annotate if the dataset is not
3- Upload the dataset on Drive
Prepare Dataset
Try Different Model
Test on Video
Generate Output

3-Gather Dataset

Gathering a dataset for object detection models requires careful consideration of various factors to ensure that it covers a wide range of objects, perspectives, and scenarios. Here is a step-by-step guide to help you gather a dataset for object detection:

3.1- Identify data sources

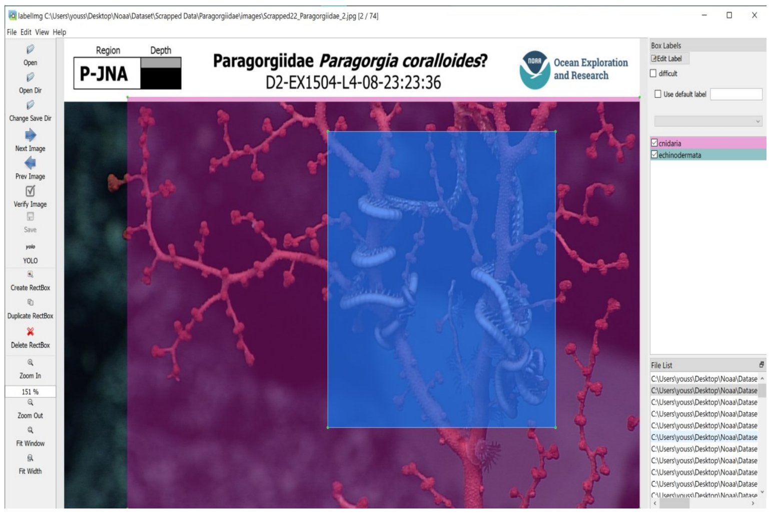
- We tried to collect dataset that have the same distribution as validation videos, then we decide to focus on videos of **NOAA Expatiations**. Also we collect from other sources to achieve generalization.

- We also collect data through **web scrapping** to automate process of gathering dataset

3.2- Annotation

- Object detection datasets require **bounding box annotations** that define the location and size of objects within the images. There are various annotation tools available,

- We decide to use labelImg for easy usage and for supporting JSON format.


labelImg C:\Users\youss\Desktop\Noaa\Dataset\Scrapped Data\Paragorgiidae\images\Scrapped22_Paragorgiidae_2.jpg [2 / 74]

Paragorgiidae *Paragorgia coralloides*?
D2-EX1504-L4-08-23:23:36

Box Labels: ☑ cnidaria, ☑ echinodermata

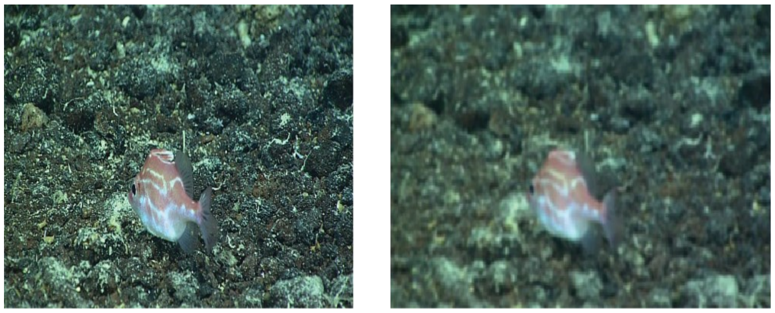
File List:
C:\Users\youss\Desktop\Noaa\Datase

3.3- Image augmentation

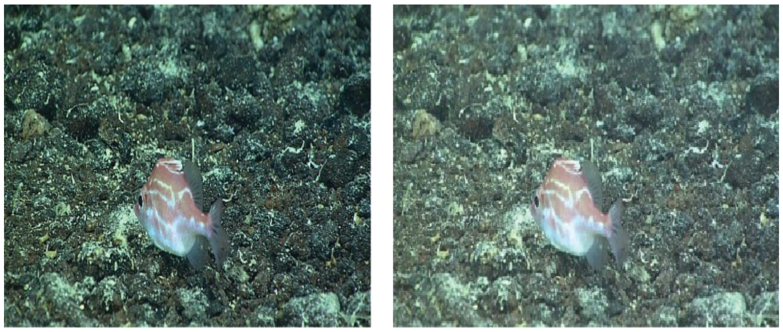
Consider applying data augmentation techniques to expand your dataset. This involves applying transformations such as **rotations**, **translations**, **scaling**, **flipping**, or **adding noise** to your existing images. Data augmentation helps to increase the diversity of your dataset and improve the model's robustness.

We apply image augmentation operation with **probability** portion as each operation needed.

