Marine Animal Detection

Di Gu, Shuangcheng Yang, Yixun Zhang, Kunmao Li

Department of Electrical and Computer Engineering
University of California, San Diego
{dgu,shy003,yiz019,kuli}@eng.ucsd.edu

***Abstract*— Driven by the exponential growth of undersea visual data and extensive exploration of the undersea world, there is significant potential in related data analysis and processing, among which marine animal detection is one of the prevailing choices. Meaningful to the environment protection, it remains a challenging task due to the limitation like high complexity of underwater background, quality of images and videos, diverse motions of marine animals. Aiming to provide people with viable solutions to marine animal detection and support for related research, this project utilized deep-learning based techniques to detect the specific marine animals with bounding boxes in the given images and videos. The basic deep learning frameworks we applied were Faster-RCNN and YOLOv3, which proved to achieve significant improvement in mean average precision (mAP) and detection time. Our project is open source and codes are available at: https://github.com/Di-Gu/Detection.**

I. Introduction

With increasingly advanced technologies, especially in imaging systems, massive undersea visual data have been collected. It is of great importance to efficiently make full use of the exponential growth of undersea visual data so as to promote related research, enhance marine animal protection and meet the prominent demand of real-time marine scene analysis. As analyzing the data of vast size manually is not feasible, our project aims to offer new solutions to the marine animal detection and hence facilitate the monitoring of their existence and livings. The models we built for this project can also benefit the related research and call for increasing attention for marine animal protection.

Object detection, closely related to image understanding and video analysis, is an influential and extensively studied approach in computer vision. It helps to precisely estimate the concepts and locations of objects and further demonstrates their presence in given images or videos by drawing bounding boxes around the objects of interest correspondingly. Rather than well-studied and popular domains like face recognition and pedestrian detection, we focused on the marine animal detection with more complicated background instead.

Different from the traditional object detection following the three-stage pipeline, we utilized powerful deep learning techniques in order to tackle the detection and localization problems with higher precision and efficiency. The challenges foreseen by us for this project were primarily in the following aspects.

- Large variations in viewpoints, pose, occlusions and lighting conditions make object detection and localization relatively hard.
- Complicated underwater background with lots of oceanic flora increases the difficulty of marine animal detection.
- Relatively limited amount of data and larger workload of hand labeling, compared to those in well-researched domains for object detection.

In order to implement the targeted task and cope with the aforementioned challenges, we came up with a series of scheme correspondingly. As shown in figure 1, detection system for identifying and localizing the marine animals would be constructed by us. Taking the input of either images or videos with multiple marine animals belonging to disparate classes in the scene, the system based on models YOLOv3 and Faster R-CNN then detects the existence of marine animals and makes classification based on the specified categories. As a consequence, the images or videos with bounding boxes framing the animals in correct quantities and appropriate sizes will be output by the system.

This paper will be based on the following arrangement. In section II, representative works in object detection will be discussed by different categories and their correlation with our work will also be analyzed. After that, comprehensive overview with full details of our dataset and features will be given in section III. Moreover, both theoretical and technical methodologies are introduced in section IV and corresponding experiments as well as the results will be detailed in the section V. In section VI, the conclusions and future work will be correspondingly presented. You may find the contributions and references at the end of the paper.

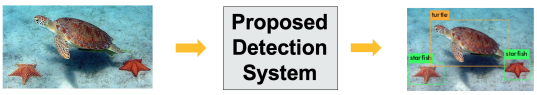

Fig. 1: General view of proposed detection system

II. Related Work

In recent years, exponentially increasing progress has been made in field of object detection. Before the deep learning method, detection algorithm often took use of low-level information like texture, color and gradient, etc. There are mainly two types of deep learning based object detection, region based and classification/regression based.

A. Region Proposal Based

Region proposal based method, the 2-stage process, gives a coarse scan of the whole scenario firstly and then focuses on regions of interest. Extracting the region of the objects, it

then compute the features and classify the extracted region to the categories the objects belong to. Among representative frameworks summarized in figure 2, R-CNN [1] by Girshick, Ross, et al is the cornerstone that intrigues extensive research afterwards. It firstly adopts selective search to produce 2k region proposals for each image and then conducts the CNN (Convolutional Neural Network) based deep feature extraction to realize high-level, semantic and robust feature representation for each region proposal. When labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.

