You Only Edit Once

CS604 Computer Vision Project

Final Report

YOEO Web App Link

GitHub Link

Group 3

He Xinyi (xinyi.he.2020@mitb.smu.edu.sg)

Lau Te Yang (teyang.lau.2021@mitb.smu.edu.sg)

Yan Licheng (licheng.yan.2021@mitb.smu.edu.sg)

Li Zihao (zihao.li.2021@mitb.smu.edu.sg)

Lim Hsien Yong (hy.lim.2021@mitb.smu.edu.sg)

6th November 2022

1. Introduction

With the growth of diving tourism, recreational diving has become a widely enjoyed with many recreational divers (such as ourselves) choose to record their dives using small, sport-grade video equipment such as Go-Pro cameras.

Recreational divers relish the thought of seeing the myriad of colorful ocean flora and fauna in every single frame of their recorded video. However, a large proportion of these recorded videos are void of ocean life[1] and/or have terrible lighting conditions[2] (see Annex A). The large expectation-reality gap means that recreational divers often spend a lot of time editing recorded videos to create a satisfactory video piece.

Inspired by our Computer Vision class on object detection, we have developed a solution (aptly) titled 'You Only Edit Once'. Our solution aims to leverage on the advancement of deep neural networks in computer vision[3] to place the burden of video editing solely into the hands of AI, thereby reducing manual labor hours. Our solution helps users to:
(1) Automatically trim their raw videos to retain only the interesting portions;
(2) Add a soundtrack with the trimmed video to form a cinematic reel; and
(3) Develop beautiful wallpaper images from the video they recorded.

In our opinion, frames of interest within a video are frames teeming with marine life. This makes object detection models the basis of our solution, as these models capture this information.

2. Data collection and processing

This step is essential for finetuning of pre-trained CV models for our use case. To ensure that the model is trained on realistic data[4], we curated our dataset using our own collection of diving videos, as well as diving videos of recreational divers from public domains such as YouTube. In total our dataset includes 477 images from 76 different video sources.

a) **Frame extraction**. As videos typically contain multiple frames per second, the number of images from 1 minute of footage alone would already exceed the total number of images in our curated dataset. However, 'putting all our eggs in one basket' prevents the model from generalizing. Hence, we utilized a frame sampling technique to randomly select 1 frame per 'n' seconds[5] so that we can harvest images from different video sources.

b) **Image annotation and augmentation.** Sampled images were annotated using Datature's image annotation tool. We annotated each image for the classes 'shark', 'coral', 'fish', 'turtle' and 'manta ray', and drew the bounding boxes as tightly as possible as the quality of image annotations are paramount for model training. Only full-bodied samples with minimal overlap were labelled[6], so that the model is trained to detect marine creatures of interest with minimal occlusion. Figure 1 shows an example of an image annotation. To ensure consistent and good annotations, we performed quality checks on all annotations before model development.
To enhance the generalizability of the model, we used Datature's autoML platform to apply image augmentation (see Annex B) as part of the training pipeline. 80% of our annotated images were used for training, with the remaining used for validation.

[1] It's always a mystery to us why so many fishes and coral dance in concert when aired on National Geographic documentaries but disappear when we take the dive to see them in-person.
[2] The ocean is typically livelier at considerable depths, which makes video recordings dark and blurred.
[3] Namely using YOLO (You Only Look Once) based models.
[4] That is the raw footage of recreational dive videos instead of beautiful marine documentary footage aired on the National Geographic channel.
[5] The selection of 'n' depends on the video's duration.
[6] i.e. we do not label coral 'forests', fishes that are not full-bodied (covered by other fishes), etc.

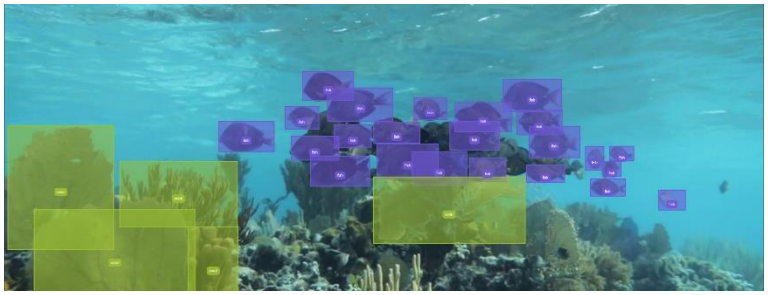


Figure 1: Annotated image with fish (purple) and coral (yellow)

3. Model development

a) Datature autoML platform. Low latency inferencing is necessary to achieve fast video trimming, enhancing user experience. Thus, we used single stage detector (SSD) models, which skip the region proposal step to run object detection over dense location sampling. Of the SSD models in Datature's extensive model repository, our team opted to use YOLOX architectures as it offers the best balance between number of trainable parameters and latency. Broadly, the YOLOX architecture adopts an anchor-free[7] and decoupled head approach, greatly improving inference and convergence speeds while maintaining good accuracy[8]. A summary of the YOLOX architectures used and our setup of the best performing finetuned YOLOX model are as follows:

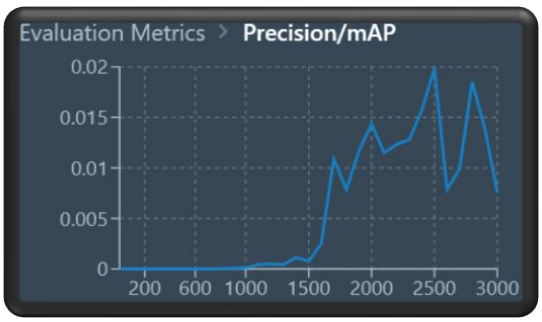
Model	Number of trainable parameters (million)	Latency (ms)
YOLOX-Small	9.0	9.8
YOLOX-Medium	25.3	12.3
YOLOX-Large	54.2	14.5

