A Machine Learning Approach for Automatic Annotation of Kina and Long-Spined Sea Urchins in Australia and New Zealand

Kelham Rawlinson

Abstract

Manual annotation of sea urchin species such as *Evechnius Chloroticus* and *Centrostephanus Rodgersii* is crucial for understanding and locating urchin barrens but very time-consuming. Past research has proven that machine learning object detection algorithms can successfully (and rapidly) detect and classify marine organisms. However, this has yet to be applied to urchin species relevant to Australia and New Zealand. In this project, we developed and evaluated a YoloV5 object detection model trained on a dataset of urchin images from Tasmania, New South Wales and New Zealand. We also identify the areas where it is likely to fail or produce errors and suggest areas of improvement and further refinement.

Introduction

Sea urchin species such as Kina (*Evechinus Chloroticus*) and long-spined sea urchin (*Centrostephanus Rodgersii*) pose a significant threat to New Zealand and Australia's underwater ecosystems. When sea urchin populations are left unchecked, destructive grazing on kelp can transform previously healthy kelp forests into regions known as urchin barrens. However, determining the extent of this problem and locating urchin barrens are challenging tasks. It requires ecologists or trained volunteers to comb through substantial amounts of benthic imagery to label and annotate the urchins they find. This can be very monotonous and time-consuming, but it is necessary to determine the size and location of urchin populations.

One possible solution to this issue is to leverage machine learning methods for automatic sea urchin detection. Recent research [1] has shown that deep learning techniques, such as deep convolutional neural networks, can be successfully applied to marine tasks. More specifically, strong results have been obtained in both classification [2] and detection [3], [4], [5] of zoobenthos (including echinoderms such as sea urchins). However, to the best of our knowledge, attempts have yet to be made to implement an object detection algorithm for urchin species relevant to Australia (*C. Rodgersii*) and New Zealand (*C. Rodgersii and E. Chloroticus*).

In this report, we will discuss the training and testing of object detection models trained to detect/locate urchins in benthic imagery and classify their species. We will also discuss the model modifications we experimented with and how they impacted performance. Lastly, we identify the model's points of failure in the dataset and potential areas for future development or research.

Dataset

The dataset used in this study was amalgamated from three separate datasets, comprising annotated benthic imagery from New South Wales, Tasmania and New Zealand. Each image in the dataset contains a set of points noting the location of urchins, each with a bounding polygon (typically a circle or rectangle), a species label (*E. Chloroticus* or *C. Rodgersii*), and

a likelihood of containing an urchin (between 0 and 1). The dataset used in the training and testing of the final set of models was up to date as of 11/01/24.

Several cleaning/preprocessing operations were applied to the dataset before it was utilised. Firstly, non-rectangular bounding polygons were converted to boxes, and boxes extending over the images' edges were clipped. Both operations were done due to constraints of object detection models, namely only predicting rectangular bounding boxes and predictions remaining entirely inside image boundaries. Secondly, boxes labelled as having a low likelihood (less than 0.7) or boxes flagged for review were removed. This decision was made based on earlier experiments training object detection models on the full dataset. In almost all cases, the model would miss boxes with a low likelihood or boxes that were flagged. Upon visual inspection of these boxes, they were often very hard to distinguish from the background, so developing a model to detect those kinds of boxes while also not making a significant number of false positive predictions would be challenging. For these reasons, these boxes were removed. Lastly, images that contained EXIF rotation data were rotated accordingly, and then that data was stripped.

In the final processed dataset, there are 5,928 images, 1,653 of which are empty (contain no urchins). Of the 27,906 bounding boxes, 10,810 contained *E. Chloroticus* and 17,096 contained *C. Rodgersii*. 80% of the data was put into a training set, 10% was put into a validation set to evaluate the model during the development process, and the final 10% was put into a test set to be an unbiased final measure of performance.

Validation

To measure the performance of an urchin detector model, predictions are made on either the validation set or the test set for the final set of metrics, which contains images that were not seen in training. These predictions are then compared to the ground truth labels from the dataset.

