Team 2: Applying Bounding Box Detection on the Invasive COTS Starfish to Protect Coral Reefs

Luke Wood
l2wood@ucsd.edu

Harsh Kishorsingh Thakur
hthakur@ucsd.edu

Sandra Villamar
svillama@ucsd.edu

Abstract

Due to the diminishing population of the Charonia Tritonis sea snail, the population of Crown-of-Thorns starfish (COTS) has grown rapidly across all regions of the Great Barrier Reef. COTS eat coral which ultimately destroys coral reef ecosystems. The species pose a massive existential threat to the Great Barrier Reef. The traditional way of detecting COTS is the "Manta Tow" method, in which a diver records sightings while being physically towed behind a boat. This method is clearly inefficient, unreliable, and costly. In order to automate this process, our team has produced a deep learning bounding box detection model to detect COTS from the reef images.

The code for all of our experiments is available in our GitHub repo as: https://github.com/lukewood/reef-net

1 Introduction

Australia's Great Barrier Reef is the world's largest coral reef system, containing a unique range of ecological communities, habitats, and species. Over half a billion people rely on coral reefs for food, income, and protection against storms and erosion along coastlines [NOA]. Unfortunately, the Great Barrier Reef is under threat in part due to the overpopulation of coral-eating Crown-of-Thorns starfish (COTS). One of the largest sea stars in the world, COTS can have up to 21 arms covered in poisonous spines. They are an integral part of the reef ecosystem, but their population exploded when their chief predators, the Charonia Tritonis sea snail, were excessively hunted by shell collectors. This has created an imbalance in the ecosystem.

Furthermore, recent increases in ocean temperatures due to human activities have caused massive coral bleaching throughout the Great Barrier Reef. Corals can survive a bleaching event, but they are under more stress and are subject to mortality. Millions of corals have already died, but we have the ability to slow down this decay if we control other factors that contribute to the deterioration of the Great Barrier Reef. Hopefully, by controlling the outbreak of COTS, the Great Barrier Reef will have the opportunity to recover. Due to the overpopulation of COTS, a large-scale intervention program was established to control COTS outbreaks to ecologically sustainable levels. The traditional way of surveying COTS is the "Manta Tow" method, where divers are towed along the reef, pausing ever so often to dive down and record how many COTS are visible. With such a large ocean bed, this method is clearly inefficient, unreliable, and costly. In order to improve the efficiency and scale of identifying COTS outbreaks, the Great Barrier Reef Foundation has collected thousands of reef images taken with underwater cameras and published this data set [Liu+21] to a Kaggle competition, in hopes of adopting a new detection method through AI technology. The Kaggle data set consists of 23k images, around 4k of which contain at least one COTS.

We use a RetinaNet to perform object detection in an attempt to localize COTS. A sufficiently performant model will obsolete the "Manta Tow" method, greatly improving the efficiency and reducing the cost of COTS detection.

Our contributions are as follows:

Preprint. Under review.

1. a data loader to produce a TensorFlow data set from the raw data
2. basic preprocessing and image augmentation utilities
3. ground truth and detection visualization tools
4. a custom Keras implementation of the RetinaNet algorithm
5. robust metric tracking, all in the TensorFlow graph using KerasCV's COCO metrics implementation [WZC+22]
6. a Keras callback to visualize results each training epoch
7. a custom tool to produce train-test splits for the TensorFlow Great Barrier Reef data set

2 Related Works

There exist countless publications on deep learning based works on bounding box detection methods. The MSCOCO challenge popularized the problem[Lin+14]. Alongside the MSCOCO object detection challenge, the dataset publishers also published a set of evaluation metrics: Mean Average Precision and Recall[Lin+14]. Our work is based on RetinaNet proposed by Li et al[Lin+17]. The loss is also proposed by Li et al in the same work.

Other modern bounding box detection models include the two stage Faster-RCNN[Ren+15], YOLOv5[BWL20] and YOLOx[Ge+21]. Other similar tasks include *Keypoint Detection* and *Semantic segmentation*[Min+22].