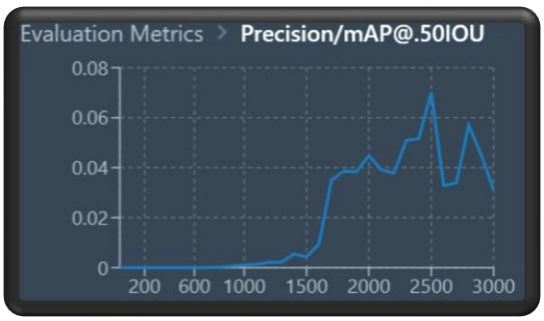
Table 1: Summary of YOLO-X architectures

Model	Train-Test Split	Batch Size	Training Steps	Detections per class	Checkpoint Strategy
YOLOX - Medium	0.2	8	3000	8	Best mAP

Table 2: Setup of our best performing finetuned YOLOX model result on Datature

We selected a batch size of 8 as larger values caused an out-of-memory error on the platform. The best overall mAP@0.5:0.9 was eventually attained at 2,500 training steps (Figure 2).





[7] This removes the need for cluster analysis to select the best parameter for anchor boxes

[8] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). YOLOX: Exceeding YOLO Series in 2021. https://doi.org/10.48550/arXiv.2107.08430

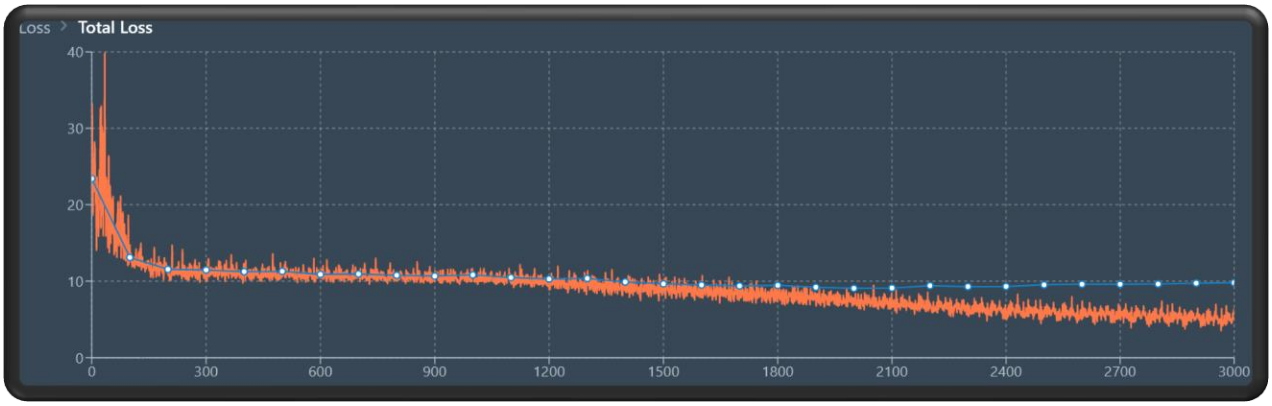


Figure 2: Graphical summary of finetuning YOLOX-Medium on Datature

We experimented with all 3 YOLOX model architectures (small-medium-large) on Datature. In summary, the performance improvement was marginal between YOLOX-large and YOLOX-medium, but the inference timing was far longer for YOLOX-large. The increase in inference time between YOLOX-medium and YOLOX-small was marginal but the increase in accuracy was not. Therefore, the YOLOX-medium was the optimal model. During inference however, we noticed several salient issues summarized in table 3 (see Annex C for a more thorough elaboration):

Issue	Description
Weak detection	Not all full bodied & visible fishes are detected
Loose bounding box	The predicted bounding boxes are very loose around the subject
Misclassification	Corals are often misclassified as fishes

Table 3: Issues when running inferences

In our opinion, these issues were 'showstoppers', and we were concerned that the final truncated videos would be unsatisfactory. However, further experiments on other models in Datature did not yield better mAP scores. We therefore finetuned YOLO-X models beyond Datature's repository.

b) Other YOLOX models explored. We explored finetuning the YOLOv5n and YOLOX-Nano models in colab and immediately observed an improvement in mAP scores. We suspect that the larger YOLOX models either overfitted to our training data[9], or that the batch size for training was insufficient for the model to reach the desired performance. We explored the following YOLOX models beyond Datature:

Model	Parameters (M)	mAP (0.5:0.95)
YOLOv5n	1.9	0.285
YOLOX-Nano	0.91	0.258

Table 4: Other YOLOX models explored

Train-Test	Batch Size	Epoch	Checkpoint Strategy
0.2	32	300	Best mAP

Table 5: Training setup of YOLOX models in Colab

Aside from the aforementioned data augmentation methods, the YOLO model series includes the Mosaic and MixUp augmentations[10,11] during training by default. However, we found that these augmentations were unsuitable if the augmentation was too extensive. We removed the MixUp

[9] See Annex C for more details.

[10] Mixup creates a superposition of multiple images, while mosaic performs a simple stitching of images. See Annex B for an illustration of these augmentations.