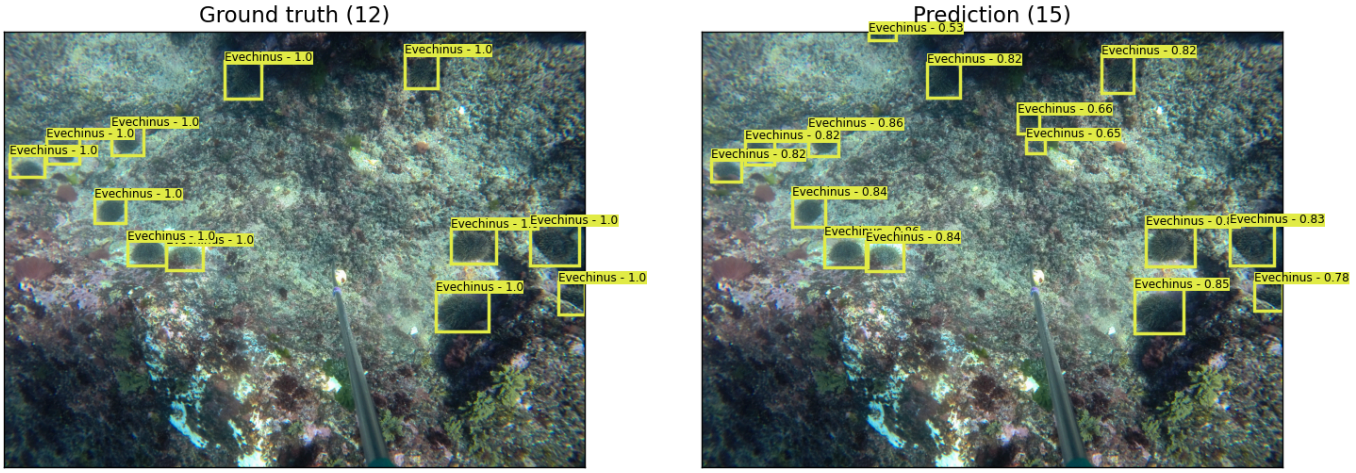

Fig. 1. Example ground truth (left) vs model output prediction (right) comparison.

To determine which predictions in an image are correct, each box is matched up with the ground truth box with which it most overlaps. This overlapping is measured with a number called IOU (intersection over union), which ranges from 0 to 1 and is calculated by dividing the area of the intersection of the two boxes (0 if they do not overlap) by the total area the

two boxes cover (the area of the union). For each of these prediction-ground truth pairs, if the IOU value is greater than or equal to 0.5 and the species predicted is correct, that prediction is correct. This is referred to as a true positive. If these criteria are not met or the predicted box has no match, then that box is a false positive. Lastly, if a ground truth box does not have a match or its match does not meet the criteria, then it is considered a false negative.

From this information, multiple different performance metrics can be calculated for a given model:

- Precision (P): proportion of predicted boxes that are correct (contain an urchin). This measure represents how correct the predictions are (e.g. High P means not many false positives)
- Recall (R): proportion of ground truth boxes that are correctly detected. This metric represents how well the model detects urchins (e.g. High R means not many false negatives)
- F1 score: harmonic mean of precision and recall. Typically, there is a trade-off between P and R, although they should ideally both be high. The F1 score represents a balance between P and R.
- mAP: The average precision when varying the confidence cutoff of predictions. This is a broader measure of how the model is performing overall.
- mAP50:95: Calculate mAP using a range of IOU thresholds (from 0.5 to 0.95) to determine correctness. These mAP values are then averaged. This value emphasises how accurately the predicted bounding boxes match the ground truth.

Yolov5

Yolov5 [6] is a widely used and popular open-source implementation of the Yolo (you only look once) object detection architecture. All models trained and used in this urchin detector project were Yolov5 models. This architecture was chosen due to its very fast prediction speed (which is essential when large amounts of images need to be analysed) and past research's success utilising yolov5 for zoobenthos detection [4], [5].

The output of a yolov5 model, after making a prediction on a given image, is a set of bounding boxes (one for each detected urchin), each with a species label (*E. Chloroticus* or *C. Eodgersii*) and a confidence score (ranging from 0 to 1). This confidence score represents the estimated probability that the predicted box contains an urchin.