3 Problem Formulation

In this project we are trying to achieve the task of object detection. Object detection consists of two sub-tasks: **Object Localization**: locating one or more target objects within the image. *Input*: an image with zero or more objects *Output*: zero or more bounding boxes represented by four number tuples. These are often represented as (x, y, width, height) coordinates of the bounding boxes

Object Classification: classification of the objects inside of each predicted bounding box. *Input*: section of the image obtained from the corresponding bounding box *Output*: class to which the objects in the predicted bounding boxed belongs to

Object detection is achieved by performing both of these tasks at once. In short, the task to object detection can be defined as transforming from an input image to one or more bounding boxes and accompanying class labels. In our case, the only valid class label is COTS starfish.

In mathematical terms the goals of Object Detection is as follows for each image:

Object Localization: $X = image$
$y = bounding_boxes$
$y\_pred = \Theta(X)$
$\Theta \ni min(MSE(y\_pred, y)$

Object Classification: $X = image$
$y\_pred = \Theta(X)$
$\Theta \ni min(CrossEntropy(y\_pred, y)$

The goal of bounding box detection is to achieve a parameterization of Θ to satisfy these conditions.

4 Method/Approach

We propose a bounding box prediction model to predict the presence and position of COTS along with a confidence score for each identified starfish. This is done using a RetinaNet model implemented in Keras that we refer to as ReefNet. Despite being a one stage object detection model, RetinaNet is the current state of the art object detection model, achieving the highest mean average precision (MaP) of any model on the *MS COCO* data set [Lin+17][Lin+14].

Our repository consists of a data loader, basic augmentation pipeline, RetinaNet model, visualization callbacks, and an evaluation pipeline. Figure 1 provides an overview of the logical flow of our approach.

