A Comparative study between YOLO and DETR algorithm for the Detection of coral-eating crown-of-thorns starfish (COTS) in the Great Barrier Reef

1st Darren Chong
Computer Science
University of New South Wales
Sydney, Australia
z5311772@ad.unsw.edu.au

2nd Lee Chung Lai
Computer Engineering
University of New South Wales
Sydney, Australia
z5210146@ad.unsw.edu.au

3rd Taida Lyu
Computer Science
University of New South Wales
Sydney, Australia
z5295834@ad.unsw.edu.au

4th Oric Lyu
Computer Science
University of New South Wales
Sydney, Australia
z5295834@ad.unsw.edu.au

***Abstract*—Great Barrier Reef is the world's largest coral reef and home to 1,500 species of fish, 400 species of corals, 130 species of sharks, rays, and a massive variety of other sea life. However, like many existing ecosystem, this ecosystem needs human intervention to maintain its stunning appearance due to the overpopulation of a starfish, the the coral-eating crown-of-thorns starfish (COTS). In order to play our part in the current coral crisis, our group have decided to look into two main method to distinguish COTS from the seabed, one being (You Only Look Once) YOLO and the other being (End-to-End Object Detection with Transformers) DETR We will then discuss the accuracy of the particular methods in the hopes of promoting automatic detection methods thus control COTS outbreak to an ecologically sustainable level.**

I. Introduction

Australia's Great Barrier Reef, the world's largest coral reef, is under threat in part because of overpopulation of the coral-eating crown-of-thorns starfish (COTS). To handle this incoming danger, research into systems of computer vision with the ability to distinguish COTS from other sea creatures and objects is regarded by the government as a technology of paramount importance to sustain the profitability and even the existence of the Great Barrier Reef.

II. Task Definition

A. Task

Our project aims to find an innovative manner to analyse video images to monitor COTS outbreaks as accurately and efficiently as possible to address the rapidly expanding population of COTS in the Great Barrier Reef. To achieve this, we decided investigate multiple literature on computer vision processes and ultimately settle on two effective automated detection algorithm - (You Only Look Once) YOLO and (End-to-End Object Detection with Transformers) DETR. We will then compare and investigate on their accuracy, through an in-depth analysis of these methods. The YOLO algorithm will utilise the processes summarised in the Paper : "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object." [1] while End-to-End Object Detection with Transformers) DETR will follow the steps outlined in the paper "End-to-End Object Detection with Transformers." [9]

B. Data Augmentation

Due to the imbalances in the dataset, preprocessing have to be done through augmentation to ensure the reliability of the experiment.

III. Literature Review

A. Deep learning in computer vision: A critical review of emerging techniques and application scenarios [10]

This paper was very useful in determining the possible supervised (deep learning) methods in solving the task of locating COTS. This literature begins by identifying eight different deep network architectures in Computer Vision some being AlexNet, VGGNet and others, as well as stating its advantages, disadvantages and complexity. After discussing all these architectures, the paper goes into the two type of recognition. One being Image classification (aims to assign a label to an input) and Object detection(aims to detect the location of a target object), which is exactly what we need for the COTS detection in the video.

The paper then describes the steps in object detection, suggesting that there is two task in the process to detection the location of COTS, one being classifying the probability of the object being there which is a classification task, second being specifying the specific location of the object using bounding boxes which is a positioning task. It then goes into the mainstream methods used by data analyst, that divides into a one stage detectors or a two stage detectors with the two stage algorithm using a region proposal network to propose the region that contains the object then use a classification layer while the one stage doing it all in one fully connected layer.

The first method described by the paper is the one stage detectors - YOLO which frames the object detecion problem as a regression problem instead of classification [10] and praising its fast detection speed while lacking in accuracy. This method is then improved by including batch normilization, dimension clusters and convolutional anchor boxes, then even further through a joint training method, forming YOLO version 2. Version 2 is then added with Darknet-53 which deepens the number of layers and include cross-layer add to ResNet, this papers then goes on into more detail on the development of YOLO.

The second method introduced is SSD but we did not look into this in depth because this method has the same accuracy as YOLOv3 but works three times faster, recommending us to choose YOLO instead of SSD, it then goes into RetinaNet but simply stating it is more accurate than YOLO by roughly 3 mAP.

Lastly, we derived the idea of Two stage detectors: R-CNN series from this paper, these detectors uses ae three module method to apply CNN on object detection. It first generates regions in the input image, the extract features using CNN layers then input these features into linear SVMs to classfiy the object which is called R-CNN, it then improved to Fast R-CNN and Faster R-CNN. This paper went into further into Image Segmentation, Tracking and Restoration, however for the sake of the current task those were not deemed at upmost importance.

B. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [13]

This paper first introduces itself by stating Fast R-CNN have sped up from R-CNN through the inclusion of sharing convolutions across region proposals by generating a convolutional feature map , then wrap them using a RoI pooling later and then fed into a fully connected layer thereby speeding up the computation time, because instead of requiring a large amount of region proposals to from a convolutional NN(Neural network), the convolution operation generates a feature map from the input image using only one operation. The paper then proposes instead of using Selective Search which is a relatively inexpensive search, EdgeBoxes which is ten times faster provides the best trade off between quality and speed. As it is obeservable from the below image, where in Fast R-CNN pipeline, we are observing the image row by row, whereby edgeboxes analyses the whole classification.

C. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors [14]

This paper begins by talking about real-time object detection and its requirements:

- A faster and stronger network architecture
- A more effective feature integration method
- A more accurate detection method
- A more robust loss function
- A more efficient label assignment method
- A more efficient training method

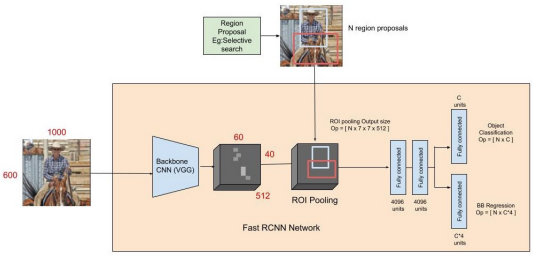


Fig. 1: Fast R-CNN architecture[2]

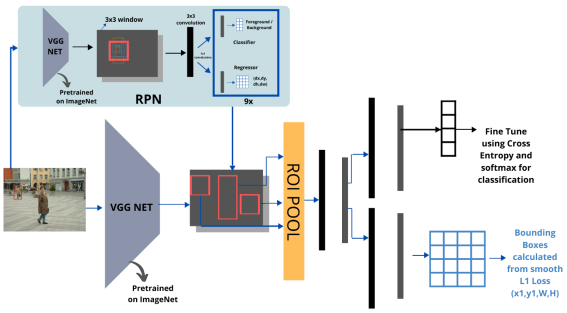


Fig. 2: Faster R-CNN architecture[3]

Then it goes into model re-parameterisation and model scaling which is the key idea that is being conveyed by this paper. Model reparameterisation is used to reduce the trainable BoF into model to speed up the computation time while model scaling is the process of scaling up a base convolutional neural network to endow it with greater computational complexity and consequently more representational power [4]. One of the main improvement for YOLOv7 is E-Elan which stands for Extended Efficient Layer Aggregation Network to increase cardianlity while maintaining the original architecture. This method is the BEST YOLO method having the highest average precision of 66.7%, lower computation power and increased FPS. This can be seen in the graph below.

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Fig. 3: Comparison of YOLO architecture[14]

The comparison of other real-time object detectors demon-

strates that it is the best object detection method in the world.

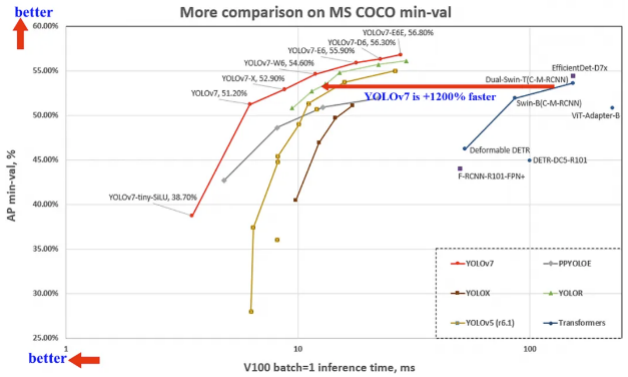


Fig. 4: Comparison of YOLO architecture to other real-time object detectors.[14]

More details will expressed in the section of YOLO since it is one of the method we are utilising to distinguish COTS from the corals.

D. End-to-End Object Detection with Transformers [9]

DETR is a model that utilises a new type of deep learning model also known as transformers. It essentially transfrom an object detection problem as a set prediction problem with an encoder-deconder model based on transformers. With regards to two stage processes that uses prediction boxes with proposals and single stage detectors making prediction with anchors. DETR sped up the detection process by immediately predicting the sets instead of an achor.

This paper then goes into he two core needs for direct set predictions in detection[5]:

- A set prediction loss that forces unique matching between predicted and ground truth boxes
- An architecture that predicts (in a single pass) a set of objects and models their relation

The paper then goes into depth with the architecture of DETR suggesting that it utilises a CNN model as a backbone to learn 2D representation of an image, which then flattens it and inputs it into a positional encoding then the transformer encoder. The transformer decoder than extracts the data through positional embeddings, and pass these output into a shared feed forward network (FFN) that detects the object or the possibility of no object.

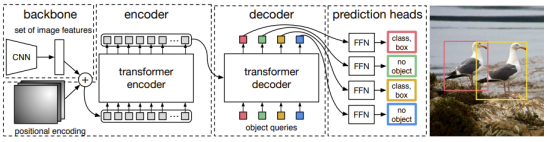


Fig. 5: DETR architecture[9]

However, although DETR is a optimised model, it suffers from its slow convergence as well as the decrease in average precision in small object detection but have large advantages in its large object detection, this can be seen in comparison of DETR with Faster RCNN.

Model	GFLOPS/FPS	#params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-DC5	320/16	166M	39.0	60.5	42.3	21.4	43.5	52.5
Faster RCNN-FPN	180/26	42M	40.2	61.0	43.8	24.2	43.5	52.0
Faster RCNN-R101-FPN	246/20	60M	42.0	62.5	45.9	25.2	45.6	54.6
Faster RCNN-DC5+	320/16	166M	41.1	61.4	44.3	22.9	45.9	55.0
Faster RCNN-FPN+	180/26	42M	42.0	62.1	45.5	26.6	45.4	53.4
Faster RCNN-R101-FPN+	246/20	60M	44.0	63.9	**47.8**	**27.2**	48.1	56.0
DETR	86/28	41M	42.0	62.4	44.2	20.5	45.8	61.1
DETR-DC5	187/12	41M	43.3	63.1	45.9	22.5	47.3	61.1
DETR-R101	152/20	60M	43.5	63.8	46.4	21.9	48.0	61.8
DETR-DC5-R101	253/10	60M	**44.9**	**64.7**	47.7	23.7	**49.5**	**62.3**

Fig. 6: DETR Comparison with Faster R-CNN[9]

E. Deformable DETR: Deformable Transformers for End-to-End Object Detection [15]

This paper goes into deformable DETR which in comparison to the original DETR model, mitigates the slow convergence issues and limited feature spatial resolution by leveraging a new deformable attention module which only attends to a small set of key sampling points around a reference [6]. By utilising deformable convolution which is effective and efficient on image recognition but lacking in relation modeling and evolving that deformable convolution with transformer attention to determine the pixel with the largest amount of attention, thus forming deformable attention which is efficient in implementation and have relevant modeling.