Yolov5 models have two important parameters independent of the trained model (i.e., they can be changed as needed) that alter their final predictions. These parameters are called the confidence and NMS IOU threshold, both ranging between 0 and 1. Yolov5 uses two processes to filter potentially incorrect predictions before returning the final results. The first process removes any predicted boxes with confidence below the confidence threshold. Raising the confidence threshold will increase the correctness of urchin predictions (higher precision) but will also cause it to miss more urchins (lower recall). Lowering the confidence threshold has the opposite effect. This parameter can, therefore, be altered before predictions based on what is more important for the specific application. The second process is called non-maximal suppression and is used to remove extra boxes that may be detecting

the same urchin. The NMS IOU threshold can be altered if issues arise with multiple predictions of the same urchin.

Techniques that improve performance

Throughout the development of the urchin detector, we have experimented with many techniques, hyperparameter settings, and modifications to increase detection performance. The following techniques had a positive impact on performance and are used in the final model.

Image inference resolution:
Before an image is processed through the model, it is resized to decrease computation time and ensure consistency across differently-sized images. By default, images are resized such that their longest side becomes 640 pixels long, and the short side is adjusted accordingly to maintain the aspect ratio. The downside is that details and essential features for detection (e.g. an urchin's spines) can be lost. This is especially detrimental for the detention of smaller urchins or urchins further away from the camera (both of which are very common). Instead, we resize images to be 1280 pixels on the long side. This increase in resolution greatly helps with detecting smaller urchins and improves performance, but it comes at the cost of significantly higher training time and memory usage.

Larger architecture:
Yolov5 offers a variety of network sizes for different applications. Larger networks can model more complex situations and typically perform better than smaller architectures. However, they also increase both the training and prediction time. Furthermore, although more complex architectures can perform better, they are also more prone to a phenomenon called overfitting, where the model effectively memorises the images seen in training and loses the ability to generalise to new situations. One of the main ways to deal with overfitting is to use more data during training. At the beginning of this project, due to a smaller dataset (approximately 10,000 annotations), we could only train models of the small architecture size. However, throughout the project, as more annotations were added and by using several other techniques (mentioned below) to reduce overfitting, we could utilise the medium architecture size. This led to better performance all round.

Increased data augmentation:
Data augmentation is another technique used to reduce overfitting that involves slightly altering the training images before the model sees them. These alterations included all the default Yolov5 augmentations (hue, saturation, value, rotation, translation, scale, flipping up/down and left/right, mosaic and mixup) at an increased rate, aswell as a Gaussian blur augmentation as seen in [7]. The blur augmentation was added because blurring is a prevalent issue with underwater image degradation.

Reduced objectness loss:
Loss is the model's measure of how poorly it performs for a given image. Yolov5 object detection loss comprises three components: box loss (a measure of the difference between the target and predicted box), class loss (difference between the target and predicted urchin species) and objectness loss. The objectness score is another name for the confidence score the model outputs alongside each box prediction. This measures the difference between the target confidence (1 if the box contains an urchin, 0 otherwise) and the

predicted confidence score. It is a commonly reported issue that this loss tends to overfit and start increasing early. To counter this, when the total objectness loss is calculated, it is decreased by a constant factor. This helped to delay the onset of overfitting.

Increased weight decay:
Weight decay is a regularisation technique that adds a penalty term to the loss based on the sum of the squares of the network parameters. This helps to reduce overfitting by encouraging the model to use smaller parameter values, creating a more stable, less complex model. Increasing this penalty puts more pressure on the network to adopt smaller weights, leading to even less overfitting.

Techniques that did not improve performance

In addition to the techniques that helped improve the final model's performance, there were several other techniques we experimented with during development that either had no effect or decreased performance.

Test time augmentation:
Test time augmentation (TTA) involves creating multiple slightly altered copies of an image, generating predictions for each copy, and then combining them to create the predictions for the original image. Typically, this increases performance, but in our case, it had a negligible effect but dramatically reduced prediction speed.

Underwater image enhancement:
Underwater image enhancement (UIE) is the name given to a class of algorithms that attempt to restore degraded underwater images. [5] utilised a UIE approach called relative global histogram stretching (RGHS) [8] as a preprocessing step which greatly improved performance, especially with sea urchin detection. When we implemented RGHS, we found that, in some cases, it could detect previously missed urchins, especially in very blue/green degraded images.