There are also some influential frameworks worth mentioning. SPP-net (spatial pyramid pooling) [4] proposed by He, Kaiming, et al. equipped with another pooling strategy, generates fixed-length representations regardless of image size/scale and thus enhance all CNN-based image classification methods in general. For tackling the drawbacks shared by R-CNN and SPP-net that (1) training is a multi-stage pipeline; (2) training is expensive in space and time; (3) object detection is slow, Fast R-CNN[2] is thus proposed to accomplish higher detection quality through single-stage training by using a multi-task loss and update all network layers. Subsequently, Region Proposal Network (RPN) was introduce in [3] by Ren, Shaoqing, et al. which shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals and further merges RPN and Fast R-CNN into a single network. Region-based fully-convolutional networks(R-FCN)[5], Mask R-CNN[7] and feature pyramid networks (FCN)[6] are the representative successors after Faster R-CNN.

Fig. 2: Development of region based methods

B. Regression/Classification Based

Regression/Classification based frameworks are based on 1 stage, mapping straightly from image pixels to bounding box coordinates and class probabilities, which consequently reduces time expense. The influential models belonging to this category are demonstrated and listed in figure 3.

MultiBox[12], AttentionNet[13] and G-CNN[11] are three pioneers. MultiBox[12] proposed a saliency-inspired neural network model for detection, which predicts a set of class-agnostic bounding boxes along with a single score for each box, corresponding to its likelihood of containing any object of interest. AttentionNet[13] quantizes weak directions pointing a target object and the ensemble of iterative predictions from AttentionNet converges to an accurate object boundary box. G-CNN, an object detection technique based on CNNs which works without proposal algorithms was introduced in [11] and made the problem of object detection as finding a path from a fixed grid to boxes tightly surrounding the objects. The series of methods You Only Look Once, YOLO[8], YOLOv2[9] and YOLOv3[10] are the representatives of non-region based solutions after that. Single Shot Detector, SSD[14] and Deconvolutional Single Shot Detector, DSSD[15] is another series.

Fig. 3: Development of regression/classification based methods

C. Correlation with our work

As stated in the design of detection system, the fundamental frameworks we intend to utilize are Faster R-CNN and YOLOv3, which belong to the region proposal based approach and regression/classification based approach respectively. On the one hand, thanks to the instinct nature of region proposal based approach, Faster R-CNN is expected to enable us to realize ideal performance in terms of bounding box generation. On the other hand, YOLOv3 inheriting the great advantage in short training time and benefits in real-time application will offer us new insights in leveraging the training efficiency.

By implementing these two prevailing methods having their own merits and drawbacks, we can thus make significant comparison of them based on the factors like training loss, mAP (Mean Average Precision), bounding box generation and etc. Based on the practical experiment results, targeted suggestion on framework application can be made.

III. Dataset and Features

Significant details concerning the dataset and features are all summarized in the table I

Data	Explanation
Sources	1. Google Images 2. OBSEA
Format	JPEG images
Quantity	over 1000
Image resolution	416 × 416
Training set	80% data
Testing set	20% data
Cross validation	5-fold
Data preprocessing	1. Image transform 2. bounding box transform
Feature learning	CNN

TABLE I: Details of dataset and features

A. Dataset

There are mainly two sources for us to collect the data. First of all, we acquired the marine animal data scripted from Google Images. In addition, we successfully obtained the approval of using the underwater camera data from the organization OBSEA (UPC)[17]. Some randomly chosen images from our dataset are demonstrated in Figure 4.

After the data collection, we split the data into training set and testing set, whose proportions were 80% and 20%

(a) Example 1: turtle

(b) Example 2: dolphin

(c) Example 3: OBSEA

(d) Example 4: OBSEA

Fig. 4: Selected examples of our dataset

respectively. Moreover, the 5-fold cross validation was also utilized with 20% of the training data used for validation set.