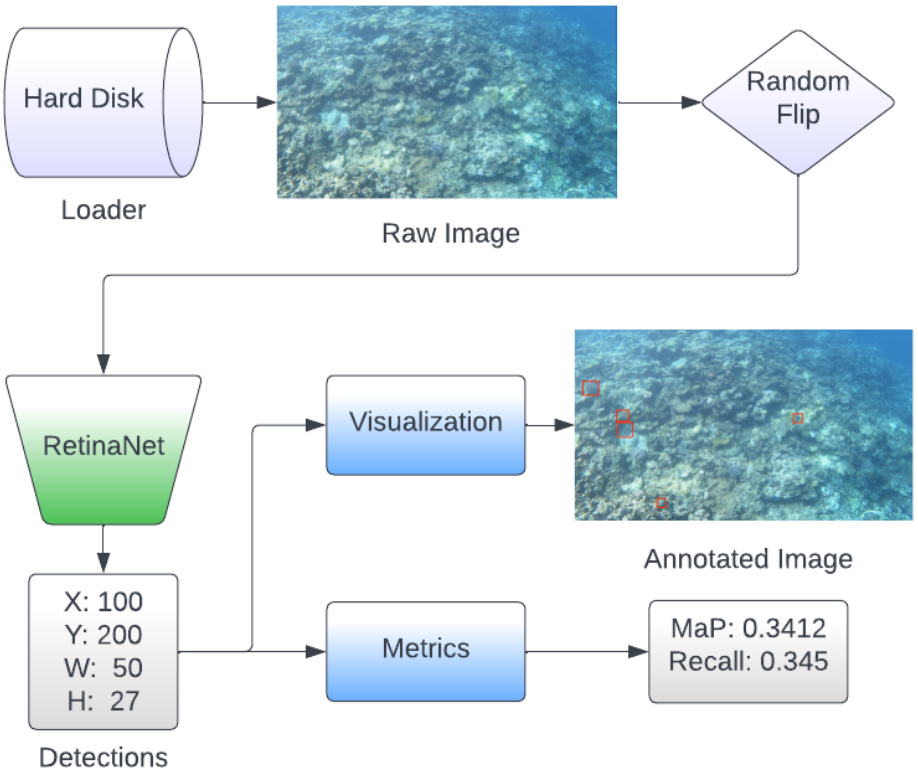


Figure 1: Overview of our Approach

4.1 Data and Preprocessing

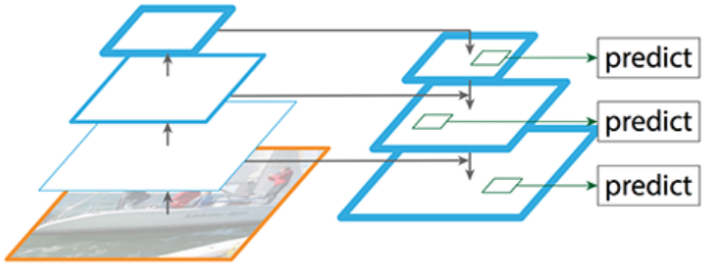
The raw data set consists of 23,501 underwater images taken at various times and locations around the Great Barrier Reef, as well as metadata for the images which include the associated video and sequence IDs, along with the bounding box coordinates of any starfish detections. Figure 2 displays two samples from the data set overlaid with the annotated bounding boxes, each identifying a starfish. We split the data set into 80% training, 10% validation, and 10% testing with about 20K examples for training and 2K examples for validation and testing.

In order to apply augmentations to our data set, we first convert the bounding box format from (x-coordinate of upper left corner, y-coordinate of upper left corner, width, height) in pixels to (x-coordinate of upper left corner, y-coordinate of upper left corner, x-coordinate of lower right corner, y-coordinate of lower right corner) in relative units to image shape. Next, we apply two types of augmentations; The first horizontally flips the image and corresponding bounding boxes with a 50% chance. The second resizes and pads the image to a given shape, using a random jitter for the smallest dimension. Lastly, we convert the format of the bounding boxes to (x-coordinate of center, y-coordinate of center, width, height) in relative units to image shape.

4.2 Model

ReefNet is an implementation of the RetinaNet object detection architecture. Although RetinaNet is a single-stage object detection model, it still manages to obtain exceptional MaP and recall by leveraging a Feature Pyramid along with the focal loss function [Lin+17]. Our Feature Pyramid

Figure 2: Sample Detection Image With Overlaid Bounding Boxes



The feature pyramid network detects objects of varying sizes by reconstructing high-resolution layers from layers with greater semantic strength. (Figure source, modified.)

Figure 3: Sample Feature Pyramid Network

Network follows the architecture shown in 3. This allows us to perform real-time inference while achieving high performance metrics. Our model in particular uses a ResNet50 encoder in the Feature Pyramid [Lin+16][He+15].

One key component of object detection networks are anchor boxes: fixed-size boxes that are used to predict bounding boxes.

Anchor boxes are used in RetinaNet in the following way: First, the model predicts whether or not a given anchor box contains an object. Following this, the model predicts the difference between the object center, width, and height from the anchor box.

During training, ground truth labels are transformed into RetinaNet prediction targets. The model generates 202,554 anchor boxes with varying sizes and corresponding aspect ratios for each training image. This is done using an AnchorBox generator, which produces 9 anchor boxes for each sampled location feature in a given feature map. The generator samples locations in strides of varying sizes, from 9 pixels for small boxes to 256 pixels for large boxes. The anchor boxes are also generated with areas ranging from $1024\ px^2$ to $262k\ px^2$. The generated bounding boxes are then assigned to either the background class (an ignore class) or the COTS class, depending on how high their intersection over union (IoU) ratio is with the ground truth bounding box. Finally, these generated boxes are used to produce box prediction and classification targets. This process matches the label encoding procedure used in the original RetinaNet implementation. These targets are then used in training alongside the RetinaNet loss, fully described in Section 4.3.

4.3 Loss

RetinaNet is typically trained using two loss functions: a classification and a box loss. The combination of these two losses is colloquially referred to as FocalLoss. FocalLoss applies a modulating term to standard cross entropy in order to focus learning on hard misclassified examples. In the case of ReeNet, the COTS examples are considered the hard examples . Focal loss is useful for cases where

there is a class imbalance and is hence particularly common in object detection, as most of the image belongs to the background class and only a few pixels in the image contain the object of interest.

The classification loss is used to identify whether the predicted bounding box belongs to the background or contains a starfish. As we are only detecting starfish, any box not containing a starfish, i.e. classified as background, is only considered in the classification loss.

Additionally, we have the box loss, which is applied to all bounding boxes that should have been classified correctly as starfish. This loss allows the model to produce more accurate predictions for each detected COT starfish. We implement box loss with smooth L1 loss, which acts as L1-loss when the absolute value of the argument is large, generating steady gradients, and acts as L2-loss when the absolute value of the argument is close to zero, causing less oscillations during updates.