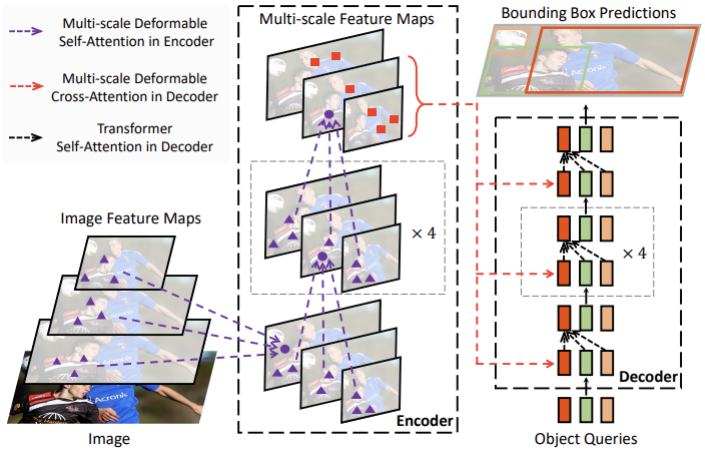


Fig. 7: Proposed Deformable DETR[15]

The proposed deformable DETR model is said to perform better then DETR especially on small object even catching up with Faster R-CNN models with ten times less training epoches. This decrease in computation time is due to chaing DETR a quadratic model to deformable DETR which is a bilinear model thus rapidly speeding up the process, deformable DETR is more suitable for our task due to the need for looking COTS which is a objectively small object in the sea, while lacking the relevant computation power to run DETR

itself. The image below is the comparison between Deformable DETR and DETR.

Method	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	params	FLOPs	Training GPU hours	Inference FPS
Faster R-CNN + FPN	109	42.0	62.1	45.5	26.6	45.4	53.4	42M	180G	380	26
DETR	500	42.0	62.4	44.2	20.5	45.8	61.1	41M	86G	2000	28
DETR-DC5	500	43.3	63.1	45.9	22.5	47.3	61.1	41M	187G	7000	12
DETR-DC5	50	35.3	55.7	36.8	15.2	37.5	53.6	41M	187G	700	12
DETR-DC5+	50	36.2	57.0	37.4	16.3	39.2	53.9	41M	187G	700	12
Deformable DETR	50	43.8	62.6	47.7	26.4	47.1	58.0	40M	173G	325	19
+ iterative bounding box refinement	50	45.4	64.7	49.0	26.8	48.3	61.7	40M	173G	325	19
++ two-stage Deformable DETR	50	46.2	65.2	50.0	28.8	49.2	61.7	40M	173G	340	19

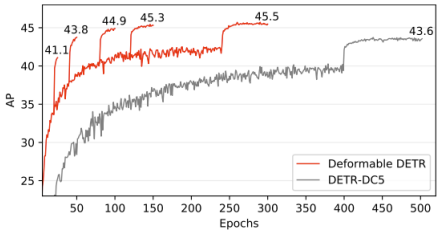
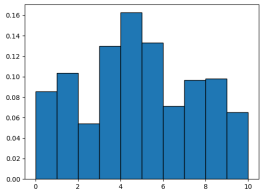


Fig. 8: Deformable DETR compared with DETR[15]

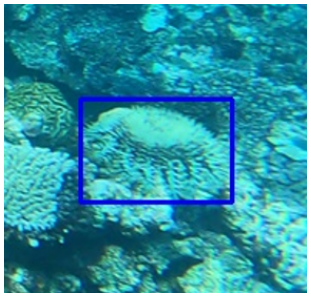
IV. Method

A. LBP

Our initial plan is to use LBP for the task. One of the main characteristics of cots are the spikes, it makes sense to use texture-based feature for the detection task. However, after some investigation, it became obvious that detection based solely on texture is insufficient for our task.

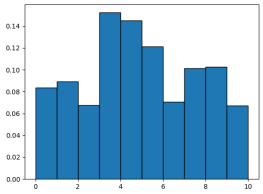


(a) LBP, P = 8, R = 0.5, uniform

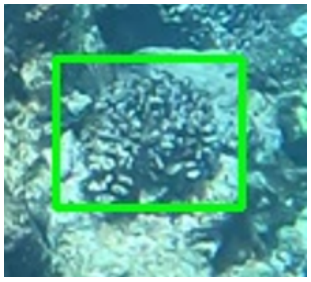


(b) Cot in image 0-1017

Fig. 9: Cots and corresponding LBP histogram



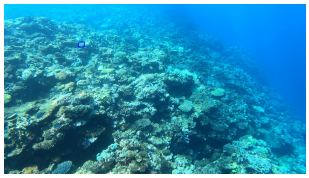
(a) LBP, P = 8, R = 0.5, uniform



(b) Spikey corals in image 0-9345

Fig. 10: Spikey coral and corresponding LBP histogram

(a) Image from prior LBP experiments in [11]



(b) Image from COTs dataset [12]

Fig. 11: Comparison of dataset

A lot of corals have similar textures compared to cots. Figure 9 and Figure 10 shows a cot and a spikey coral with their corresponding LBP histogram. Their LBP histograms are very similar and it will be difficult to discriminate between them. Previous studies had shown the effect of spikey corals on cots detection [11]. The results in [11] have significant false positives due to spikey corals. As images in our cots dataset [12] set were shot at a high altitude, this effect could only worsen considering the high coral density in our dataset compared to the dataset used in [11] as shown in Figure 11.