In order to gain better results, it was necessary for us to pre-process the data before training our models. Since this was an object detection task, image transform and bounding box transform were applied.

B. Features

CNN is used to learn a hierarchy of features from low- to high-level concepts.

IV. Method

A. CNN

We need to give a brief introduction to what is a convolutional neural network. This network is similar to ordinary Neural Network and popular used in image processing. CNN can learn a hierarchy of features from low- to high-level concepts. In CNN, some inputs are given to each neuron and dot product method is used then applied non-linearity activation functions for each neuron. We then can take back-propagation method to learn weights and bias for each hidden layer neurons based on loss function in the fully connected layer. These parts are the same as ordinary neural network. However, there are some special features. The first one is there are three dimensions for each neuron. Except the width and height of the input picture, the additional dimension is called channel. Usually, there are two kinds of channel selections. One is grey and the other is red, green and blue. The other difference is there is a pooling layer in CNN. This layer will perform a down-sampling operation along the spatial dimensions (width, height), resulting in volume and this layer don't have bias or weights learn-able parameters.

The convolutional layer is the core building block of a convolutional network that does most of the computational heavy lifting. The convolutional layer's parameters consist of a set of learnable filters including bias and weights. Each filter is small spatially, but extends through the full depth of the input volume. In CNN, the weights are 2-dimensinal matrix and are called filters. During forward process, each layer's input volume is fixed. So we will use pre-established filters to manage the input values. Each output element is computed by element-wise multiplying input volume (take same size as filters) and filters, summing it up.

Pooling layer is commonly used in-between successive convolutional layer periodically. The most important function of pooling layers is to reduce the spatial size of the representation and then reduce the amount of parameters in the following hidden layers. The most common downsampling operation is max which gives rise to max pooling. And usually, the pooling layer filer size is a 2 by 2 filter.

Fully-connected layer is the output layer of a CNN. It has the same functional form as convolutional layer which should compute dot products between input and weight parameters. The only difference is that the neurons in the convolutional layer are connected only to a local region in the input. However, neurons in a fully connected layer have full connections to all activation functions in the previous layer.

B. Faster RCNN

In Faster RCNN, there are 3 main networks. First is pre-trained CNN (ResNet101) which is used to generate a feature map of images. The feature map is the output from the last fully-connected layer from a CNN which contains important features of the image like edges and shapes.

Then the map is fed into a region proposal network (RPN) which contains a classifier and a regressor. The key concept in RPN is anchor. Anchor is used to put bounding box with fixed size to reshaped image which is used as a reference for objection localization. For this task we used default anchor ratio [0.5,1,2] with scale [8,16,32]. With the set anchors, RPN takes reference bounding box, then output two parameters, one is the target score for if the proposed bounding box contains possible object or not, the other one is the regressed coordinates of the proposals. To be more specific, the classifier will output the score of the target as foreground or background for each proposals and the regressor will output $\triangle$x, $\triangle$y, $\triangle$weight, $\triangle$height to get a final proposal for the bounding box. Additionally, RPN is designed on fully-connected convolutional layer to achieve high performance.

After RPN, the proposals without desired class are fed into RoI-pooling model. In this step, the feature map from CNN is used. Since the proposed bounding boxes have different size, the feature map is cropped according proposals, then bilinear interpolation is used to change each cropped map to a fixed size and use max-pooling to get final feature map, where a classifier is designed based on fully-connected layer to output the score for target object.

Combine all the models together, we have four different losses, 2 for RPN and 2 for RoI-pooling. For RPN regressor, smooth L1 loss is used:

$$L_{\text{reg}}\left(t_i, t_i^*\right) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}\left(t_i - t_i^*\right) \qquad (1)$$

in which:

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \qquad (2)$$

for the classifier in RPN, the log loss (L_{cls}) over two classes (foreground and background) is used.