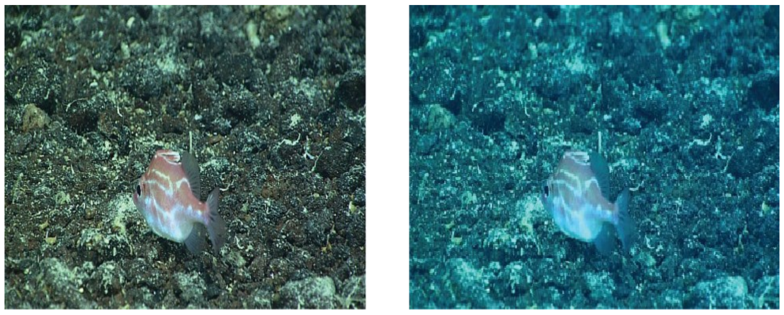
- **Blurring**



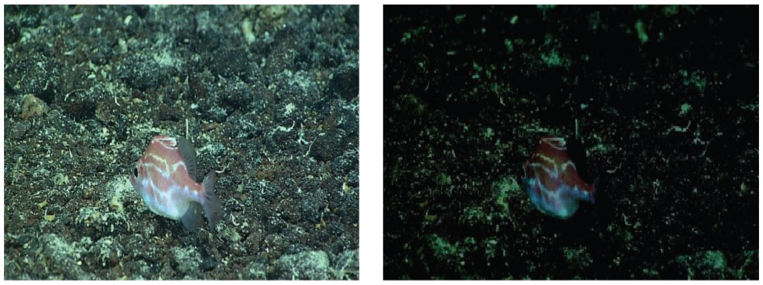
- **Brightness**



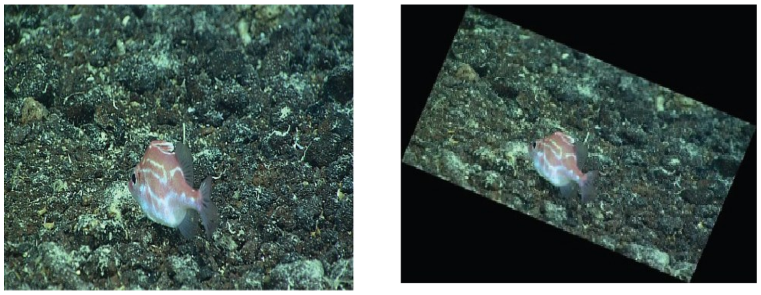
- **Hue Shift**



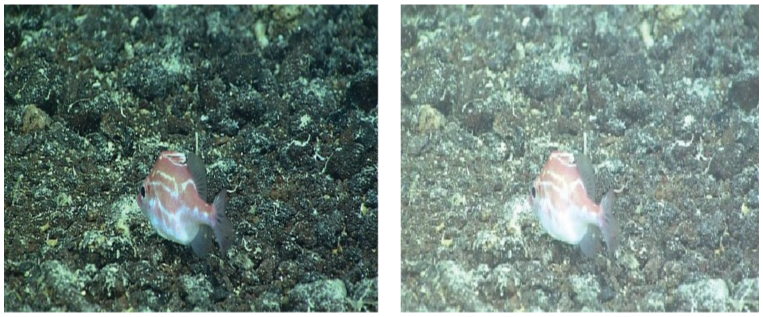
- **Darkness**



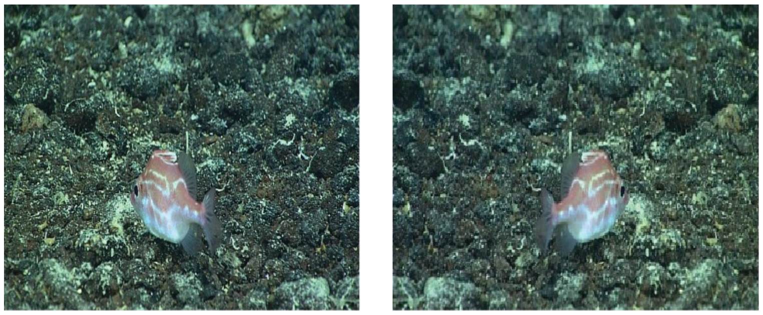
- **Rotate**



- **Noise**



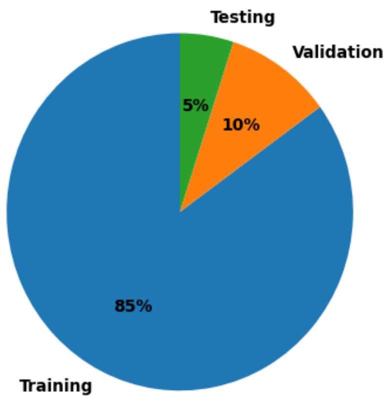
- **Horizontal Filp**



3.4- Splitting Dataset

Divide your dataset into appropriate subsets for training, validation, and testing. Typically, a significant portion is allocated for training (e.g., 70-80%), while the remaining portion is used for validation and testing. This division helps evaluate model performance and avoid overfitting.

Because we have relatively large dataset which is **17,500 image** we decide to split our data to 85% training dataset (14,875 image), 10% validation dataset (1,750 image), and 5% test dataset (88 image), for exploiting more data in training 3,000 image is sufficient for **hyperparameter tuning.**



3.5- Dataset Analysis

When we gather our dataset we take care to satisfy **diversity** in different levels, such as:

- **Diversity in source:** That is our dataset come from different sources to achieve generalization and the model become less prone to **overfit**.
- **Diversity in Size:** That is our dataset contain object of interest in different, and diverse sizes, this make our model could detect large as well as small objects.

We also could increase this diversity by **image augmentation**, but we need to tune this operation to become effective to our need.