[11] From the YOLOX GitHub page: https://github.com/Megvii-BaseDetection/YOLOX. From the YOLOV5n GitHub page: https://github.com/ultralytics/yolov5

augmentation entirely for our YOLOX-Nano model due to poor training performance (see Annex D for our full training parameters). Due to our bias towards fast inference speeds, we deployed the YOLOX-Nano architecture in our application as the compromise in model performance is minimal.

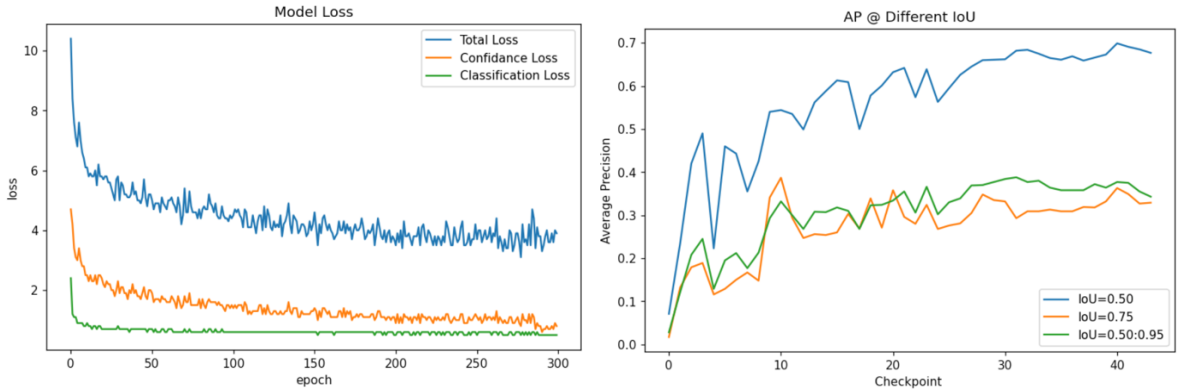


Figure 3: Loss vs avg precision plot for YOLOX-Nano.

When running inferences, we noticed improvements compared to the models trained on Datature:

1) **Accurate prediction:** the model would be able to recognize most of the fish in a frame and give each prediction a high confidence score.
2) **Tight and stable bounding box:** the model would be able to recognize full body fish in a frame giving a stable bounding box.

4. Post-processing algorithms

a) Video truncation: The finetuned YOLOX-Nano model was used to make inference on every frame of the input video. The frames were then scored using two custom metrics (see Annex E):

1. **Proportion of the total area of objects within the frame:** We assume here that scenic frames are made up of large areas occupied by objects of interest[12].
2. **Weighted count score of objects within the frame:** We also assume that scenic frames contain many objects of interest, and that frames with rare objects are more desirable. Therefore, we conferred different scores to different classes with the classes of larger sea creatures, namely 'turtle', 'shark' and 'manta ray', given higher weights.

Each frame is given a proportion of area score and weighted object score as described above. The respective scores are normalized before summation to yield the final score. Frames with a final score below the threshold are filtered away, and additional masking is applied to frames that do not contain the user specified objects of interest[13].

To prevent the stitched video from being choppy due to intermittent filtering, a trimming strictness parameter was introduced to allow previous n frames of any accepted frame to be kept. This also prevents the video frame from instantly 'jumping' to an object of interest, and allows the object to enter the frame gradually, creating a more cinematic video.

Additionally, we observed that the duration of inference might still be significant despite using YOLOX-Nano (on CPU), as every single frame in the video is scored by default. This might not be necessary in our opinion. For example, the 24 frames within the same second of a 24 fps[14] video are likely to be homogenous and contain similar information. Thus, it is possible to still capture majority of the information and reduce inferencing time if we were to score frames selectively. Inferring on only 12 frames for an 24 fps video (or once every 2 frames), halves the duration of inferencing. Accordingly,

[12] The more the frame is filled with objects of interest, the more interesting/scenic/beautiful it is. A frame with a fish the size of a dot would not be considered scenic (or interesting to our users).
[13] For instance, if the user specifies to see sharks, we filter away frames with very good scores if they do not contain 'shark'.
[14] 'fps' refers to frames per second. A 24 fps video means that there are 24 frames in 1 second.

we offer this customization as an advanced option for our users. Users can specify the number of frames per second for the model to perform inference.

Our algorithm also adds a soundtrack with the truncated video to form a cinematic reel. As the audio and video files might be of different duration, special care is taken to ensure that there are no silent moments in the video. The algorithm truncates the audio to the length of the video if the audio track is longer, otherwise if only one audio track is specified, the audio track is cloned until it is longer than the video before truncation to match the video's length. Though successful in our development phase, we regret that we were unable to confer the users with flexibility to stitch an audio of choice on our application due to incompatibilities between Streamlit and the youtube-dl library[15]. We stitch all truncated videos with a default audio track instead.

b) Image Beautification. Diving videos can be a great source of wallpaper images, and we wanted to confer our users this option to help them get more from their recorded videos. The main steps of the image extraction algorithm are:

1. **Extract the top n scoring frames**: 'n' is a user input, and frames are extracted based on frame scores[16]. We took special care to ensure that adjacent frames are not selected even if they had the best scores, so that users will not get 'n' very similar images[17].
2. **Beautify frames**: Diving images are typically dark, blurry, and full of blue light. Therefore the 'beautification' of diving images needs to address this.
   - We start by increasing the amount of red light and decreasing the amount of blue light in the image. This is done using OpenCV's lookup table (LUT) function. We specify the specific color channel's gradient change in its respective lookup table and return the sum of all color channels as the filter. When applied to an image, it has the effect of lifting the image's brightness to a less dour and melancholic hue.
   - We measure and adjust the brightness of the image past a threshold. Perceived brightness is computed for each image using Finley (2006)'s[18] equation:

     $$brightness = \sqrt{0.299R^2 + 0.587G^2 + 0.114B^2}$$

     Adjusting the image's brightness has the effect of adjusting the image's contrast as well, hence we experimented with tuning the acceptable brightness range on our own diving images. Brightness and contrast are related by the following equation[19]:

     $$Brightness += int(round\left(255 * \frac{1 - contrast}{2}\right))$$

     We found that a perceived brightness range of 135-145 was ideal. Therefore, if images had a brightness below the range, we increased the brightness of the image by the difference between the image's brightness to the center of the brightness range. We also reduced the brightness of the image until it was within range if the image was too bright.
   - We measured the image's sharpness and adjusted the sharpness of the image if necessary. An image's sharpness was measured by computing the image's variance of Laplacian as

[15] Please do use our tool available on our GitHub page (notebooks/postprocessing/moviemaker.ipynb) to stitch videos with an audio of your choice from YouTube!
[16] Total score comprises of summing up scaled area and scaled weighted sum scores.
[17] See the 'get_top_frames' function in src/utils/beautify.py in our GitHub page.
[18] Finley, D. (2006). HSP Color Model — Alternative to HSV (HSB) and HSL. Extracted from: http://alienryderflex.com/hsp.html
[19] How do I increase the contrast of an image in Python OpenCV. Extracted from: https://stackoverflow.com/questions/39308030/how-do-i-increase-the-contrast-of-an-image-in-python-opencv

suggested by Rosebrock, A (2015)[20]. In his article, Rosebrock suggested that images with variance of Laplacian values below 100 be classified as 'blur'. We found that a threshold of 50 was more suitable for our use case, as further sharpening images beyond this threshold caused the image to be unnatural. If an image has a variance of Laplacian value below the threshold, the algorithm sharpens it automatically by convolving the image against a series of convolution kernels derived until the threshold is met. The algorithm returns the sharpened image from the smallest kernel that meets the threshold.

- To give users further customizability, we allow for users to add an Instagram filter of their choice on top of the processed image. The final images enhanced by the filters are then returned to the user.

An example of the entire beautification process can be seen in Annex F.

5. The 'application'

We operationalized our application with the YOLOX-Nano model on Streamlit. The interface is designed to be friendly and intuitive, allowing users to easily navigate through the application. The use of the fine-tuned lightweight YOLOX-Nano model allowed for quick, accurate inference of videos. The relevant frames were then processed and the resultant outputs returned to the user. To provide explainability to our users, we included the object detection video along with descriptive statistics and plots to allow comprehension of what is happening in the video and what is being done to the video. Additionally, we included advanced configuration options for users to specify speed of optimization, trimming strictness, the configuration of the video trimming algorithm, audio file and Instagram filter for beautification. A detailed glossary of these configurations can be referenced in Annex G.

As our account on Streamlit is a free account, we would advise prospective users to not upload diving videos beyond 100MB to prevent the app from crashing. Alternatively, one can clone and run the code locally from our GitHub page or on Colab, which allows for larger file sizes.

6. Challenges

a) **Image annotation.** It is important to keep annotation consistent and accurate throughout the dataset. Though we were aligned on annotation ground rules, decision-making in practice turned out to be quite challenging. We debated annotating a school of closely packed fish as one entity instead of individual fishes as it was very difficult to do so. Eventually, we decided to label individual fishes as we may not have enough training data for schools of fish. Heeding to the advice of labeling objects that we want the model to detect, we annotated fish that were not occluded and have all the distinct features (eyes, tail, fins, body-shape) present. We did not annotate forests

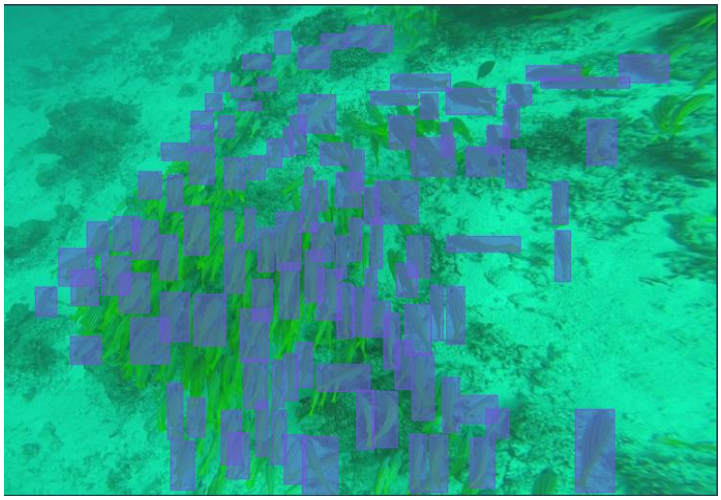


Figure 4: Annotation example for school of fish

[20] Rosebrock, A (2015). Blur detection with OpenCV. Extracted from: https://pyimagesearch.com/2015/09/07/blur-detection-with-opencv/

of corals that lack aesthetic features as we assumed leisure divers were not interested in capturing them[21].

b) Class imbalance. A novelty of our project was the use of personal diving videos for training. We believe personal videos are most representative of the footages a leisure diver has. However, encounters with large sea creatures, such as turtles, manta rays and sharks, are relatively rare. As a result, there was a large class imbalance in the annotation counts we obtained. Due to limited time to further collect and annotate data on the minority classes[22], all our models were trained on the imbalanced dataset.