The combined loss definition for a image is:

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right) \tag{3}$$

, where p_i is the probability of anchor as object, $p_i^* = \begin{cases} 0 & \text{negative label} \\ 1 & \text{positive label} \end{cases}$ is the ground truth label, $t_i = \{t_x, t_y, t_w, t_h\}$ is the coordinates for predicted bounding box and t_i^* is the ground truth coordinates.

Similarly loss is used on RoI-pooling model with slight modification, where L_{cls} is a log-softmax function and L_{reg} still used smoothed-L1 loss.

Algorithm 1: Faster RCNN

```
repeat
    Feature_map ← pre_trained_CNN(image)
    RoIs ← region_proposal_network(Feature_map)
    for RoI in RoIs do
        patch ← roi_pooling(Feature_map, RoI)
        classification_loss, bbox_loss ← detector(patch)
    end
until loss converage
```

Figure 5 below shows the overall structure of a Faster RCNN model.

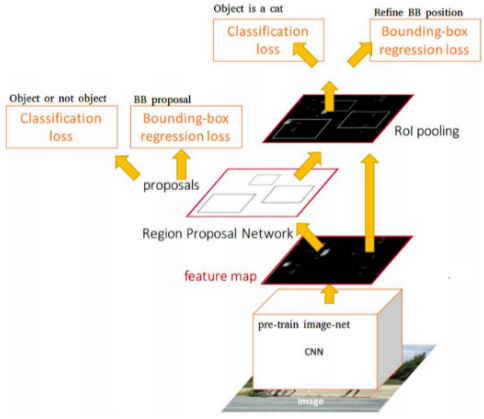

Fig. 5: Structure of Faster RCNN

C. YOLOv3

YOLOv3 model is established combining with the whole darknet53 model. As a total, YOLOv3 model has 105 convolution layers, followed by 3 fully connected layers. The actviation function for each hidden layer is ReLU function which is a function takes the maximum value in between input x and 0. Alternating convolution layers reduce the features space from preceding layers, and finally get the final output of our work.

YOLOv3 is used to predict multiple bounding boxes per grid cell. For the purpose of computing the loss for the true positive, we can select the one with the highest IoU (intersection over union) with the ground truth. This strategy leads to specialization among the bounding box predictions. Each prediction gets better at predicting certain sizes and aspect ratios.

The loss of YOLOv3 is mainly using the sum-squared error between the predictions and the ground truth bounding boxes, and it contains:

1) Localization loss: It represents the errors between the predicted boundary box and the ground truth, in YOLOv3, the square root of the bounding box width and height are predicted instead of the direct coordinates. It is described as:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right] + \lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}\left[(\sqrt{\omega_i}-\sqrt{\hat{\omega}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2\right] \tag{4}$$

where $\mathbb{1}_{ij}^{obj}$ denotes that jth bounding box predictor in cell i is "responsible" for that prediction.

2) Confidence loss: This is related to the object of the box, once an object is obtained, the confidence loss is described as:

$$\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}(C_i-\hat{C}_i)^2 \tag{5}$$

If an objected in not detected in this box, then this loss is:

$$\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{noobj}(C_i-\hat{C}_i)^2 \tag{6}$$

3) Classification loss: In each cell, it is the squared error of the class conditional probabilities[2] for every classes and it can be described as:

$$\sum_{i=0}^{S^2}\mathbb{1}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2 \tag{7}$$

where $\mathbb{1}_i^{obj}$ denotes if object appears in cell i.

Finally, we can have the whole loss function by adding

the previous loss together:

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(\sqrt{\omega_i} - \sqrt{\hat{\omega}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
&+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
\tag{8}
$$

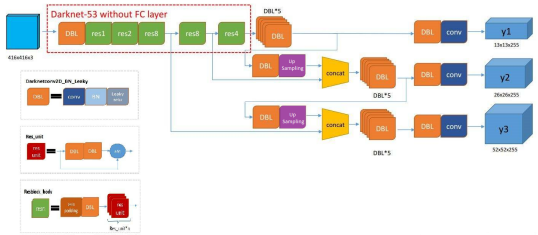

Fig. 6: Structure of YOLOv3

V. Experiment and Results

A. Experiment

1) Evaluation Metric: The most common evaluation metric that is used in object detection is 'mAP', which stands for 'mean average precision'. It is a number from 0 to 100 and higher values are better, but its value is different from the accuracy metric in classification.