We also take care of **randomization** that mean, to mean our model to be less biased, achieve that by:

- **Shuffle Dataset:** We shuffled our dataset to train and validate from the same distribution this make model less biased.
- **Proportion Augmentation:** We also tune proportion of each augmentation operation to ensure less bias dataset.
- **Cover vulnerabilities:** after each dataset version we try to cover it's vulnerability.

4- MODEL

4.1- Choose Framework

There are several popular object detection frameworks available that provide pre-built models, tools, and resources to develop and deploy object detection models efficiently. Here are some widely used object detection frameworks:

Most important two-stage object detection algorithms:

- RCNN and SPPNet (2014)
- Fast RCNN and Faster RCNN (2015)
- Mask R-CNN (2017)
- Pyramid Networks/FPN (2017)
- G-RCNN (2021)

Most important one-stage object detection algorithms

- YOLO (2016)
- RetinaNet (2017)
- YOLOv3 (2018)
- YOLOv4 (2020)
- YOLOR(2021)
- YOLOv7(2022)
- Yolov8(2023)

Transformer-based object detection algorithms:

DETR (2020)

DETR 2.0 (2022)

4- MODEL

4.3- RE-DETR

Why We Chose RE DETR from Ultralytics:

We decided to use RE DETR because it builds on the innovative DETR framework, which introduced transformers to object detection, but adds significant optimizations for real-time performance. Key features of RE DETR include:

Efficient Hybrid Encoder: RE DETR uses a hybrid encoder that decouples within-scale interactions from cross-scale feature fusion, allowing it to handle multi-scale features more effectively. This results in reduced computational costs while maintaining high accuracy.

IoU-Aware Query Selection: This feature improves the initialization of object queries by focusing on the most relevant objects in the scene. It enhances the model's ability to accurately detect and localize objects, even in complex environments.

Adjustable Inference Speed: Unlike many other models, RE DETR allows users to adjust the inference speed by modifying the number of decoder layers used during inference, without the need for retraining.

Real-World Applications: RE DETR is particularly well-suited for applications where both speed and accuracy are critical, Its ability to maintain high performance in real-time makes it an ideal choice for these demanding scenarios.

4.2- Training

We trained our RE DETR model for 100 epochs, with each epoch taking approximately 30 minutes on a T4 GPU on Google Colab. This extensive training allowed us to achieve a high mean Average Precision (mAP) and low loss values, ensuring the model's effectiveness in real-time object detection tasks.

The RE DETR model we used, rt-detr-l, is highly optimized with 673 layers, comprising 32,970,476 parameters. The model has a total of 0 gradients and requires 108.3 GFLOPs, making it computationally efficient while still delivering high accuracy. The detailed summary of the model parameters is as follows: 673 layers, 32,970,476 parameters, 0 gradients, 108.3437056 GFLOPs.

During training, it took approximately 20 hours to complete 50 epochs. However, this training duration was sufficient to reach an optimal level of performance. We used input images of 1024x1024 pixels to ensure the model could accurately detect objects across various scales, particularly small objects.

4.4- Deployment

A) Detection Source

we could deploy the model on different multi-media:

- Live stream from camera.
- Static stored video.
- Videos from YouTube.
- Folder of stored videos.
- Folder of stored images.
- Single image.

B) Detection Excel

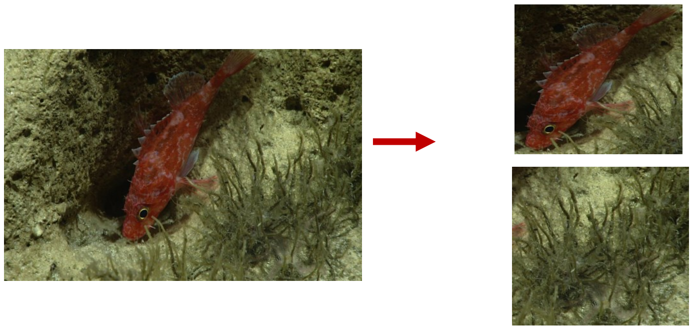
Our driver could generate a excel file that has the stream detections with the localization information.