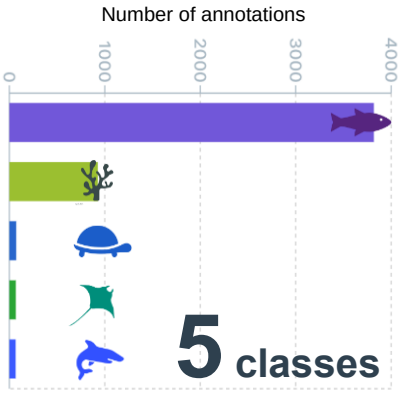


Figure 5: Distribution of classes across annotated images

c) "Beautiful" images are subjective. Thresholds for brightness and sharpness have to be manually tuned by human judgment of images produced by the beautification algorithm. These thresholds would therefore need to be re-tuned from scratch if our application were to be applied to an entirely different domain (for e.g. editing wedding videos). In our opinion, this inconvenience is unavoidable as different domains have very different expectations of what a 'good' image is. For example, an image captured while night cycling would need an even lower brightness threshold or the enhanced image would appear very artificial. Ideally, we envision to use a more mathematical approach to automatically identify and adjust the amount of brightness and sharpness in an image.

d) Handling misclassifications. We face a peculiar error. Our model regularly identifies and classifies humans in our test videos as one of the finned species (fish, shark) in our dataset. We speculate that this is because there are very little humans in our training data, the humans in the diving videos wear diving fins on their feet, and because of the color of their diving suits. This could possibly be ameliorated by using training data with more human divers so that the model learns to classify humans as the background.

7. Conclusion & Future Work

We successfully developed an AI-centric video truncation and wallpaper generation tool for recreational divers and couldn't be more thrilled with the results. To pay homage to the YOLOX model architecture for serving as the foundation of our project, we have decided to name our application "You Only Edit Once (YOEO)". We sincerely hope that recreational divers will find our tool a convenient utility and will never have to undertake a manual video editing task again.

Nonetheless, the project has room for improvement, and we have several ideas on how to do this.

a) We believe that solving chronic class imbalance can help to improve our current model performance. Therefore, an immediate step is to include more training samples on the minority classes, namely sharks, turtles and manta rays. We should also be mindful to include training samples with humans to reduce the misclassification rate of humans.

b) As future work, we intend to build a machine learning model that takes user usage data as input and churn out recommendations for users on potential optimal advanced settings, based on their historical preferences and usages. We also intend to employ a more AI-centric approach to operationalize how picturesque each individual frame is by training a machine learning regressor

[21] We acknowledge that this can be subjective, but we were unanimous in not wanting to see dull-looking corals that were undistinguishable from rocks picked out from our own diving videos.
[22] We also don't have many videos for some of these classes in our personal repository.

to predict a score for each frame, which will require much more labelled training data than what we currently have.

c) We envision applying the YOLOX object detection model to other symbiotic use cases under the YOEO umbrella. The YOEO app can be further engineered to be deployed on edge devices.
    i. Our first idea pertains to the automatic switching action cameras on and off for underwater diving video recording. Given that diving is a dangerous sport requiring much concentration and equipment, beginners usually struggle to attend to both diving equipment and their cameras. Using YOEO to video record automatically allows divers to focus on the dive, while saving (optimizing) camera battery and memory as only the scenic parts of the diving journey will be recorded. The YOLOX-Nano model used in YOEO requires small compute power and is suitable to provide fast inference on low resolution images collected, making this use case feasible.
    ii. Our second idea is an extension of the first, to use YOEO to automatically take images when the camera detects interesting frames in real time. The images can be uploaded to a repository when there is access to connectivity and the YOEO application can immediately work on beautifying these images for the user.
    iii. We propose that the YOEO application can be used on domains outside of diving, but the amount of work to generalize the YOEO application is non-trivial as there are too many different domains with too many classes of interest within them. A generic YOEO application might also perform better for some domains and worst in other domains. Nonetheless, we are encouraged by what we were able to achieve in this project and believe that it is possible for YOEO to be generalized using domain adaptation strategies[23] and meta learning algorithms[24].

We conclude by thanking our professors and Dr. James for unselfishly availing themselves for discussion and providing us quality feedback. We thank the Datature team for their prompt replies and support as well, especially in guiding us on using ONNX model objects saved from the platform to run inferences.

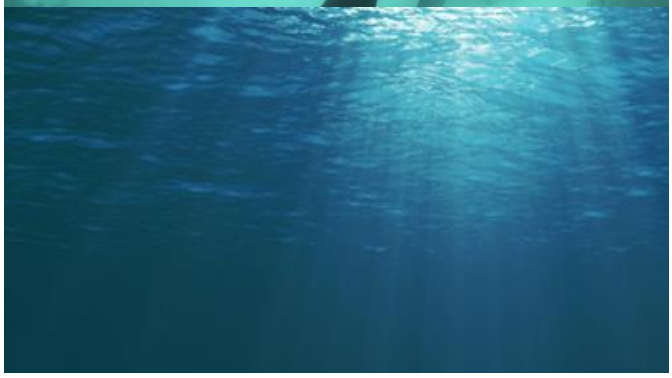
-END

[23] Oza, P., Sindagi, V.A., Vibashan, V.S., & Patel, V.M. (2021). Unsupervised Domain Adaption of Object Detectors: A Survey. https://doi.org/10.48550/arXiv.2105.13502

[24] Antonelli, S., Avola, D., Cinque, L., Crisostomi, D., Foresti, G.L., Galasso, F., Marini, M.R., Mecca, A., & Pannone, D. (2022). Few-Shot Object Detection: A Survey. ACM Computing Surveys (CSUR).