We first calculate a score for each bounding box. The score is computed by using the area of overlap between predicted bounding box and ground truth box divided by the area of union of these two boxes. Then, based on the scores for each class, a precision-recall curve is computed by varying the score threshold. The average precision (AP) is the area under the precision-recall curve. Next, the AP is computed for each class and finally averaged over the different classes to get mAP. The mAP metric avoids having extreme specialization in few classes and thus weak performances in others. Note that a detection is a true positive if it has an 'intersection over union' (IoU) with the ground truth box greater than some threshold. In this project, we use mAP@0.5.

For classification, we need to use the evaluation metric called accuracy. During the validation process, soft-max function is used to classify different classes. Each class will have their own percentage to justify category in a validation image. The output of soft-max function will compare with the result of ground truth value, and the total number of different features between ground truth and predicted will be stored for each image. And finally, the total number we get above will be divided by the total number of each class in all validation images. In this way, we can get classification accuracy percentage for each class by computing one minus the percentage of error.

2) Hyper Parameters: In this project, a shuffled mini-batch size training dataset combined with stochastic gradient descent method is used. We want to learn more different features of each given training dataset, so a shuffled method is applied. Usually, a mini-batch contains 50 to 256 data points is applied. Mini-batch gradient descent is not only faster, but also finds better solutions. In this project, we use a mini-batch size 8. Stochastic gradient descent method can make training faster if the gradient of each sample is a good proxy for the total gradient. Additionally, SGD can get out of local minima as it explores the space in random fashion. Mini-batch stochastic gradient descent can smooth noise but require adaptive learning rate. The calculation method of adaptive learning rate is shown in 9. Base on the formula, there is a Adam function in pytorch that we can use in training process. It will use larger rates for directions that remain flat and smaller rates if they keep changing. In this way, the initial choice of learning rate won't be a strict criteria. Normalized initialization is applied randomly to prevent unit from learning the same thing. Normalize the inputs with zero-mean and unit variance is applied in this project.

$$
\gamma_{i,j}^{t} = \frac{\gamma^0}{\sqrt{\sum_{k=1}^{t} \left| \frac{\partial E(\boldsymbol{W}^k)}{\partial w_{i,j}^k} \right|^2 + \varepsilon}} \tag{9}
$$

B. Results

Generally, we have achieved satisfactory progress in model training and appealing results concerning bounding box generation.

1) Model training and validation: The training and validation procedures can be observed in Figure 7, in which there are two plots representing the training loss and validation Mean Average Precision (mAP) as well as Average Precision (AP) during 150 epochs respectively. As for the growth tendency of training loss, it started from around 150 firstly and soon decreased remarkably , dropping below the level of 25 at the $50th$ epoch. After that, it continuously experienced the descent trend with a smaller decrease rate and eventually reached the level close to 0. The aforementioned decreasing tendency of training loss also indicates our model based on YOLOv3 is capable of realizing fast convergence.

In order to evaluate the performance of model, we introduced two factors mAP and AP during validation. AP is defined as the under the precision-recall curve (PR curve), the x-axis and y-axis of its figure will be recall and precision respectively. Any detection with score lower than the score cutoff was regarded as the false positive. After getting all the precision-recall value pairs corresponding to the score cutoff, the AP/PR curve can be plotted. As can be seen from the graph, its increase was not monotonic but with fluctuation. After 150 epochs, it finally exceeded 0.92. Furthermore, mAP, computed by averaging all AP values over classes,

ascended much more quickly along with the epochs with a larger rising rate than that of AP, since it reached the values close to 1.0 before 50 epochs and fluctuated near the same level mildly during the rest epochs.