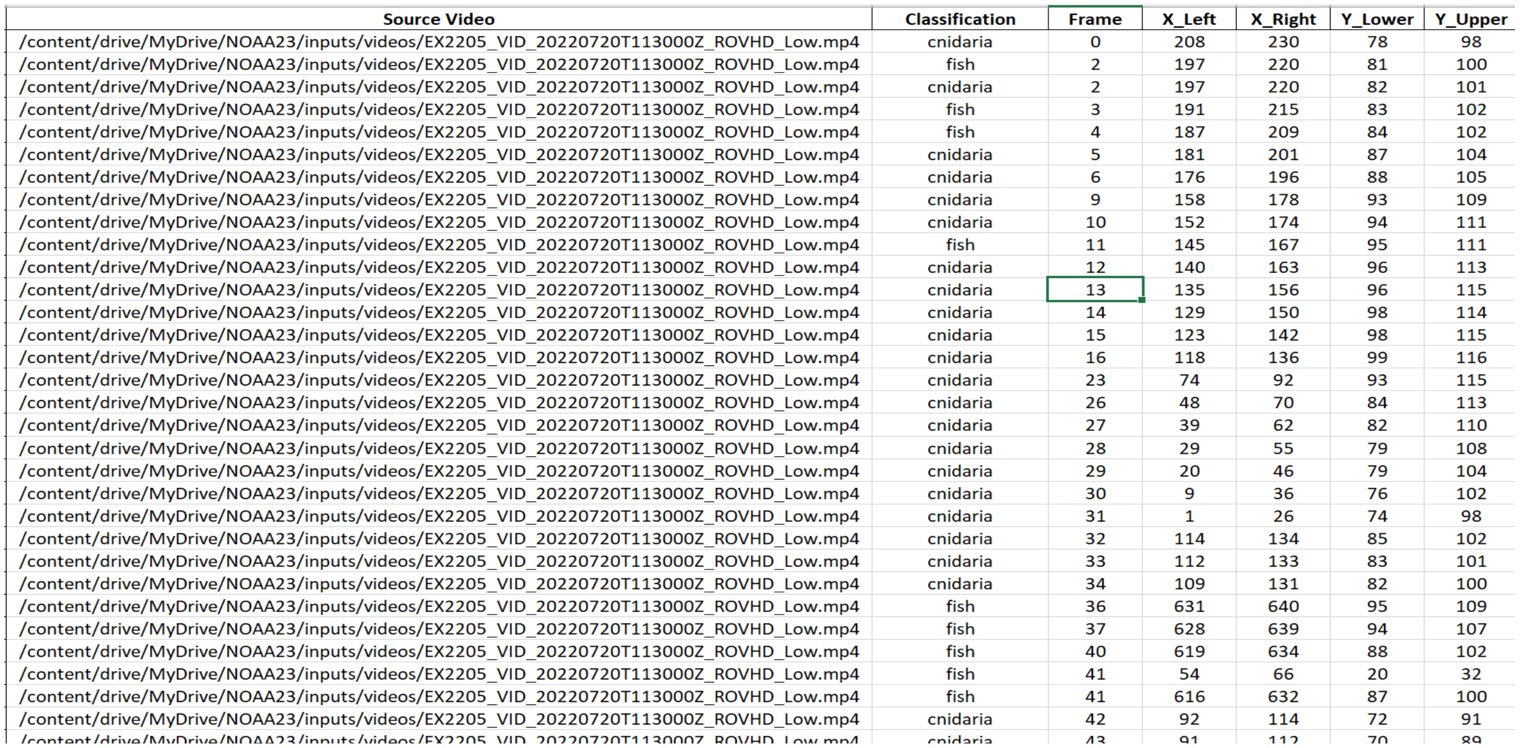
C) Detection Processing

Our driver could be deployed both on CPU or GPU using **CUDA** for that we support two model one is named **Accurate** which is large architecture for deployed real time on GPU, and other named **Fast** which is small architecture for run real time on CPU.

D) Cropped Bounding boxes

Our driver could extract cropped bounding box and save them.





Source Video	Classification	Frame	X_Left	X_Right	Y_Lower	Y_Upper
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	0	208	230	78	98
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	2	197	220	81	100
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	2	197	220	82	101
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	3	191	215	83	102
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	4	187	209	84	102
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	5	181	201	87	104
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	6	176	196	88	105
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	9	158	178	93	109
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	10	152	174	94	111
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	11	145	167	95	111
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	12	140	163	96	113
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	13	135	156	96	115
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	14	129	150	98	114
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	15	123	142	98	115
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	16	118	136	99	116
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	23	74	92	93	115
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	26	48	70	84	113
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	27	39	62	82	110
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	28	29	55	79	108
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	29	20	46	79	104
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	30	9	36	76	102
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	31	1	26	74	98
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	32	114	134	85	102
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	33	112	133	83	101
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	34	109	131	82	100
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	36	631	640	95	109
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	37	628	639	94	107
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	40	619	634	88	102
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	41	54	66	20	32
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	fish	41	616	632	87	100
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	42	92	114	72	91
/content/drive/MyDrive/NOAA23/inputs/videos/EX2205_VID_20220720T113000Z_ROVHD_Low.mp4	cnidaria	43	91	112	70	89

MIA robotics

USER MANUAL

2023

NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NOAA
U.S. DEPARTMENT OF COMMERCE

1- Command Line Interface (CLI)

We build an easy to use CLI to deploy the model with a set of feature to infer in different types of **inputs**, generate the needed **output**, and **display** the stream in suitable manner. Also this CLI not only for user interface but also it's the base backend of our **Web interface**

1.1– Arguments

CLI has many arguments to interact with and this arguments serve different aspect of deployment process such as take stream, and customize output. (most of this arguments have default values except one that have *).

Argument	Description
model	* Path to model used
input	* Path to input multimedia (e.g. --input inputs/video1.mp4)
output	Path to output folder that will have the generated output (if not exist it will be created)
device	Name of device used GPU/CPU (e.g. if CPU: --device cpu, If GPU: --device 0)
name	Name of the generated outputs such as annotated video and output excel
update_excel	The number of frames to update output excel (e.g. if each frame --update_excel 1, -1 if only at end)
conf	Confidence threshold of bounding box generated. (e.g. --conf 0.5)
iou	IOU threshold of Non-maximum suppression. (e.g. --iou 0.2)
classes	Path to classes file which is yaml file has the classes name (used to filter output classes also).
imgsz	Input frame size to the model (e.g. —imgsz 640 --> 384x640, —imgsz 1280 --> 736x1280)
show	To show annotated stream (e.g. --show --> to show stream)
gen_excel	To generate and save annotation excel file (e.g. --gen_excel --> to save excel)
save	To save annotated stream (e.g. --save --> to save stream)
save_crop	To save cropped bounding boxes of the stream (e.g. --save_crop --> to save cropped images).
hide_labels	To hide information of bounding box such as name and confidence for lite display (e.g. --hide_labels --> to Hide)
port	Port to stream on if needed (e.g. --port 5000 (to stream on 5000))