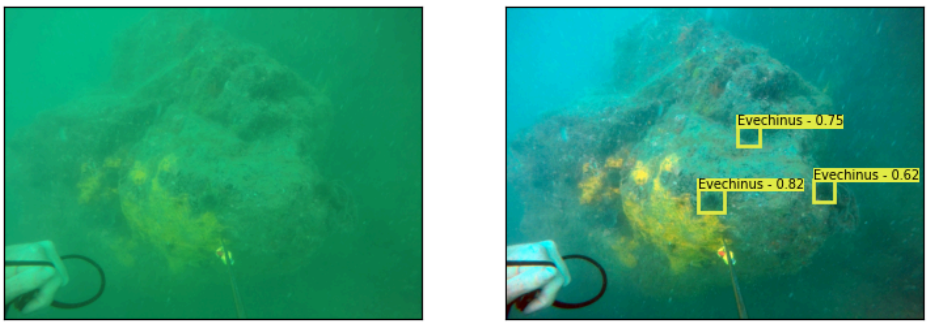

Fig. 2. Incorrect predictions of the unenhanced image (left) versus three correct predictions on the enhanced image (right).

However, in many other cases where the images were already relatively clear, had artificial lighting, or other types of degradation, RGHS significantly worsened image quality and caused the model to miss urchins it could previously detect. This is because RGHS is tailored to a specific kind of water colour and turbidity, meaning it does not perform well in

our dataset, which has a large variety of underwater scenes. Overall, this led to a slight decrease in performance.

Results

Precision, recall, F1 score, mAP and mAP50:95 were calculated on both the validation and test set and the results are displayed below.

TABLE I
VALIDATION SET RESULTS

Class	P	R	F1	mAP	mAP50:95
E. Chloroticus	0.883	0.900	0.891	0.931	0.505
C. Rodgersii	0.856	0.843	0.849	0.880	0.439
Both	0.869	0.872	0.870	0.906	0.472

Metrics calculated at conf = 0.45 (optimal conf determined on validation set).

TABLE II
TEST SET RESULTS

Class	P	R	F1	mAP	mAP50:95
E. Chloroticus	0.884	0.906	0.895	0.927	0.565
C. Rodgersii	0.848	0.823	0.835	0.861	0.460
Both	0.866	0.865	0.865	0.894	0.513

Metrics calculated at conf = 0.45 (optimal conf determined on validation set).

Overall, the validation and test set metrics are relatively similar, except for *C. Rodgersii* in the test set, which has metrics that are a few per cent lower across the board compared to the validation set. It is also important to note that the model never incorrectly predicts the species of a detected urchin, i.e. the model can perfectly differentiate between *E. Chloroticus* and *C. Rodgersii*.

There are also a few other notable metrics. The first is the proportion of perfect detections. These are images where all urchins are detected correctly, and no false positives exist. On the validation set, this is 53.2%, and on the test set, it is 53.3%. Another metric is the proportion of at least one correct prediction. These are images where at least one urchin is correctly detected (or in the case of images with no urchins, then no urchins are detected). On the validation set, this is 94.4%, and on the test set, it is 95.1%.

The model's performance was further investigated by grouping images by urchin count and calculating metrics on the images in those groups. As expected, the perfect detection rate steadily decreases as the number of urchins in an image increases. The rate of at least one correct prediction increases steeply as the count increases, reaching 100% when an image contains five or more urchins. All the other metrics (P, R, mAP, etc) also increase alongside the number of urchins. The stats for the images in the one to four urchin group (most images in the dataset are in this group) are particularly low, with precision and recall less than 0.8.

Discussion

Overall, the urchin detector manages to correctly detect and locate the majority of urchins in the dataset. One interesting note is the difference in performance between the two species. *E. Chloroticus's* metrics are much higher than *C. Rodgersii's* in the validation and test set. This is particularly interesting because there are almost twice as many *C. Rodgersii* annotations in the dataset, and typically, with more data to learn from, the model would perform better on that species, but this is not what we observe. The reason for this disparity may be because *C. Rodgersii* is just harder to detect, although it may also be related to the IOU threshold for determining if a prediction is correct.