This provides evidence that detection based solely on a single feature is far too simplistic and unreliable for our task as the vast types of corals have different overlapping features with cots like shape and color, there is no single discriminating feature and multiple features must be taken into consideration, which this leads us to CNN-based approaches such as YOLO and deformable-DETR.

B. Pre-Processing the Dataset

1) EDA: This dataset [12] is a collection of underwater images taken at various time and locations around the Great Barrier Reef where a single image my contain zero or more starfish. The COTS dataset contains roughly thirteen thousand images of size 1280 x 720, with bounding boxes for all images listed in the form of .csv file with the following fields:

- video_id - ID number of the video the image was part of.
- video_frame - The frame number of the image within the video.
- sequence - ID of a gap-free subset of a given video.
- 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 the corresponding image_id in format of JSON array. Each bounding box is a JSON object and has 4 fields:
  - x - The top left x-coordinate of the bounding box
  - y - The top left y-coordinate of the bounding box
  - width - The width of the bounding box
  - height - The height of the bounding box

There are 23,501 images in training sets in total.

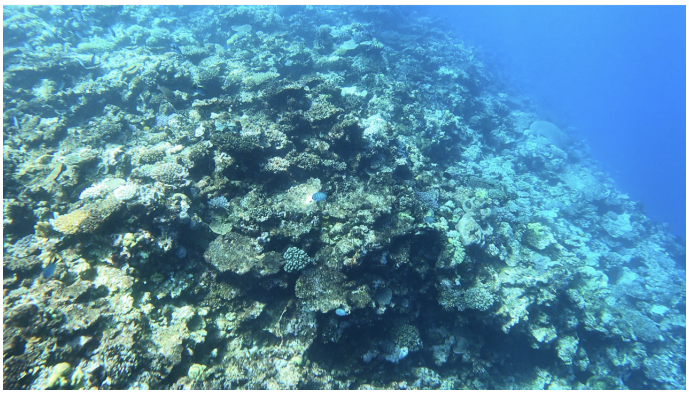


Fig. 12: Example of Image from COTS Dataset[12]

- 0 corrupted data.
- 0 duplicate data.
- 6,708 data from video 0.
  - 2,143 labelled
- 8,232 data from video 1.
  - 2,099 labelled.
- 8,561 data from video 2.
  - 677 labelled.
- 4,919 labelled data in total.

We can also count the pixels of the three channels in fig 13, each channel has a total of 21,658,521,600 pixels. And we can see for the whole dataset, the red channel has most 0 value, and it's not evenly distributed. This provides some ideas for our image enhancement.

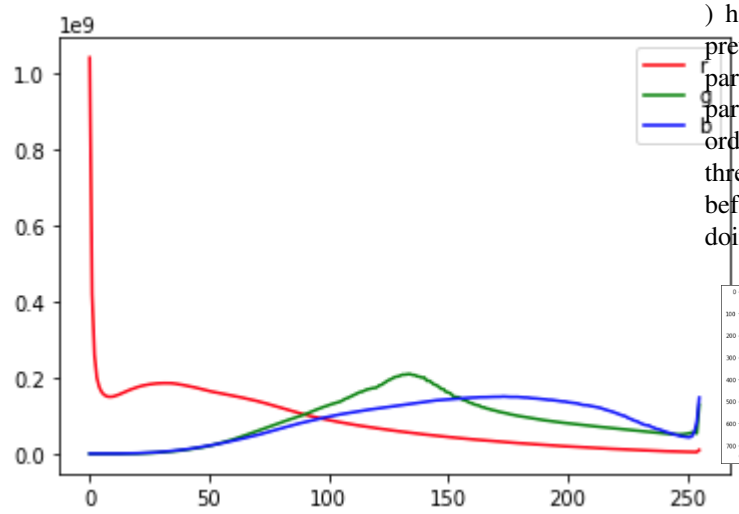


Fig. 13: pixel count

2) Augmentation: In the COTS dataset, the data are extremely imbalanced, with the number of images without objects being 79% of the total dataset. Therefore, data augmentation has to be done on the dataset to minimize training error. First, the dataset is split into two subsets, data that contain the COTS and data that have no objects in them. Then, an 80:20 split ratio is used to acquire a train set and a validation set for both subsets. The data containing the COTS in the train set are afterwards augmented to 4 times of the original size to roughly match the size of the data with no objects. The augmentation methods include flipping the images horizontally, vertically and rotating the images by 180 degrees. The two subsets are then combined and be used in the two models, namely YOLO and Deformable-DETR.

3) Image Enhance : We all know that underwater images are covered with a blue mask because of the water surrounding the equipment. So objects photographed underwater are obscured by blue and hard to see, or their colour is very different from their true colour.

So we came up with the idea of using square equalization to enhance the contrast in the red and blue channel. And we also do HE for all channels to compare. If EH is applied to all channels, it sometimes performs well, but when the contrast of all three channels is not high enough, it will cause excessive enhancement. In general, HE can enhance the contrast to a certain extent, but it will also have a great impact on the image in some cases.

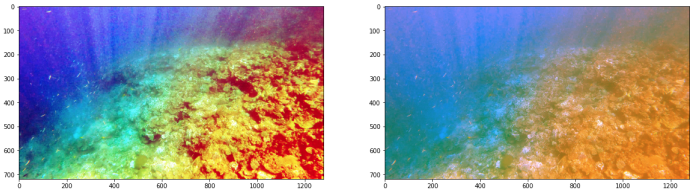


Fig. 14: HE for RGB VS B&R

CLAHE(Contrast Limited Adaptive Histogram Equalization) has long been used to enhance underwater images, and it prevents the image from being over-enhanced. Firstly, compared with conventional HE, CLAHE is locally enhanced; this part is also called AHE(Adaptive Histogram Equalization). In order to prevent the image from being excessively enhanced, a threshold is set, and the part beyond the threshold is cropped before AHE is performed. And what we like to see is that it's doing very well.

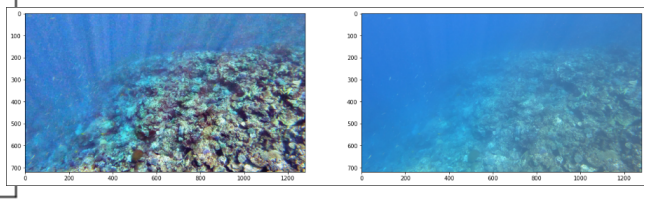


Fig. 15: CLAHE vs Origin

Usually, underwater images are dark, so using gamma correction to adjust them can make them look more comfortable. So finally, we added gamma correction to complete our image enhancement.

C. YOLO- You Only Look Once

1) Preprocessing: Folders for the training set and validation set are created, with each folder containing a folder for labels

and images. The structure is as follows.

- train
  - images
  - labels
- val
  - images
  - labels

The images in "images" folder are in format "image_id.jpg". Files in "labels" folder are .txt files in format "image_id.txt" containing annotations on the location and size of ground truth bounding boxes in "image_id.jpg". For each object in an image, a new line in the corresponding .txt file in the format of

class-label relative-x-center relative-y-center relative-width relative-height

is created. As we are concerned with only one object, class label is 0, and all other variables can be easily obtained from the annotation field of the original COTs .csv file with the formulas below:

$$\text{class-label} = 0$$
$$\text{relative-x-center} = \frac{x + 0.5\text{width}}{1280}$$
$$\text{relative-y-center} = \frac{y + 0.5\text{height}}{720}$$
$$\text{relative-width} = \frac{\text{width}}{1280}$$
$$\text{relative-height} = \frac{\text{height}}{720}$$

Label text file corresponding to image with no object is an empty .txt file. In the case of image enhanced dataset, all images in the "images" folder are augmented and enhanced images.

2) Steps: YOLOv7 is chosen because it is the fastest and most accurate real time object detection model, and have one of the best Intersection over Union in bounding boxes. Intersection over Union is a metric used to measure the localisation accuracy and error in object detection model. The IOU is derived from the area of intersection of predicted bounding boxes and ground truth bounding boxes divided by the area covered by both bounding boxes.

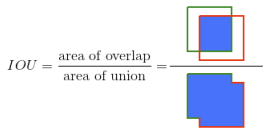


Fig. 16: Equation of Intersection over Union[7]

The YOLO algorithm begins by spliting the image into N grids, each having equal square region, where each of these N grids will be individually used for object detection of the object. These girds will predict bounding box coordinates along with the object label using classification and the probability of the object being in that bounding box. This process will greatly lower the computation but cause lots of duplication, hence YOLO will utilise Non Maximal Suppression to solve this problem.

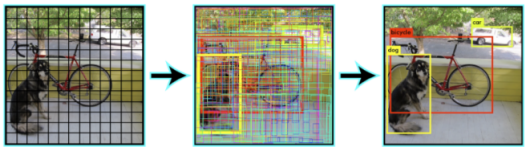


Fig. 17: Non Maximal Suppression Process[7]

What happens in the non maximal suppression is that YOLO get rids of all the bounding boxes with lower probability scores, so it essentially suppresses the bounding boxes with the largest IOU with the highest probability, repeating this step till the final bounding boxes are obtained.

The image below is the structure of the YOLOv7 model:

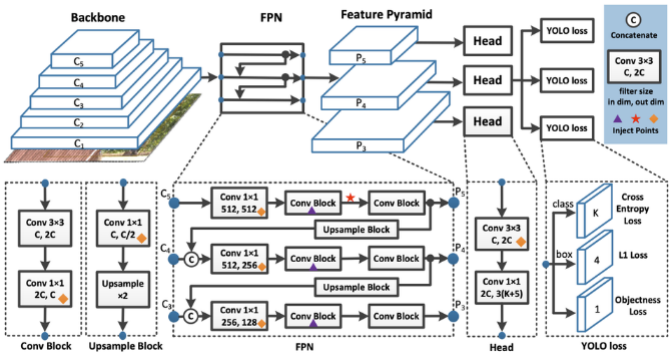


Fig. 18: YOLOv7 network architecture[8]

YOLOv7 architecture is derived and improved from YOLOv4 , this model uses E-ELAN(Extended Efficient Layer Aggregation Network). This architecture enables the framework to expand, shuffle and merge cardinality to enhance the learning ability of the network by increasing the number of convolutional layers. This E-ELAN is the backbone of the YOLO detector which pools image pixels to form features, in our case, this backbone is pre trained on a classification dataset, MsCOCO. By using E-ELAN we are able to enhance the features learned by different feature map of our dataset of COTS.

Another improvement that YOLOv7 is a compound model scaling which considers the following parameters :

- Resolution (size of the input image)
- Width (number of channels)
- Depth (number of layers)
- Stage (number of feature pyramids)

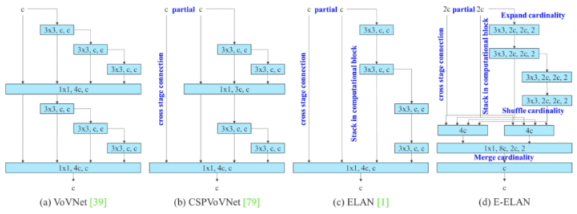


Fig. 19: E-ELAN architecture[8]

And YOLOv7 have decided to use a compound model scaling approach where the width and depth are scaled for concatenation based models where the scaling factors are all independent

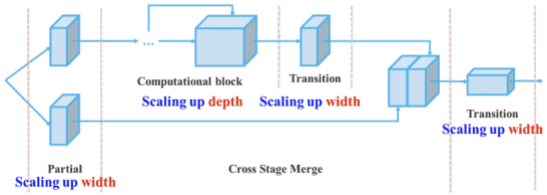


Fig. 20: Compound Scaling[8]

Finally to improve the performance of our model, we plan to introduce the following BoF methods following the paper, reparametrisation increases training time but improves results,in this case we utilise module level reparameterisation where the model training process is split into multiple modules where the outputs are ensembled to obtain the final model.

3) Results: To reduce training time, pretrained weights of YOLO for MS COCO detection task is chosen. We used hyper-parameters used in MS COCO task for baseline performance but had disabled probabilistic image flipping as our augmented dataset already contains flipped images. Figure 21 shows the box loss and objectness loss for YOLOv7 model over 51 epochs.

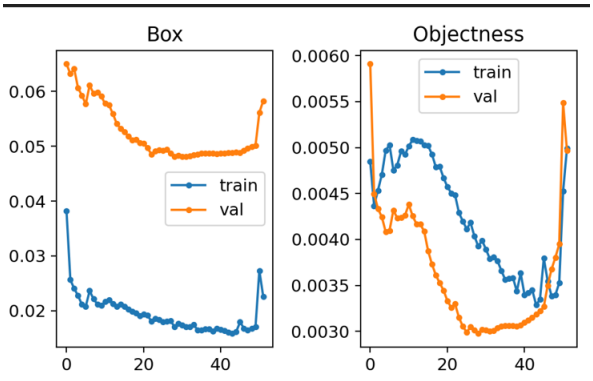


Fig. 21: Box loss and objectness loss for YOLOv7 over 51 epochs

nms-iou/conf-thres	0.01	0.05	0.1
0.65	0.306	0.591	0.559
0.6	0.315	0.592	0.559
0.55	0.321	0.592	0.558
0.5	0.325	0.591	0.558
0.4	0.330	0.591	0.557
0.3	0.333	0.591	0.557
0.1	0.334	0.585	0.551

Fig. 22: f2-score of original dataset for different confidence score threshold (column) and non-maximal suppression IOU threshold (row), *corrected to* 3 *sig fig*

As shown in Figure 21, model starts to converge at around epoch 30, but after epoch 42 both training loss and validation loss increases. Also, validation objectness loss is lower than train objectness loss. The phenomena is normal and is due to regularization term being applied on training loss but not validation loss, inflating training loss. More will be discussed in discussion section.

In testing stage, confidence score threshold and non-maximal suppression IOU threshold are applied at different values. Figure 22 shows the F2 score proposed in [12] of our model. The best F2 score of 0.592 is obtained for 0.05 confidence-score threshold and 0.55 NMS-IOU threshold.

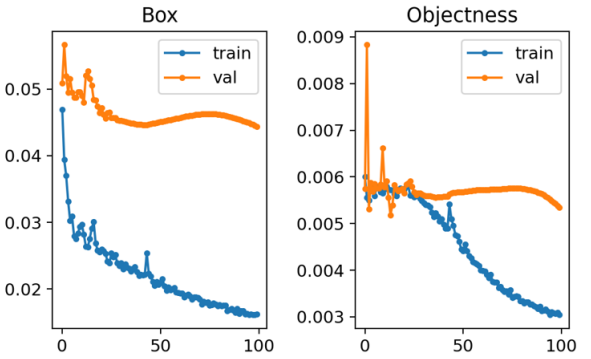


Fig. 23: Box loss and objectness loss for YOLOv7 over 51 epochs, enhanced dataset

For enhanced images, YOLOv7 converges at around epoch 30 and afterward show signs of overfitting as shown in Figure 23. In Figure 24, applying NMS IOU threshold 0.65 and confidence score threshold 0.005 during testing give us the best F2 score of 0.47. It is slightly worse than the original dataset. More will be in the discussion.

D. Deformable DETR (Deformable Transformers for End-to-End Object Detection)

1) Pre-Processing the Dataset: The model treats the input data as COCO dataset format, therefore the data have to be converted beforehand. We have also tried two different dataset to try to improve the performance of the model, with one

nms-iou/conf-thres	0.001	0.005	0.01
0.8	0.196	0.454	0.466
0.75	0.217	0.467	0.466
0.7	0.230	0.469	0.462
0.65	0.237	0.470	0.461
0.6	0.242	0.470	0.460
0.55	0.244	0.467	0.460

Fig. 24: f2-score of enhanced dataset for different confidence score threshold (column) and non-maximal suppression IOU threshold (row), *corrected to* 3 *sig fig*

dataset using the image enhance method and another one without.

2) How does it work?: For the second method, Deformable-DETR is chosen over DETR mainly because it has a faster training time. Deformable-DETR is a variant of DETR, which is a model that uses the attention mechanism to encode features with attention information. Both models also have an end-to-end pipeline, meaning there is no need to employ non-maximum Suppression (NMS) to optimize bounding boxes, reducing the number of hyper-parameters. Deformable-DETR is also better at detecting small objects, and considering the COTS are generally small in sizes, it is preferable to use Deformable-DETR over DETR.

For the backbone of Deformable-DETR, a pretrained residual neural network of 50 layers (ResNet-50) is chosen, so that the model can focus more on training the transformer part, reducing the time for convergence. The input images first go through the backbone and have their features extracted. The features are then passed into the transformer encoder, encoding the features with position and attention information. After that, the encoded features are passed to the decoder, and the object queries, which are trainable weights, will be used to reference and acquire information from the encoded features. The output are then passed through a feed forward layer to obtain the predicted class, bounding box coordinates, and the confidence score of the object detected.

3) Results: The results show that the maximum F2 score of the model is 0.964 at a confidence score threshold of 0.2. The maximum F2 score of the model is worse if the data is pre-processed using image enhancement. The training and validation loss of both using and not using the image enhancement pre-processing method are roughly the same, with the training loss of the image enhancement method converging slightly earlier.

V. Discussion

A. YOLO & DETR

We think there are two main factors that resulted in poor YOLOv7 performance.

For our original dataset, box and objectness loss for both training and validation sets increase after epoch 42 as shown in Figure 21. This is likely due to learning rate and momentum

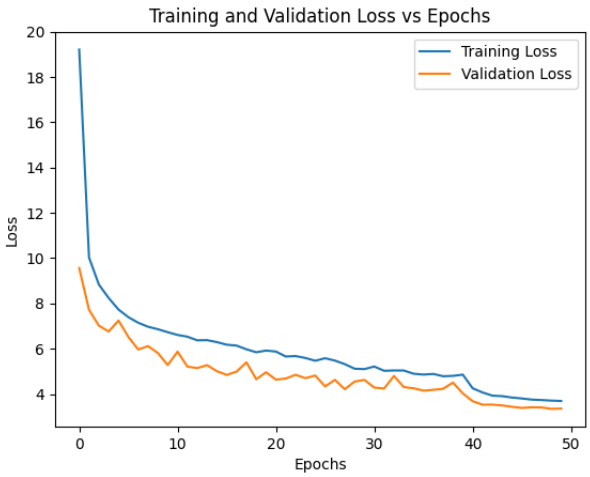


Fig. 25: Deformable-DETR training and validation loss over 50 epochs

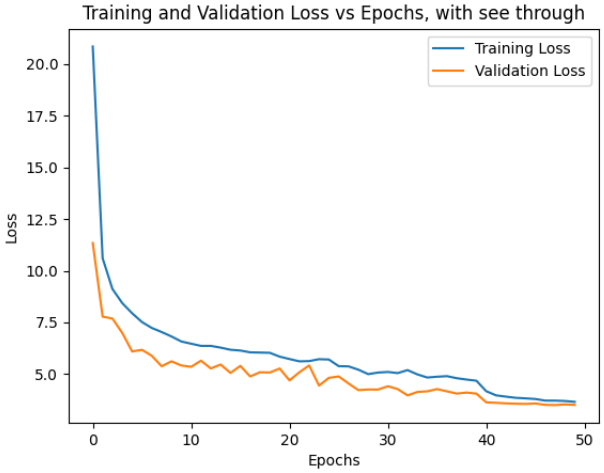


Fig. 26: Deformable-DETR training and validation loss over 50 epochs, data pre-processed with see through

for MS COCO task being too high for our task. This resulted in overshooting when the loss function of the model is near minimum and the model did not attain it's minimum. Varying learning rate and momentum can prevent overshooting.

Another factor we think that lead to poor performance of YOLOv7 is how default parameters of YOLOv7 used in the MS COCO task resizes the image to 640 x 640, lowering the input resolution. A large portion of false positives results from the misclassification of gray blobs as cots. An example is shown in Figure 29. It classifies objects as cots even it does not have the spikey characteristic that cots have. Downsampling the dataset removes fine-grained features, and for small cots, the spikey characteristic is loss. Figure 30 provides more evidence for this claim. It shows large false positive even for low IOU threshold (TP FP FN calculated according to [12], IOU threshold = 0.3). Although Deformabe-DETR also applies resizing, it randomly picks a scale from [480, 512, 544, 576,

conf thres	F2 score
0.005	0.223
0.01	0.516
0.05	0.936
0.1	0.959
0.2	0.964
0.3	0.962
0.4	0.960
0.5	0.957
0.6	0.951

Fig. 27: F2 score at various confidence score threshold with original data, *3 sig fig*

conf thres	F2 score
0.005	0.033
0.01	0.572
0.05	0.939
0.1	0.953
0.2	0.951
0.3	0.946
0.4	0.936
0.5	0.928
0.6	0.920

Fig. 28: F2 score at various confidence score threshold with enhanced image, *3 sig fig*

608, 640, 672, 704, 736, 768, 800] for each image so the effect of resizing is not detrimental. This can be prevented by using a resolution higher than 640 x 640.

To further improve the model, we could further augment our data applying different types of transformation such as perspective transformation to deal with data imbalance instead of simple flipping and rotation. We could also scale up the width (number of channels) and depth (number of layers) of our YOLO model to capture more fine-grained details.

Despite the poor performance, the result is not fully reflected on the F2 score as the F2 score proposed in [12] penalizes FN more and has more tolerance to FP. Nevertheless, there is much room for improvement.

Unlike our attempt on the task using YOLOv7, the results of Deformable-DETR is shockingly accurate, and we believe one of the reason behind its high F2 score is due to the frames similarity. During the pre-processing stage, the data are split using a train-test-split ratio of 80:20 randomly. By doing so, frames which are extremely similar to each other as they are only 1 frame apart are randomly assigned to the train and validation set. Therefore, validation accuracy can be high due to the similarity of images between the train and validation set.

There is also one shocking discovery from the results generated, the model is detecting the COTS earlier than the ground truth. As shown in Figure 31, there is a COTS inside the blue bounding box, but it is ignored and classified by the

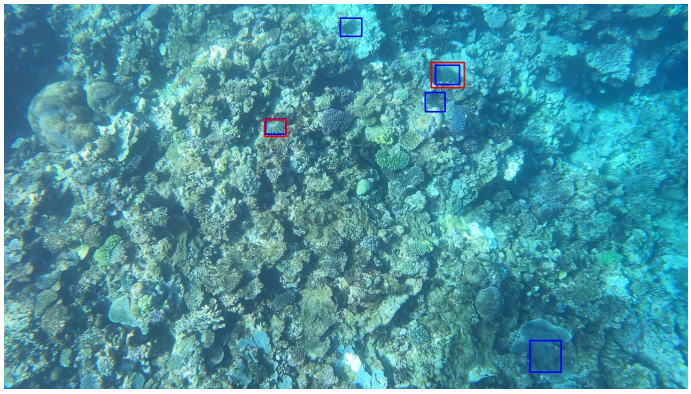


Fig. 29: False positive result in blue bounding box from YOLOv7, Red box = Ground Truth, Blue box = Predicted

TP	FP	FN
1544	2157	527

Fig. 30: TP, FP, FN for IOU threshold = 0.3, calculation according to [12]. YOLO score threshold = 0.05, NMS-IOU threshold = 0.55

ground truth frames later in Figure 32. This means that the performance of the model can be even higher as there are less false positive due to misclassification of the labels.

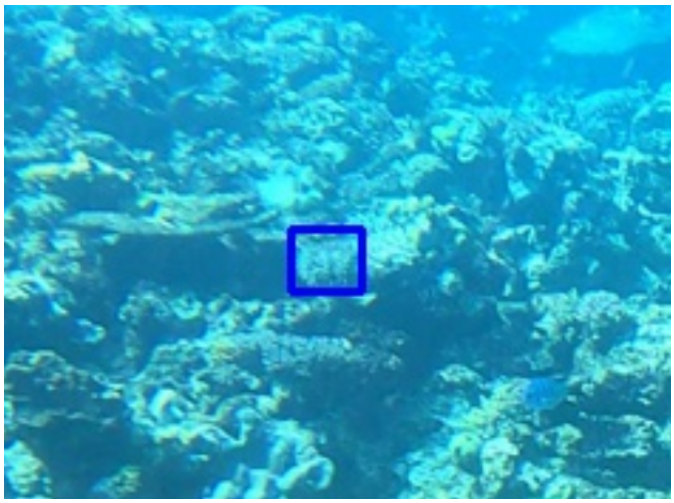


Fig. 31: Predicted Bounding Box on Image with Video-ID 0, Frame-ID 324

Compare the two models from the F2 score, we can see DETR doing much better than YOLO, this is because the YOLO is not good at detecting very small targets and YOLO has the advantage of processing fast, according to the results analysis of YOLO we know that, YOLO did much FP targets. For DETR we can know it get an extraordinary score, according to the competition leaderboard we can know the score is unusual, we can also get the reasons in the DETR result analysis, there are continuous frames in the video, and these frames are very similar, and these similar images are split into

Fig. 32: Predicted and Ground Truth Bounding Box on Image with Video-ID 0, Frame-ID 330

train set and test set. If we could have a separate video to test it we would get a normal score. Hence YOLO get a low score in the same situation proof that YOLO is not good at dealing with small targets.

B. Origin & Enhanced Image

Image enhancement does not help the model distinguish the target better, although they have similar scores in the DETR model, in practice image enhancement is not very helpful for deep networks. This is because the neural network obtains image features through deep convolution, rather than the traditional feature extraction method. So some means of augmenting data can be used for pre-processing, and I think it is more helpful for convolutional networks to find the std and mean values of three-channel and then standardize them rather than image enhancement. Image Enhancement can only help humans observe information in images more easily, so in the final output programs, we should obtain bounding boxes with original images and then present them to human users through Image Enhancement processing, so as to achieve ideal effects.

REFERENCES

[1] URL: https://arxiv.org/pdf/2207.02696.pdf.

[2] URL: https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022.

[3] URL: https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97.

[4] URL: https://medium.com/syncedreview/model-scaling-thats-both-accurate-and-fast-facebook-ai-proposes-novel-scaling-analysis-framework-249aec9bdfac.

[5] URL: https://medium.com/analytics-vidhya/end-to-end-object-detection-with-transformers-detr-by-facebook-ai-833f4086280a#:~:text=The%20DETR%20Model&text=Two%2Dstage%20detectors(R%2D,these%20initial%20guesses%20are%20set..

[6] URL: https://huggingface.co/docs/transformers/model_doc/deformable_detr.

[7] URL: https://www.v7labs.com/blog/yolo-object-detection#h3.

[8] URL: https://blog.roboflow.com/yolov7-breakdown/.

[9] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. DOI: 10.48550/ARXIV.2005.12872. URL: https://arxiv.org/abs/2005.12872.

[10] Junyi Chai et al. "Deep learning in computer vision: A critical review of emerging techniques and application scenarios". In: *Machine Learning with Applications* 6 (2021), p. 100134. ISSN: 2666-8270. DOI: https://doi.org/10.1016/j.mlwa.2021.100134. URL: https://www.sciencedirect.com/science/article/pii/S2666827021000670.

[11] Ryan Clement, Matthew Dunbabin, and Gordon Wyeth. "Toward Robust Image Detection of Crown-of-Thorns Starfish f or Autonomous Population Monitoring". In: *Proceedings of the 2005 Australasian Conference on Robotics and Automation, ACRA 2005* (Mar. 2012).

[12] Jiajun Liu et al. *The CSIRO Crown-of-Thorn Starfish Detection Dataset*. 2021. arXiv: 2111.14311 [cs.CV].

[13] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. DOI: 10.48550/ARXIV.1506.01497. URL: https://arxiv.org/abs/1506.01497.

[14] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. DOI: 10.48550/ARXIV.2207.02696. URL: https://arxiv.org/abs/2207.02696.

[15] Xizhou Zhu et al. *Deformable DETR: Deformable Transformers for End-to-End Object Detection*. 2020. DOI: 10.48550/ARXIV.2010.04159. URL: https://arxiv.org/abs/2010.04159.