1.2– Multimedia Inputs

A) You can set input as **static video** file that exist in our local machine.

```
python  interface.py  --model <model_path>   --input <video_path>   --output  <folder_path>   --name <folder_name>  --show   --gen_excel
```

B) You can take input as **live stream** from connected device (e.g. --input 0)

```
python  interface.py  --model <model_path>   --input <camera_id>   --output  <folder_path>   --name <folder_name>  --show   --gen_excel
```

C) You can set input as **folder of videos** and the output will be the annotated images.

```
python  interface.py  --model <model_path>   --input <folder_videos>   --output  <folder_path>   --name <folder_name>  --show   --gen_excel
```

D) You can set input as **folder of images** and the output will be the annotated images.

```
python  interface.py  --model <model_path>   --input <folder_images>   --output  <folder_path>   --name <folder_name>  --show
```

E) You can also set input as **single image**, (e.g. inputs/images/image1.jpg).

```
python  interface.py  --model <model_path>   --input <image_path>   --output  <folder_path>   --name <folder_name>  --show
```

F) You can set input as **YouTube video** URL, (e.g. https://www.youtube.com/watch?v=5Ds1PxDoQag).

```
python  interface.py  --model <model_path>   --input <youtupe_url>   --output  <folder_path>   --name <folder_name>  --show   --gen_excel
```

1.3– Processing Unit

Deployment driver can run on either GPU or CPU. You can explicitly specify the device needed to run on or you can let the driver **automatically** choose the suitable device that run automatically on GPU if exists.

A) Deployment on **automatic** mode.

```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>  --device auto
```

B) Deployment on **GPU** specified device (e.g. --device 0)

```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>  --device <device_id>
```

C) Deployment on CPU.

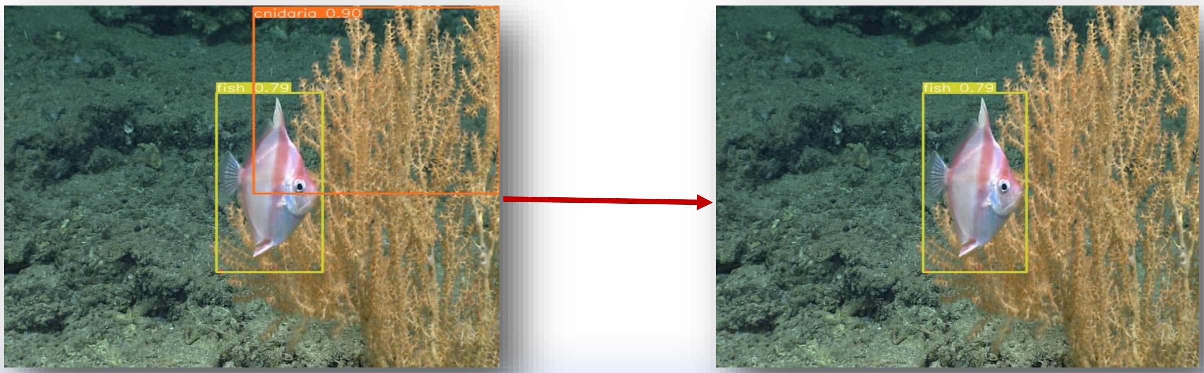
```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>  --device cpu
```

1.4– Classes Filter

You could also filter the output classes to specific classes, that mean we could exclude some of output classes, (e.g. we could annotate only one category such as fishes). You could control this filtration by **yaml** file which include name object that have the needed classes.

- Could also use this yaml file to change names of classes.

```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>   --classes fishes_only.yaml
```



1.5– Input Frame Size

We could also change the input frame size that will be inferred to suite our need, that is, if we need to detect very small objects we can make imgsz to be large to keep the information of the small objects.

- Large imgsz will need more computations and therefore the speed of deployment will be decreased.
- The default value of imgsz is 640 which make the input frame in shape of (**384x640**).
- Imgsz must be multiple of 32.

A) Infer with input shape (**736x1280**) to detect tiny objects.

```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>  --imgsz 1280
```