By visually inspecting the errors the model makes, we can see that there is a relatively common case where the model does detect an urchin, but due to differences in box shape and size with the target, it is marked as an incorrect prediction. One possible cause for this issue is slight inconsistencies in annotation sizing around urchins. The model box predictions typically attempt to contain all the visible parts of an urchin, with no gap between the urchin and the box edges. However, in some cases, annotations from the dataset may be slightly too small and cut off parts of the urchin or too large, which leaves a lot of space between the urchin and the box or some other variation. These differences can cause a prediction to be counted as incorrect by reducing the box overlap (IOU) to below the default threshold of 0.5. Based on visual inspection, these issues are more common with *C. Rodgersii*. We can recalculate the metrics at a lower threshold, such as 0.3, to investigate this further.

TABLE III
VALIDATION SET RESULTS AT LOWER IOU

Class	P	R	F1	mAP
E. Chloroticus	0.900	0.920	0.910	0.957
C. Rodgersii	0.902	0.894	0.898	0.941
Both	0.901	0.907	0.904	0.949

Metrics calculated at conf = 0.42 (optimal conf determined on validation set) and IOU = 0.3 (as opposed to the standard 0.5).

TABLE IV
TEST SET RESULTS AT LOWER IOU

Class	P	R	F1	mAP
E. Chloroticus	0.888	0.898	0.893	0.938
C. Rodgersii	0.887	0.918	0.902	0.914
Both	0.887	0.908	0.897	0.926

Metrics calculated at conf = 0.42 (optimal conf determined on validation set) and IOU = 0.3 (as opposed to the standard 0.5).

In addition to these results indicating better performance overall, we can also see that the performance across species becomes much more similar, indicating that lowering the IOU threshold has a greater impact on the *C. Rodgersii* results. One thing to consider when lowering the IOU threshold is that it can allow inaccurate (off-target) predictions to be

considered correct. However, given that the intended use of the urchin detector is to analyse sets of images at larger scales, slight inaccuracies at the single image level are less important; what matters more is understanding how well the model detects urchins, whether it localises accurately is less important. It is also possible that lowering the IOU threshold would allow false negative predictions near actual urchins (that were otherwise undetected) to be counted as correct. This situation, however, seems to be very rare, so this issue is likely to be insignificant.

Areas the model fails

By visually comparing model predictions to the annotations from the dataset, we can gather insights into what can cause the AI to miss an urchin or detect an urchin that is not there.

Image quality

One of the model's most common errors is missing urchins in blurry images (e.g. motion blur from towed video) or images suffering from underwater image degradation. Both of these factors can obscure the details of urchins and make them blend in with the background, making them very challenging to detect. Checking characteristics of an image (such as blurriness) can be used as an early warning to suggest that the model may perform particularly poorly.

Obscured urchins

In many cases, urchins are partially obscured, and only a portion of the urchin is visible. This most often occurs when an urchin is hidden in a crevasse or between rocks but also occurs when an urchin is on the edge of the image or some other obstruction (e.g. camera pole). When this happens, the chance that an urchin is detected decreases. If a significant portion of the urchin is hidden (e.g. only the tips of the spines are visible), it is very unlikely that the urchin will be detected. Furthermore, the model may detect each half separately if an urchin is bisected by another object (e.g., a piece of seaweed). Occasionally, the model will also make false positive predictions along dark areas, such as crevasses.

Urchins missed in the dataset

Sometimes the model will detect an urchin that was missed by the annotators when the dataset was created. Because these urchins do not have a corresponding label in the dataset, they are counted as incorrect, causing the metrics to decrease slightly. However, the number of urchins like this is relatively small, so it is unlikely to have a huge impact on the results.

Annotations that may not contain urchins

Some annotations in the dataset may not contain an urchin and are very difficult to distinguish from the background. Many of these boxes are not flagged for review or have a likelihood parameter of one, meaning they cannot be filtered out. The model almost always misses these kinds of boxes.

Large groups of urchins

When large groups of urchins are in an image, the model can fail in several ways. If two or more urchins overlap, they are often grouped and detected as one large urchin. Furthermore, in images with lots of urchins, there is a higher rate of urchins missed by the annotators, and a higher rate of off-center annotations.