4.4 Training

In our experiments we trained two models. These models are trained using identical procedures, with the exception of optimizer choice. One model is trained using the Adam optimizer with a $learning\_rate = 0.001$, $\beta_1 = 0.9$ value of 0.9, $\beta_2 = 0.999$ value of 0.999, and an $\epsilon = 1^{-10}$. The SGD optimizer is configured with $momentum = 0.9$, a custom learning rate schedule with learning rates [2.5e-06, 0.000625, 0.00125, 0.0025, 0.00025, 2.5e-05], and learning rate boundaries set at [125, 250, 500, 240000, 360000] steps respectively.

Both models are trained with a batch size of 2 due to the high memory requirements of the encoded labels. Our models are trained to convergence when possible, which typically occurs in under 100 epochs. ReefNets are trained only on images with at least one COT starfish present. Images are also fed through the data augmentation pipeline which performs a random flip with a 50% probability.

During training, the models generate predictions. These predictions are used to evaluate the MSCOCO evaluation metrics: MaP and Recall. These predictions are also used to generate visualizations, which are ultimately merged into a video that showcases the training process of the ReefNet. For real time loss and metric monitoring our metrics are exported to both TensorBoard and Weights and Biases. An example video is available in the ReefNet GitHub repository.

5 Results

The model trained using Adam performs extremely poorly. Figure 4 shows that the loss never stabilized, and also converged at an extremely high value. In addition to the loss of the Adam trained model having extremely high variance throughout the entirety of training, the loss also failed to converge. The same pattern is present in the validation loss.

Compare this to the SGD trained model, shown by the yellow line in figure 4. The SGD trained model achieves a much more stable learning curve. The SGD trained model is also able to converge on a significantly lower loss. This pattern holds when examining the box loss, also shown in figure 4, as well as the classification loss.

During training, our custom Keras visualization callback performs model inference on a single image on each epoch end. Figure 6 shows the ground truth of one such image (left) as well as our models prediction at the final training epoch (right). Our GitHub repository also contains a video showcasing predictions after each training epoch of the final SGD model.

Figure 5 shows that ReefNet successfully converges, while figure 7 shows the results of both the MSCOCO Mean Average Precision and MSCOCO Recall evaluation metrics[Lin+14]. Reefnet uses the KerasCV COCO metrics implementation[WZC+22]. Our model quickly achieves a high Recall score, indicating that the model is able to consistently locate over 70% starfish. Unfortunately, ReefNet's Mean Average Precision score is significantly weaker. The discrepancy between Recall and Mean Average Precision indicates that the model is likely performing too many false positive predictions. Figure 6 shows that this is indeed the case, with the model having made several false positive predictions on the right hand side of the image.