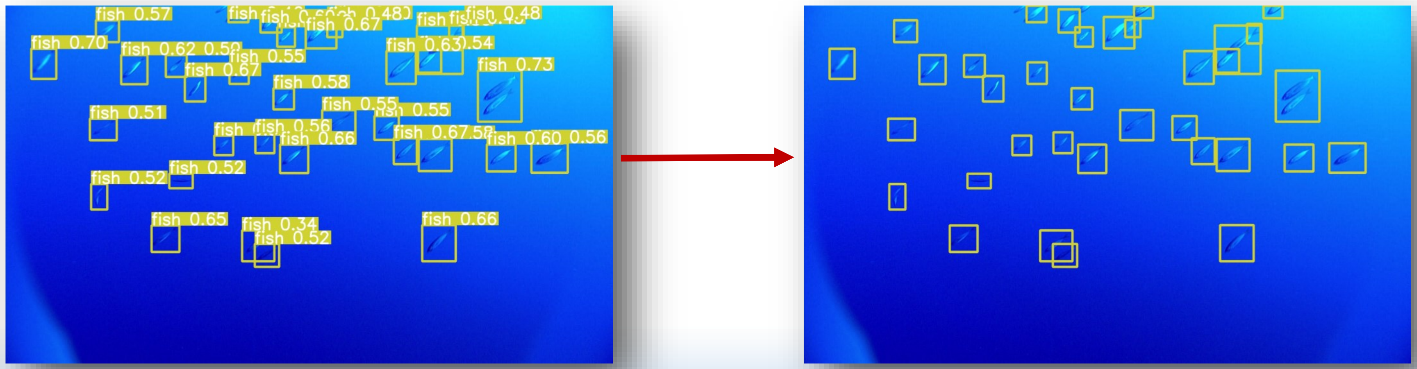
B) Infer with input shape (**320x320**) to very fast deployment.

```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>  --imgsz 320
```

1.6– Hide Labels

Sometimes when there are **multiple objects** in single frame it's well be hard to visualize the scene clearly due to the labels of the bounding box, so the driver support option to remove the label and show only the bounding boxes in which each class has distinct box color.

```
python  interface.py  --model <model_path>   --input <media>   --output  <folder_path>   --name <folder_name>  --hide_labels
```

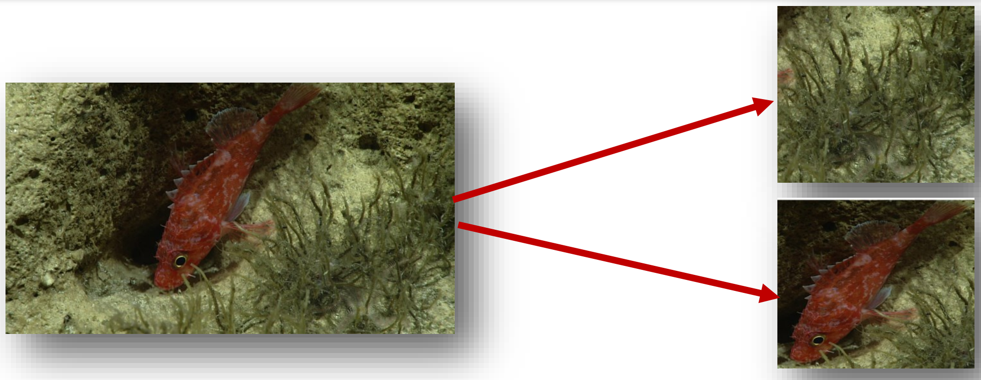


1.7– Save Cropped

You could also save the cropped images from bounding boxes this could be used in many thing such as object classification.

- The cropped images will saved in output/crop folder each class will have it's folder with the name of the class (e.g. output/crop/fish).

```
python interface.py --model <model_path>  --input <media>  --output  <folder_path>  --name <folder_name>  --save_crop
```

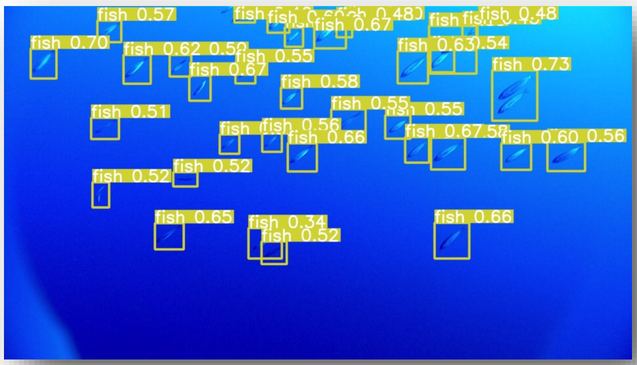


1.8– Confidence Threshold

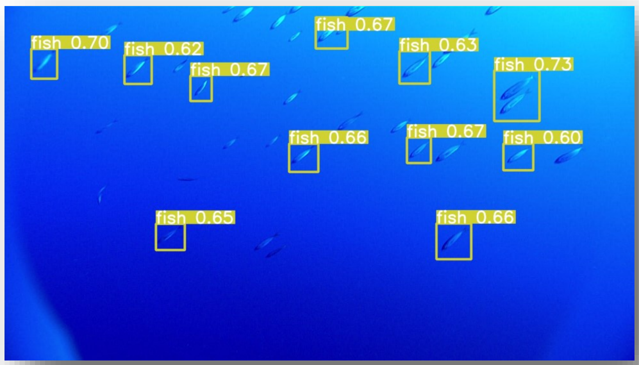
Confidence threshold parameter control the tradeoff between **Precision** and **Recall**:

- High conf (e.g. conf>0.5) make detections high precise by removing any bounding boxes that have confidence lower than 0.5, this reduce the noise but also reduce recall.
- Low conf (e.g. cong<0.5) make detection high recall by let many more bounding boxes, this increase probability of detecting object, but increase noise.

```
python interface.py --model <model_path>  --input <media>  --output  <folder_path>  --name <folder_name>  --conf <threshold_percent>
```