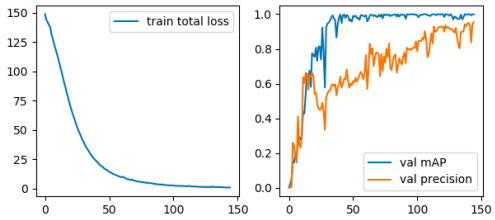

Fig. 7: Training loss and validation results of YOLOv3

2) Comparison of mAP and AP over classes for YOLOv3 and Faster R-CNN: For better understanding and more comprehensive study of these two metrics we utilized for validation, the mAP and AP values of each class for both YOLOv3 and Faster R-CNN are summarized in table II. Faster R-CNN realized slightly better performance than YOLOv3 in terms of higher mAP and AP values for two thirds of categories. Based on the ranking of AP values for all classes, we can find out that dolphin and jellyfish were two classes the easiest to be detected and precisely bounded by the box. Generally, Faster R-CNN was more accurate in detecting fish, turtle and starfish while YOLOv3 did a better job in detecting swordfish.

Self Labled Datas	YOLOv3	Faster R-CNN
mAP	95.9	96.3
Dolphin	99.7	98.7
Fish	93.4	96.2
Turtle	92.4	95.0
Jellyfish	98.7	98.8
Starfish	94.6	95.3
Swordfish	97.3	94.2

TABLE II: mAP and AP for each class

3) Bounding box generation results for YOLOv3 and Faster R-CNN: As mentioned before, our dataset is comprised of two sources, Google images and underwater images from OBSEA. Hence, we selected two images corresponding to the these two sources respectively. The demonstration images we have chosen contained large amount of marine animals with different classes simultaneously, so that the readers are able to gain a better view of how successful our model could be in marine animal detection. Figure 8 and figure 9 demonstrated the detection result on the same Google image by Faster R-CNN and YOLOv3 respectively. As shown in the figure 8, 17 bounding boxes generated by Faster R-CNN successfully caught the marine animals with categories of fish, turtle and starfish while in figure 9, fish and turtle were detected by 7 valid bounding boxes by YOLOv3. Therefore, Faster R-CNN is superior to YOLOv3 in both the quantity of valid bounding boxes and the number of classes successfully detected in such an image with dense occurrences of objects.

Fig. 8: Sample output based on Faster R-CNN

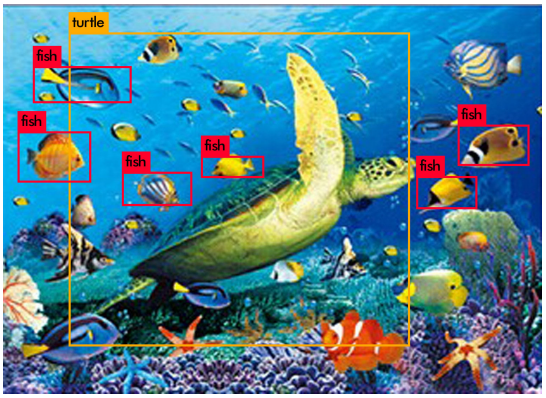

Fig. 9: Sample output based on YOLOv3

Different from Google images, the underwater images collected by OBSEA had darker background and inferior lighting conditions. This could help us assess the performance of our model in challenging occasions and thus made adjustment to enhance its robustness. As shown in figure 10, the difficulty of animal detection task got increased appreciably by the somber background with bad lighting and dense occurrences of animals. However, YOLOv3 could conduct a really perfect job in detecting all the existing animals precisely by the bounding boxes with appropriate sizes.

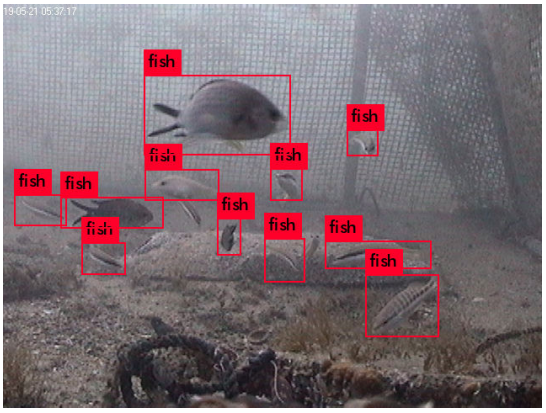

Fig. 10: Sample output based on YOLOv3

Taking aforementioned results into consideration, our results were ideal and appealing in the majority of cases where

there might be multiple and overlapping animals appearing in the same image which suffered from somber background and dark lighting.