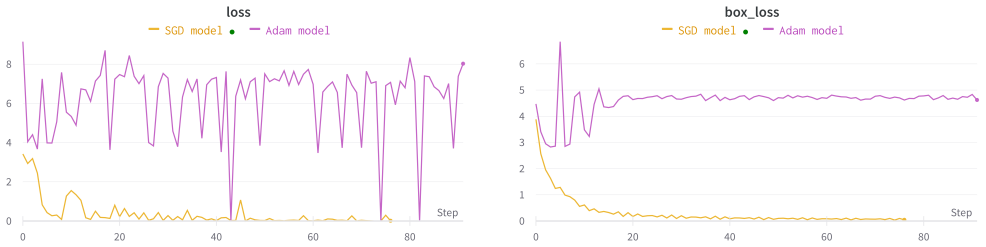


Figure 4: Loss Comparison of the SGD and Adam Trained Models

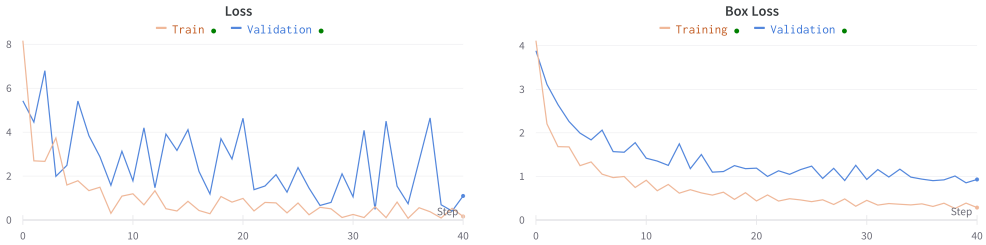


Figure 5: Loss and

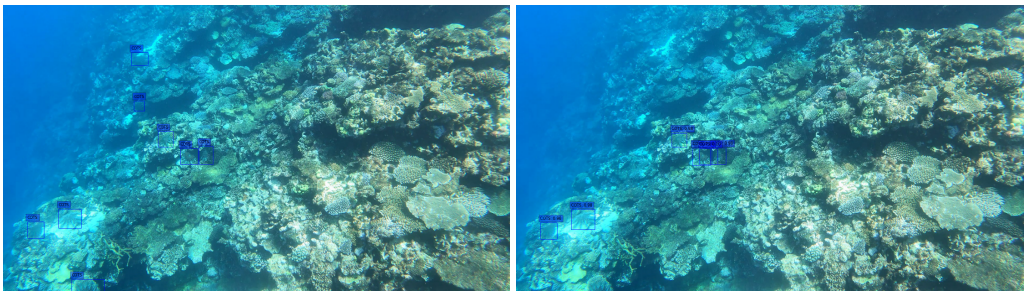


Figure 6: Ground truth (left) and ReefNet predictions (right)

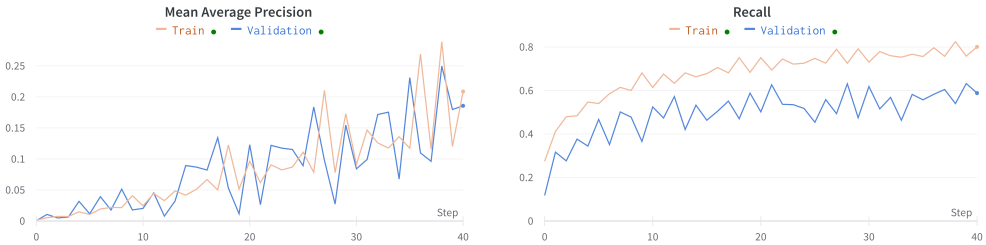


Figure 7: MSCOCO Evaluation Metrics

6 Conclusions and Next Steps

Despite achieving a low loss and strong Recall, ReefNet does not currently achieve strong Mean Average Precision. Them model also converges on a substantially higher validation loss than training loss, as seen in figure 5. These factors both indicate that ReefNet is likely suffering from overfitting. Although our data set includes 23k images, only 4k of those contain starfish and all of the images were generated from three continuous videos.

This implies several issues: particular starfish present in these videos consistently show up in the same region of the video, all frames from one video have the same lighting conditions and hence same color distribution, and overall training data is extremely limited. In order to remedy these issues, a good future step would be to introduce stronger data augmentation techniques. Based on the results we have observed, We recommend the use of several specific augmentation techniques:

Mosaic Augmentation. The Mosaic augmentation, first proposed in the YOLOv4 paper [BWL20], simply cuts out different pieces of training images and meshes them together. In the case of our data set, this will be incredibly powerful—it will allow us to re-use starfish that only appear in a specific region of images and apply them across all training images.

RandAugment. RandAugment is a data augmentation technique consisting of both spatial and color distortions [Cub+19]. It corrupts images in various ways, producing numerous similar images from a single image. During train time, RandAugment samples n operations the list of color and spatial distortions. Following this, it samples magnitudes from a normal distribution for use with each of these operations. The way magnitude is used is specified on a per-operation basis; for example, in the Shear X operation the magnitude is used to determine how aggresively the image should be sheared. By distorting training images using RandAugment, the resulting model tends to be more performant on images not included in the original training set.

CutMix. CutMix is a regional dropout strategy [Yun+19]. CutMix simply cuts a rectangle from one image and overlays it onto a target image. Any bounding box labels present from within the CutOut section are used in the resulting image's detection targets. CutMix forces classifiers to become more strongly localized. Similar to Mosaic augmentation, this will allow us to remedy the issue of starfish only showing up in a limited number of locations in our training data set.

MixUp. MixUp is a technique to mitigate against overfitting and memorization. MixUp blends the pixel values between two images into a single resulting image. This forces models to favor simple linear behavior in between training examples [Zha+17]. In the case of bounding box detection, MixUp simply uses the labels for both images as its prediction targets. Keras already offers a MixUp implementation with bounding box support [WZC+22].