--conf 0.2

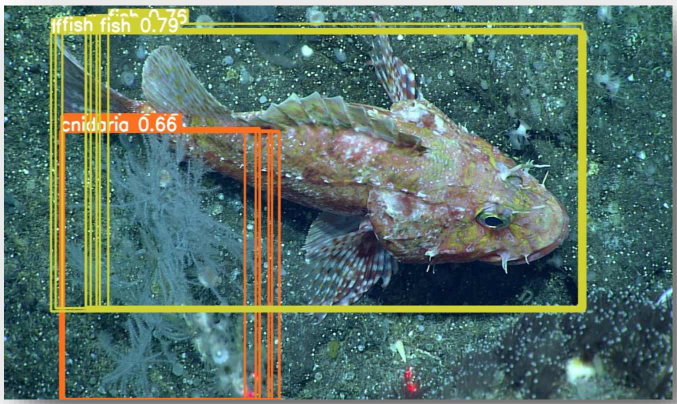


--conf 0.6

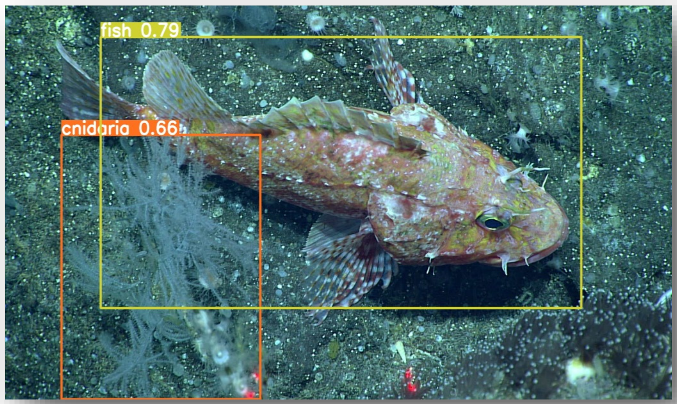
1.9– Non-maximum Suppression Threshold

We could also control the **NMS threshold** to be suitable to our application, if the environment has many adjacent object from the same, it's better to increase **iou**, this will make detector apply to detect multiple object from the same class with small distance between them but also increase the chance of **over annotate** single object.

```
python interface.py --model <model_path> --input <media> --output <folder_path> --name <folder_name> --iou <iou_thresold>
```