VI. Conclusion and Future Work

A. Conclusion

In this project, Faster R-CNN and YOLOv3, belonging to region proposal based and regression/classification based frameworks were utilized for the construction of detection model taking input of images/videos containing multiple marine lives and outputting them with precise bounding boxes. Following the methodology and experiment setting introduced previously, we achieved satisfactory and appealing results in terms of training and bounding box generation. Based on the results, the following conclusion can be acquired.

1) Faster R-CNN has a higher mAP in object detection, which indicates it performs better in bounding box generation in terms of sizes and locations.

2) YOLOv3, which is obviously faster than Faster R-CNN can greatly reduce the time expense.

3) Basically, Faster-RCNN is a region based convolutional neural network, it can have a relatively better feature hierarchy than YOLOv3 model. In contrast, YOLOv3 is much faster because it only uses convolutional layers.

4) The selection of models depends on your specific application of detecting marine animals. That is to say, if you focus on identifying the specified marine species precisely in the underwater scene, then go for Faster R-CNN. If you pay more attention to the real-time detection of marine lives, use YOLOv3 instead as it can still ensure good accuracy.

B. Constraints and Future Work

Though the results are relatively satisfying till the moment, there still exist some constraints:

- Current existing detection model is not good enough.
- Limited dataset and limited class numbers. It is really a small category set.

The following work is supposed to be finished in the future:

- A baseline model should be developed for simple detection tasks
- More data are supposed to be collected and labeled and can be either test data set or train data set.
- More related work will be carried out to optimize the performance of the model.
- The model will be further improved based on the constraints, like improving the accuracy in complicated background.
- The project will be further promoted into a real-world application. For example, now the marine animals are classified to 6 categories and in the future, the categories are supposed to increase to the number in reality.

VII. Contributions

- Di Gu: training and testing on Faster RCNN, C++ based YOLOv3
- Shuangcheng Yang: YOLOv3 model training and testing using Pytorch
- Yixun Zhang: data processing, finalizing documents
- Kunmao Li: data collection, finalizing documents

References

[1] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[2] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[3] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems. 2015.

[4] He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37.9 (2015): 1904-1916.

[5] Dai, Jifeng, et al. "R-fcn: Object detection via region-based fully convolutional networks." Advances in neural information processing systems. 2016.

[6] Lin, Tsung-Yi, et al. "Feature pyramid networks for object detection." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[7] He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017.

[8] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[9] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[10] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[11] Najibi, Mahyar, Mohammad Rastegari, and Larry S. Davis. "G-cnn: an iterative grid based object detector." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[12] Erhan, Dumitru, et al. "Scalable object detection using deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[13] Yoo, Donggeun, et al. "Attentionnet: Aggregating weak directions for accurate object detection." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[14] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016.

[15] Fu, Cheng-Yang, et al. "Dssd: Deconvolutional single shot detector." arXiv preprint arXiv:1701.06659 (2017).

[16] Shen, Zhiqiang, et al. "Dsod: Learning deeply supervised object detectors from scratch." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[17] The Western Mediterranean Expandable Seafloor Observatory (obsea), http://www.obsea.es

[18] CS231n: Convolutional Neural Networks (CNNs/ConvNets) from Stanford University, http://cs231n.github.io/convolutional-networks/

[19] Pytorch-YOLOv3, https://github.com/eriklindernoren/PyTorch-YOLOv3

[20] A Faster Pytorch Implementation of Faster R-CNN, https://github.com/jwyang/faster-rcnn.pytorch/tree/pytorch-1.0

[21] Yolo-v3 and Yolo-v2 for Windows and Linux, https://github.com/AlexeyAB/darknet