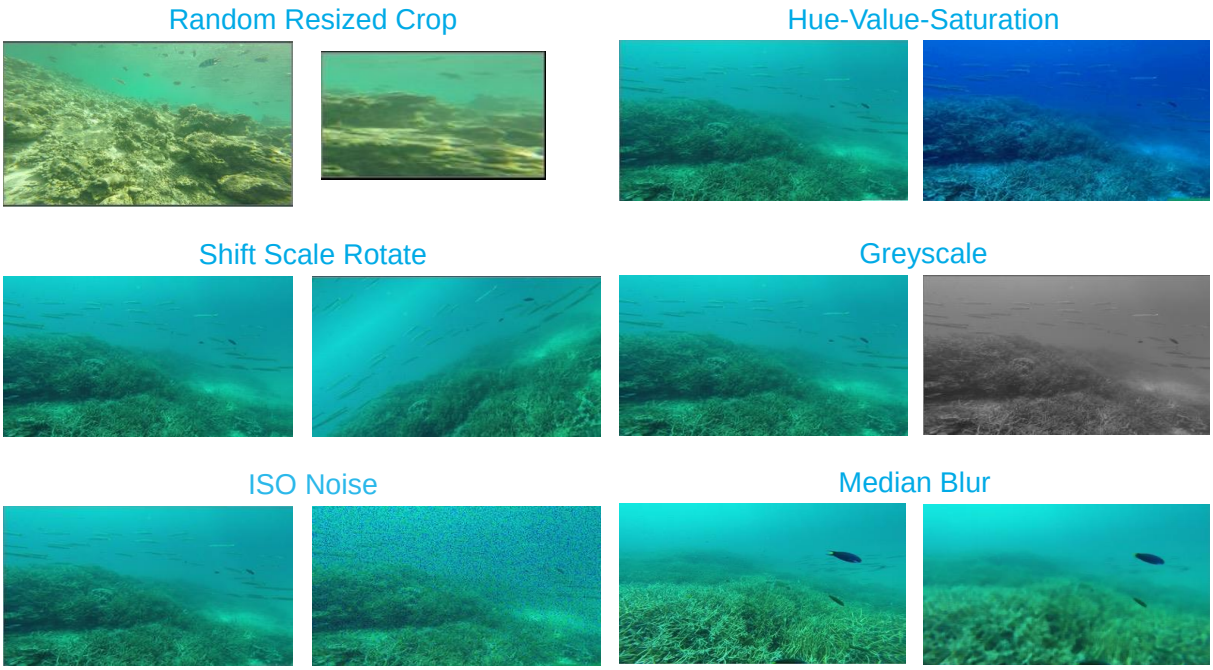
Annex A: The expectation-reality gap

Like some of our recreational diving peers, we were inspired to undertake the sport to ‘see the fishes up close’ like filmmakers of marine documentaries. However, what we saw in our dives was nowhere as colorful or as lively. Instead, marine life is ‘shy’ and tends to swim quickly away into hiding from attention. Therefore, our personal dive videos differ greatly from marine documentaries with most of the footage not being very useful. It can be quite discouraging.

What we expect to see in our dive videos	What we actually see in our dive videos
	
	

Annex B: Image Augmentation

a. Augmentations applied on the Datature platform:



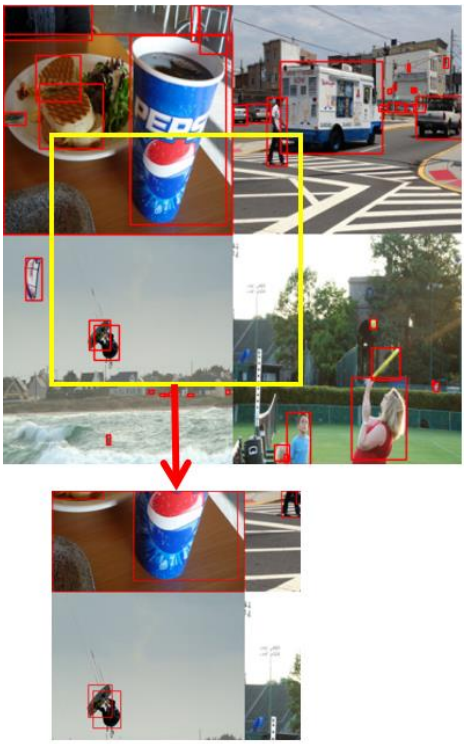
In general, we selected position and color space augmentations to achieve a good mix of augmented data for training. Since marine creatures come in various forms and sizes, position augmentation serves to generate more samples of marine creatures with various shapes. Color space augmentations such as greyscale and hue value saturation improve the robustness of the model by making it invariant to the color of the marine creatures.

As underwater videos are usually affected by dark conditions and movement, we added ISO Noise and Median Blur to generate more images that simulate such conditions.