Closing Thoughts. In this work we contribute a custom RetinaNet model written in Keras. In order to combat gradient explosion and numerical overflow we implement global norm gradient clipping. To further improve visibility into the learning process, we provide prediction visualization Keras callbacks. Our model has sophisticated metric tracking achieved through the use of Keras train_step overriding. Our training script integrates with Weights and Biases as well as TensorBoard to produce interactive metric dashboards.

Our model performs poorly when trained with the Adam optimizer [KB14], converging on a relatively high loss. On the other hand, training with the SGD optimizer [Rud16] yields strong results, with the model converging on a near-zero loss.

Our model has issues scoring a high Mean Average Precision. This is likely due to the small volume of training data and lack of data augmentation techniques present in our pipeline. This claim is supported by figure 5, which shows a massive discrepency between training and validation loss. Follow up works should implement data augmentation techniques such as the Mosaic Augmentation, RandAugment, CutMix, and MixUp. Combining these augmentations will produce a more robust version of ReefNet.

7 Individual Contributions

Luke Wood. wrote the outline for the launcher, the custom train and test steps for the RetinaNet class, wrote the data loader that produces a TensorFlow dataset from the raw dataset, implemented

sophistcated metric trackign to debug the NaN issues, implemented gradient clipping, generated the Weights and Biases report, and integrated the COCO metrics, implemented the visualization callback, and generated the training video. Luke also worked a great deal on the midterm poster and final paper.

Harsh Kishorsingh Thakur. got the original model training on GPU, contributed to the data loader, played a large role in debugging format swap errors, worked on the train test split tool, and put a great deal of effort into getting the TensorFlow models package to work. Unfortunately, TensorFlow models has many compatibility issues with different versions of TensorFlow, and we were unable to resolve them. This was one of our reach goals. Harsh also worked a great deal on the midterm poster and final paper.

Sandra Villamar. contributed a great deal of writing to both the proposal, milestone, and final report writing. Sandra also contributed a great deal of code to the original RetinaNet implementation, before we moved to the model subclass implementation. Sandra also put substantial effort into a YOLOv5 implementation, however due to the encroaching deadline we directed our resources into perfecting the RetinaNet implementation.

References

[KB14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[Lin+14] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. DOI: 10.48550/ARXIV.1405.0312. URL: https://arxiv.org/abs/1405.0312.

[He+15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[Ren+15] 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.

[Lin+16] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2016. DOI: 10.48550/ARXIV.1612.03144. URL: https://arxiv.org/abs/1612.03144.

[Rud16] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[Lin+17] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2017. DOI: 10.48550/ARXIV.1708.02002. URL: https://arxiv.org/abs/1708.02002.

[Zha+17] Hongyi Zhang et al. *mixup: Beyond Empirical Risk Minimization*. 2017. DOI: 10.48550/ARXIV.1710.09412. URL: https://arxiv.org/abs/1710.09412.

[Cub+19] Ekin D. Cubuk et al. *RandAugment: Practical automated data augmentation with a reduced search space*. 2019. DOI: 10.48550/ARXIV.1909.13719. URL: https://arxiv.org/abs/1909.13719.

[Yun+19] Sangdoo Yun et al. *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. 2019. DOI: 10.48550/ARXIV.1905.04899. URL: https://arxiv.org/abs/1905.04899.

[BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[Ge+21] Zheng Ge et al. *YOLOX: Exceeding YOLO Series in 2021*. 2021. DOI: 10.48550/ARXIV.2107.08430. URL: https://arxiv.org/abs/2107.08430.

[Liu+21] Jiajun Liu et al. *The CSIRO Crown-of-Thorn Starfish Detection Dataset*. 2021. arXiv: 2111.14311 [cs.CV].

[Min+22] Shervin Minaee et al. "Image Segmentation Using Deep Learning: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (2022), pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.

[WZC+22] Luke Wood, Scott Zhu, François Chollet, et al. *KerasCV*. https://github.com/keras-team/keras-cv. 2022.

[NOA] NOAA. *Coral Reef Ecosystems*. https://noaa.gov. URL: https://www.noaa.gov/education/resource-collections/marine-life/coral-reef-ecosystems.