Conclusion

Through this project, we have developed an urchin detector AI capable of correctly detecting the majority of urchins in an image and perfectly classifying their species. This model is ready to be deployed on Squidle to aid and assist with further sea urchin detection. We have also identified and discussed the errors the model makes and where they are likely to occur. Although the model achieves reasonable results, it could be improved in several ways with further development and research.

Many of the images in our dataset are video frames. Given that if an urchin appears in one frame, it is likely to occur in the next consecutive frame, it could be possible to leverage this and do some analysis on a run of consecutive video frames to improve performance. Video frames could be grouped using some combination of time, longitude or latitude metadata, all available in the dataset.

As mentioned earlier, larger model architectures often improve performance if overfitting can be avoided. As the dataset used in this project is still being worked on and more labels are being added, it may be possible to train larger models in the future. Furthermore, due to the significant time it takes to train models of a medium or larger size, only a limited amount of hyperparameter tuning has been performed. With more time, these parameters could be further optimised, which could directly lead to better performance or further reduce overfitting.

Given that we have a reasonable understanding of when the model is likely to fail, it may be possible to develop some kind of detection system. This would allow images we think the model may perform poorly with to be flagged and checked by a human expert. This technique is known as selective classification or machine learning with a reject option [9] and has been explored extensively with classification tasks, but there is little to no research on selective object detection.

Lastly, as mentioned in the areas the model fails section, there are a number of urchins in images in the dataset that have been missed, and there may also be some urchin annotations that do not contain an urchin. It is very difficult to determine the extent of this problem as there is no systematic or automatic way to locate these, as beyond human perception, they are indistinguishable from genuine false positives and false negatives. One way to ameliorate this would be to verify and refine the dataset, although this would be very time-consuming and may not be practical to do with all images.

References

[1] C. Beyan and H. I. Browman, “Setting the stage for the machine intelligence era in marine science,” *ICES Journal of Marine Science*, vol. 77, no. 4, pp. 1267–1273, Jul. 2020, doi: 10.1093/icesjms/fsaa084.

[2] Z. Zhou *et al.*, “EchoAI: A deep-learning based model for classification of echinoderms in global oceans,” *Front. Mar. Sci.*, vol. 10, p. 1147690, Apr. 2023, doi: 10.3389/fmars.2023.1147690.

[3] A. Mahmood *et al.*, “Automatic detection of Western rock lobster using synthetic data,” *ICES Journal of Marine Science*, vol. 77, no. 4, pp. 1308–1317, Jul. 2020, doi: 10.1093/icesjms/fsz223.

[4] L. Zhang *et al.*, “Marine zoobenthos recognition algorithm based on improved lightweight YOLOv5,” *Ecological Informatics*, vol. 80, p. 102467, May 2024, doi: 10.1016/j.ecoinf.2024.102467.

[5] Y. Li, X. Bai, and C. Xia, “An Improved YOLOV5 Based on Triplet Attention and Prediction Head Optimization for Marine Organism Detection on Underwater Mobile Platforms,” *JMSE*, vol. 10, no. 9, p. 1230, Sep. 2022, doi: 10.3390/jmse10091230.

[6] G. Jocher. *YOLOv5 by Ultralytics*. (2020). Ultralytics. Accessed: December 11, 2023. [online]. doi: 10.5281/zenodo.3908559. Available: https://github.com/ultralytics/yolov5

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A Simple Framework for Contrastive Learning of Visual Representations,” 2020, doi: 10.48550/ARXIV.2002.05709.

[8] D. Huang, Y. Wang, W. Song, J. Sequeira, and S. Mavromatis, “Shallow-Water Image Enhancement Using Relative Global Histogram Stretching Based on Adaptive Parameter Acquisition,” in *MultiMedia Modeling*, vol. 10704, K. Schoeffmann, T. H. Chalidabhongse, C. W. Ngo, S. Aramvith, N. E. O’Connor, Y.-S. Ho, M. Gabbouj, and A. Elgammal, Eds., in Lecture Notes in Computer Science, vol. 10704. , Cham: Springer International Publishing, 2018, pp. 453–465. doi: 10.1007/978-3-319-73603-7_37.

[9] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis, “Machine Learning with a Reject Option: A survey,” 2021, doi: 10.48550/ARXIV.2107.11277.

Appendices

Github project link: https://github.com/kraw084/Urchin-Detector