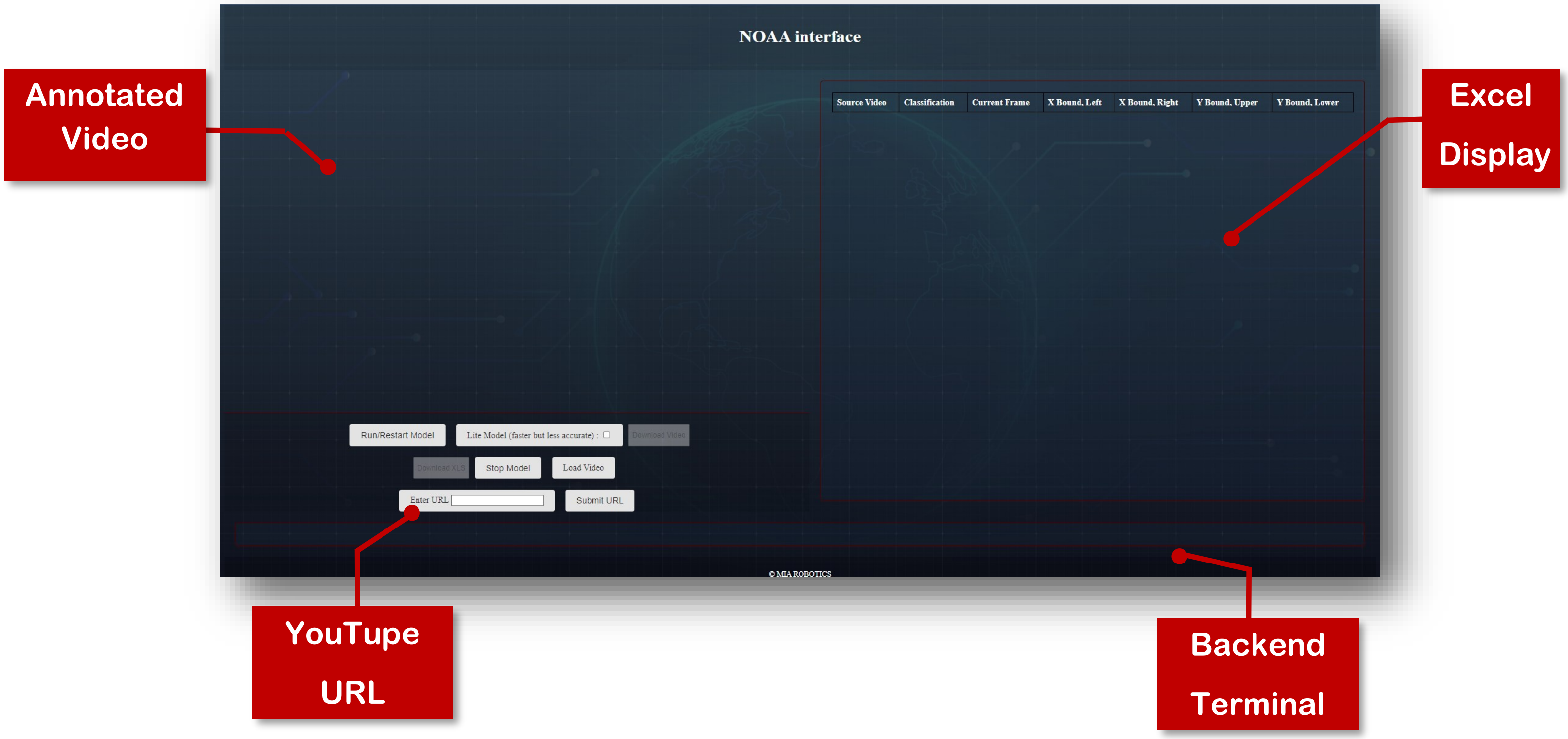
--iou 0.99

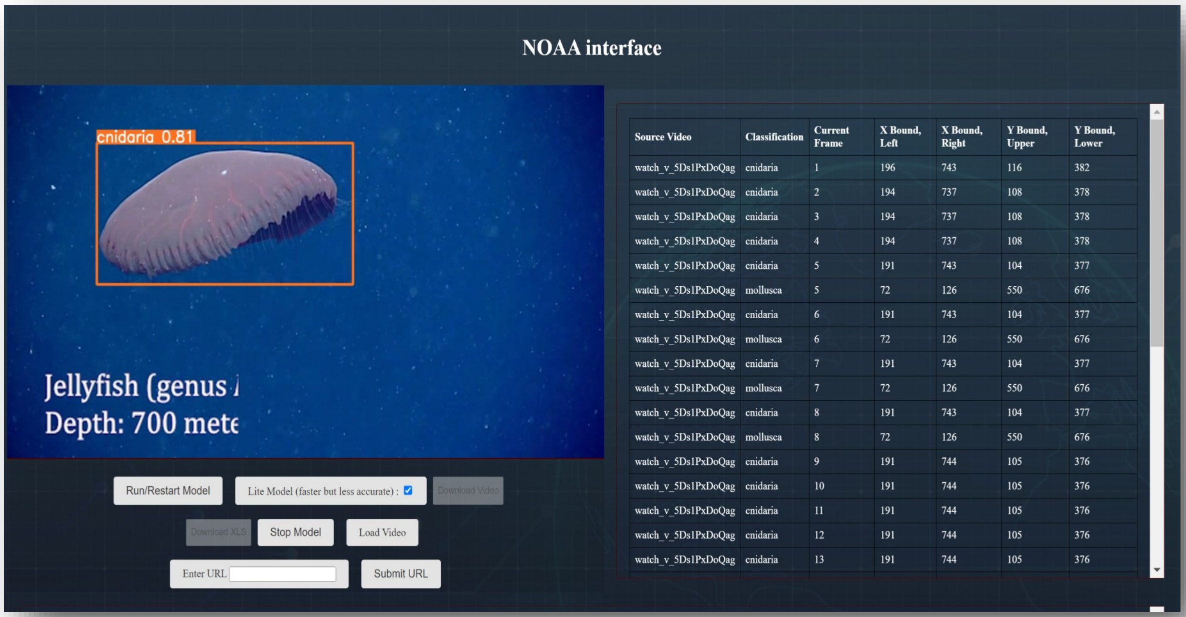


--iou 0.3

2– Web Interface

A Web Based interface is provided that handles deploying the ML model and shows the live output, with the ability to download the results. It works as a simple GUI for the CLI, in fact it spawns a CLI process for each client connecting to it. An online demo is available for testing (notice it doesn't provide high performance GPU, it is just for illustration).





Source Video	Classification	Current Frame	X Bound, Left	X Bound, Right	Y Bound, Upper	Y Bound, Lower
watch_v_5Ds1PxDoQag	cnidaria	1	196	743	116	382
watch_v_5Ds1PxDoQag	cnidaria	2	194	737	108	378
watch_v_5Ds1PxDoQag	cnidaria	3	194	737	108	378
watch_v_5Ds1PxDoQag	cnidaria	4	194	737	108	378
watch_v_5Ds1PxDoQag	cnidaria	5	191	743	104	377
watch_v_5Ds1PxDoQag	mollusca	5	72	126	550	676
watch_v_5Ds1PxDoQag	cnidaria	6	191	743	104	377
watch_v_5Ds1PxDoQag	mollusca	6	72	126	550	676
watch_v_5Ds1PxDoQag	cnidaria	7	191	743	104	377
watch_v_5Ds1PxDoQag	mollusca	7	72	126	550	676
watch_v_5Ds1PxDoQag	cnidaria	8	191	743	104	377
watch_v_5Ds1PxDoQag	mollusca	8	72	126	550	676
watch_v_5Ds1PxDoQag	cnidaria	9	191	744	105	376
watch_v_5Ds1PxDoQag	cnidaria	10	191	744	105	376
watch_v_5Ds1PxDoQag	cnidaria	11	191	744	105	376
watch_v_5Ds1PxDoQag	cnidaria	12	191	744	105	376
watch_v_5Ds1PxDoQag	cnidaria	13	191	744	105	376