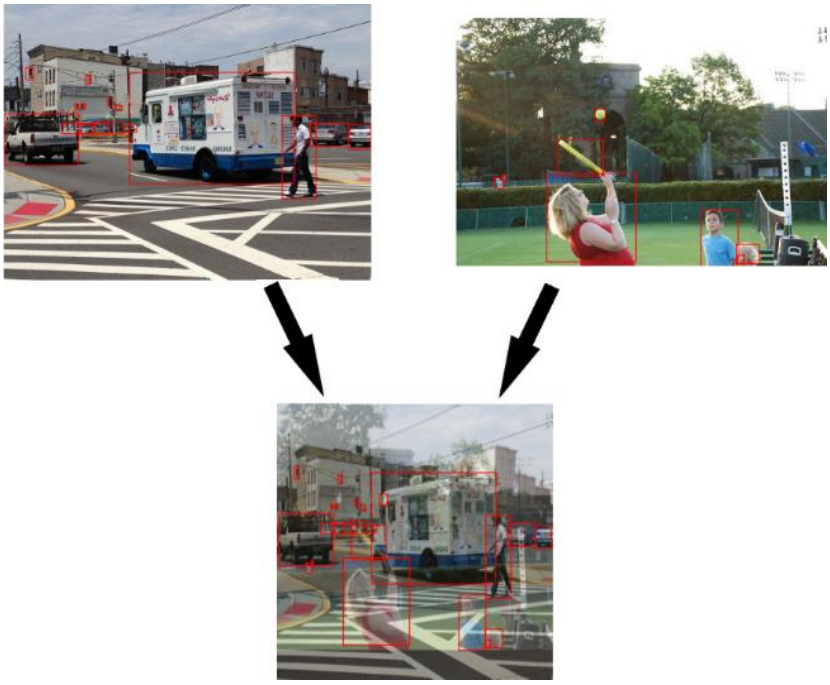
b. Default augmentations applied by the YOLOX GitHub repository:

The YOLOX GitHub repository applies further image augmentation by default, namely mosaic and mix-up image augmentation. The mosaic augmentation was conceived on the idea to allow model to learn how to identify objects at a smaller scale than normal[25], creating an augmented image by cropping an image combined from 4 images. An example of the augmented image is as follows:

[25] Solawetz, J. Data Augmentation in YOLOV4. Extracted from: https://blog.roboflow.com/yolov4-data-augmentation/



Conceived as an augmentation to improve model generalizability[26], mix-up superimposes 2 pictures through a weighted combination, creating an output picture with bounding boxes from both pictures. An example of mix-up augmentation is as below:



[26] Zhang, H. et al. (2018). Mixup: Beyond empirical minimization. ICLR 2018. Extracted from: https://arxiv.org/pdf/1710.09412v2.pdf

Annex C: Challenges faced when running inferences with YOLOX-medium

The YOLOX-medium model was trained and tuned on Datature's autoML platform. We faced several 'showstopper' challenges in using the final tuned model to run inferences.

1. Weak detection

Not all full-bodied fishes were identified despite being clearly visible with a distinctive background. Confidence scores of detected object also tend to be very low.

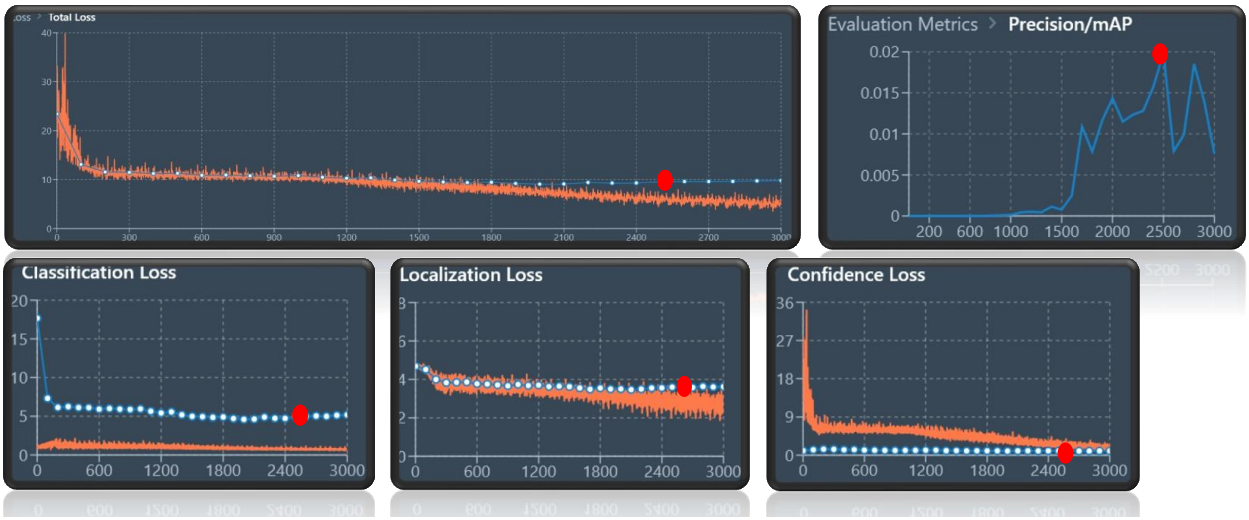
2. Loose bounding boxes

Often, we noticed that the predicted bounding box does not match the size of fish, instead, several fishes are included within the same bounding box.

3. Frequent misclassification

We were perplexed to observe that corals were frequently misclassified as fishes

4. Potential overfitting



From the mode evaluation plots furnished by Datature's AutoML platform, it would appear that model overfitting occurred at the 2,500th epoch, which is the epoch with our best mAP score. We clearly see that testing losses are always higher than training losses for classification and localization losses respectively, indicating high variance. This might explain some of the issues as stated prior.

Annex D: Hyperparameters of YOLOX training on Colab

Keys	YOLOX-nano	YOLOv5n
eval_interval	10	10
num_classes	5	5
depth	0.33	0.33
width	0.25	0.25
input_size	(416, 416)	(640, 640)
mosaic_scale	(0.5, 1.5)	1
enable_mixup	False	0
max_epoch	300	300

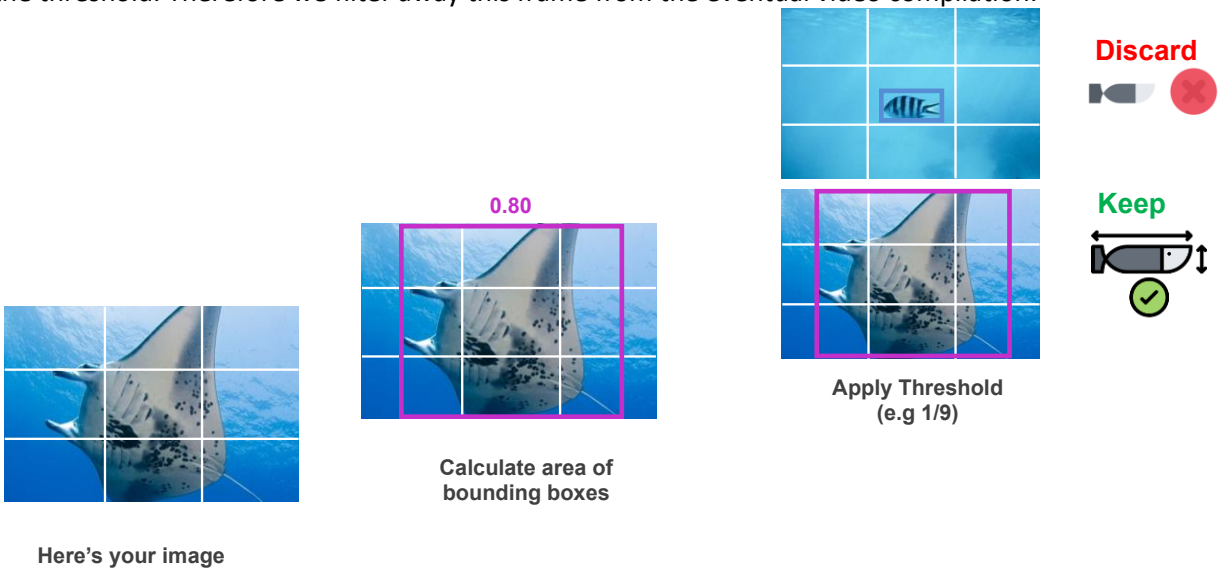
We briefly define the hyperparameters of interest as follows:
eval_interval: This is the interval of model evaluation. The default value is 10, meaning that model evaluation is executed every 10 epochs.
Depth & width: These are values between 0-1 depicting the model's scale. Larger YOLO models have higher depth and width (e.g., yolov5m depth=0.67, width=0.75).
Input_size: the image will be augmented and resized into the respective dimension required by the respective models.
Mosaic_scale: Takes in a tuple of floats as the scale range of augmentation, or an individual number. In general, higher mosaic_scale values intensifies the mosaic augmentation applied.
enable_mixup: Is a True/False/integer parameter. We disabled the mixup augmentation for YOLOX-Nano and chose not to apply mixup in the training step for YOLOv5n as it adversely impacted model accuracy.
max_epoch: Maximum number of epochs to train the model.

Annex E: Illustrative example of frame selection metrics

1. Computing area of object scores

We assume that a frame is scenic if a good portion of the frame contains the objects of interest. Hence, we took the ratio of the sum of the bounding boxes for each frame against the frame's area. We then filter away frames that do not meet a threshold.

In the diagram below, the threshold is arbitrarily set to 1/9 for illustration. Given a frame with a single manta ray, we computed that the manta ray occupied 80% of the frame's area and therefore we keep it. In contrast, another frame has a fish that occupied less than 1/9 of the frame's area which is lower than the threshold. Therefore we filter away this frame from the eventual video compilation.



2. Computing weighted count scores

We count the number of objects detected in the frame and confer a weight to each class.

In consolidation, here is an example of how the scores are computed for a video with just 10 frames that have objects detected:

Frame Number:	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
Area Scores	0.6	0.7	0.2	0.1	0.8	0.7	0.6	0.2	0.2	0.9
Weighted Count Scores	50	50	2	3	60	70	30	2	1	60
Scaled Area Scores	0.63	0.75	0.13	0.00	0.88	0.75	0.63	0.13	0.13	1.00
Scaled Count Scores	0.71	0.71	0.01	0.03	0.86	1.00	0.42	0.01	0.00	0.86
Sum Up Scores	1.34	1.46	0.14	0.03	1.74	1.75	1.05	0.14	0.13	1.86
Threshold Applied	True	True	False	False	True	True	True	False	False	True

We start by computing the area scores and weighted count scores as stated above. We then scale the respective scores of each frame against the minimum and maximum respective scores of the entire video (i.e. we scale against the min and max of all 10 frames). Following which we sum up the scores and filter away frames that are below the final threshold (in this case the threshold was set to 1).

With the frames that are left, we further mask away frames that do not carry the objects requested by the user - the user might request to see ‘shark’.

Annex F: Illustration of image sharpening algorithm

1. Original

2. Blue-red light adjusted

3. Brightness-contrast adjusted

4. Sharpness adjusted

5. 'Hudson' Instagram filter added

Annex G: Glossary of advanced configuration options

The glossary of advanced options for YOEO and their accompanying explanations are as follows: