Auto-detection of southern sea otters (Enhydra lutris nereis) from aerial imaging on the Monterey Peninsula

Margaret Daly, Carlota Parés-Morlans, Mohammed Salman
Department of Computer Science
Stanford University
mad297@stanford.edu, cpares@stanford.edu, salmanmo@stanford.edu

Abstract

The Southern sea otter (Enhydra lutris nereis) is a keystone predator and a protected marine mammal that inhabits the California coast and beyond. Current methods to study otters are expensive, inaccurate, and inefficient. To improve the use of resources for government officials, ecologists, and other researchers, we are proposing an auto-detection algorithm using drone aerial imagery as inputs. Using 842 images of sea otters from Monterey Bay and 1018 background images extracted from online datasets, we labeled, clustered, and augmented images to train a modified version of YOLOv5. The resulting model with a 75% F1-score and 26% $mAP_{0.5:0.95}$ outperforms human performance as it detects otters that remained undetected to humans in the labeling process and were later identified.

1 Introduction

The Southern sea otter (Enhydra lutris nereis) is a marine apex predator that forages in coastal waters throughout the Northern Pacific. As a keystone predator, sea otter populations and their foraging behaviors can have a significant impact on the biodiversity and overall health of kelp forest ecosystems [1]. The Southern sea otter is listed as "threatened" under the Endangered Species Act. Since 1982, the California Department of Fish and Wildlife, the U.S. Geological Survey, U.S. Fish and Wildlife, and the Monterey Bay Aquarium monitor the sea otter population with an aerial survey (plane or helicopter) in the spring and fall every year.

This surveying task is expensive, time-consuming, and yields 1-2 values for sea otter population sizes per year. Given the technological advances in commercial aerial imaging and machine learning statistical algorithms, current monitoring protocols may potentially be replaced by these technologies someday. The objective of this project is to develop an algorithm that can successfully detect sea otters in aerial imaging. The data has been directly coordinated with U.S. Fish and Wildlife, California Fish and Wildlife, and Sea Otter Savvy with applications in policy, marine science, and oil spill reconnaissance procedures.

Unlike in other natural images, sea otters in UAV RGB imagery are very small, with high variability in acquisition lighting and weather conditions. Moreover, datasets are usually small due to sampling constraints. These conditions pose a major challenge in terms of both precision and detection speed. In response to this challenge, and based on the ideas proposed in Pham et al.[2] and Liu et al.[3], we modified the YOLOv5 model [4] to improve its performance on small object detection. Using RGB images as inputs and obtaining bounding boxes around the sea otters as outputs, the modified YOLOv5 improved the original YOLOv5 model F1 and $mAP_{0.5:0.95}$ scores by 7.8%, and 3%, respectively.

2 Related Work

The field of computer vision has been extensively explored over the past years, primarily as a consequence of the great advances in deep learning. Object detection is one of the most popular tasks due to the wide range of applications and improvements in recent years. Broadly, we can divide object detection tasks into 2 categories: two stage detectors, such as Fast R-CNN [5], Faster R-CNN [6], R-FCN [7], Feature Pyramid Network [8], Mask R-CNN [9], etc., and single stage detectors, such as YOLO [10] and all its improved versions (YOLOv2, YOLOv3, YOLOv4, and YOLOv5), EfficientDet [11],etc. In general, two-stage detectors favor detection accuracy while one-stage models are faster [2].

UAV object detection has been an active research topic over the past decade. Several researchers have proposed models that focus on small object detection [2], [3]. Pham et al. [2] added several modifications to the YOLOv3 model, including the removal of two coarse detection levels and replacing them with two finer detection levels as well as the removal of some layers from the backbone. Their experiments on multiple datasets show a $\sim 4.5\%$ mAP score increase on average. Liu et al. [3] optimized the Resblock by concatenating two ResNet units that have the same width and height. Moreover, the authors increased the convolution operation at an early layer to enrich spatial information. Their experimental results show that *UAV_YOLO* improved YOLOv3 mAP results by $\sim 3\%$.

In marine science, there have been a few applications such as the work done by Gray et al.[12], where they used a CNN to detect sea turtles in drone imagery. To increase the sampling size, Gray et al. cropped the images around each labeled sea turtle. While they have a 76.5% recall, their model's precision hovers at 16.3%. Researchers have also monitored other marine megafauna such as sharks [13], whales [14], and birds [15]. Sharma et al.[13] proposed method obtained a 90.04% precision, however, the authors did not disclose any other metrics, and the number of epochs ranged from 60,000 - 70,000. Even if these results are very promising, UAV marine animal detection remains an open challenge.

3 Dataset

3.1 Raw Data

During the months of July-October of 2020, aerial images (n = 842) with a resolution of 5472 x 3648 pixels were collected, showing sea otter presence. The distance from sea level ranges from 20-100m at 4 locations in Monterey Bay. Images were also recorded in various weather conditions. This metadata is included in the filenames for future purposes. Given this dataset, we employed a train-validation-test split of 84%-8%-8% on the 842 total images (see Table 1).

Dataset	#Images	#Otters	#Imgs w/ Ott.	#Ott. / Img
train	702	5507	702	7.845
validation	70	609	69	8.7
test	70	554	70	7.914

Table 1: Mavic2Pro DJI Drone Sea-Otter Imagery Dataset

3.2 Resizing

All images were originally 4K DNG files. Files were converted to JPG format and downscaled to 1024 x 682. We chose this resolution to allow our model to train faster but still keep enough resolution for the otters since some of them are a few pixels in size. Sea otters tend to be at the center of the image, and their size tends to be between 1% and 2%of the image size.

3.3 Labeling

To train our YOLO model we manually labeled all the images using the software LabelImg [16]. It took ~ 30 hours in total and was done by a single researcher so that images were labeled with the same subjectivity. Other classes such as kayak, seal, boat, and bird were labeled as well for future purposes, given their little occurrence in the dataset.

3.4 Image Clustering

Some images were taken in succession while the drone was flying over the sea otters. Hence, successive images are highly similar with the otters at different positions since some of them were swimming and others went underwater. This caveat could potentially lead to overfitting if two successive images were split between the training set and the test set. Inspired by Flomo[17], we used a pretrained VGG model and modified the last layer to get a feature of size 4,096 that represents each image. Then, we downsized this feature vector to 50 using Principal Component Analysis and finally applied k-means to find the image clusters. We used the elbow method to find the optimal number of clusters, which turned out to be 50. From these clusters, we selected 42 clusters for the training set, 4 different clusters for the validation set, and the remaining 4 clusters for the test set.

3.5 Background Images

The primary focus of the sea otter study that provided us with the dataset was to understand behavioral changes in sea otters in the presence of commercial drones. Consequently, most images include sea otters, and there are no background images, which are treated as negative samples in the dataset. Open source background images from an oyster reef survey [18] and an Arctic cetacean study [14] were collected to supplement the original dataset, adding 1,018 images.

3.6 Image Augmentation

Given the small size of our dataset and to improve our model performance, we employed several data augmentation techniques. These included Hue-Saturation-Value (HSV) augmentations, scaling, rotation, and vertical/horizontal flip. Image shearing does not occur due to the experimental design of image capturing, so we neglected translation and shearing augmentations. Additionally, we also applied mosaic data augmentation, a data augmentation technique available on YOLOv5[4], which combines random ratios of image tiles into a single image.

4 Methods

4.1 Proposed Model

Inspired by the existing literature on small object detection presented in section 2, we modified the original YOLOv5 model to include a detection level for small objects. Moreover, we removed the detection level for big objects since all objects are small in size in our study. In Figure 1, we can see highlighted in gray the layers that we added to the original model.

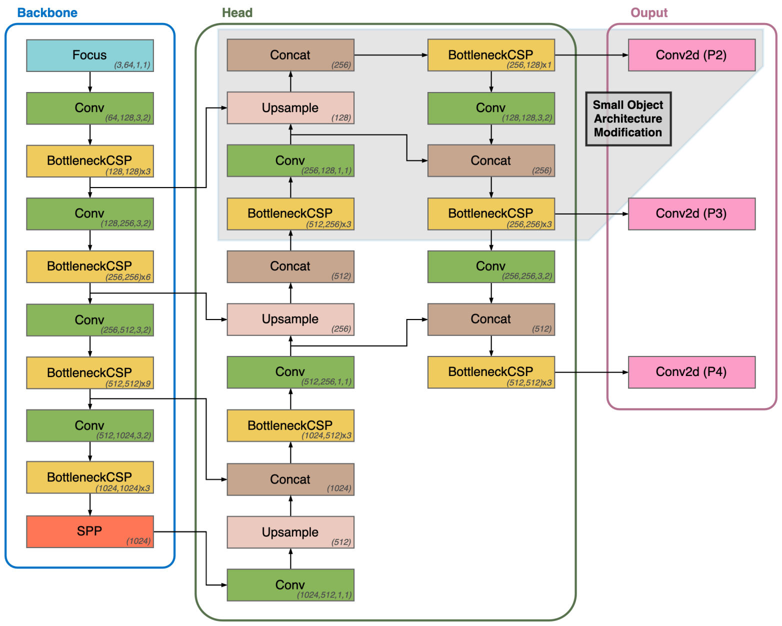

Figure 1: Proposed YOLOv5 model for small object detection, modifications are grey.

4.1.1 Loss function

In this work, we have used the original loss function from YOLO [19] (see equation 1). This loss function can be broken into three parts: (i) bounding box coordinates and size prediction, (ii) confidence score, (iii) and class prediction. All three parts are Mean-Squared error losses, which are modulated by lambda parameters. Two indicator functions are used to (1) penalize classification if the object is present in the grid cell and (2) penalize bounding box coordinates if that box has the highest IoU. One important thing to note from the function is that width and height parameters are under a square root. This helps penalize smaller bounding boxes, which is necessary in this project.

$$
\begin{aligned}
\mathcal{L}(\theta, \hat{\theta}) &= \overbrace{\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_i^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_i^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]}^{\text{(i)}} \\
&\overbrace{+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2}^{\text{(ii)}} \\
&\overbrace{+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \,\in\, \text{classes}} (p_i(c) - \hat{p}_i(c))^2)}^{\text{(iii)}}
\end{aligned} \tag{1}
$$

4.1.2 Anchors

Given our proposed model, it has been necessary to compute anchors sizes that fit the new detection levels. To this end, we used the autoanchor function included in YOLOv5[4], which computes the most optimal anchors for the given dataset using a genetic algorithm.

5 Results

5.1 Evaluation Metrics

Given the purpose of this project, we have chosen F1-score and mAP-score as our main performance indicators. With these two metrics, we are assuring that we detect as many otters as possible while not overly estimating the sea otter population and keeping bounding boxes close to ground truth. From the two metrics, we have chosen mAP as our main performance indicator to be able to compare our results to those from the research community, which has adopted mAP as the main metric.

- F1: Harmonic mean of precision and recall. $\mathrm{F1} = 2(\mathrm{P} \cdot \mathrm{R})/(\mathrm{P} + \mathrm{R})$.

  Precision (P): ratio of true positives and all positive predictions.
  Recall (R): ratio of true positives and all positive ground truth.

- mAP (mean Average Precision): Mean of the area under the precision recall curve.

It is important to note that for determining whether a predicted outcome is a true positive or a false positive, the Intersection over Union (IoU) ratio is used. Specifically, the IoU ratio is the amount of overlap between the predicted bounding box and the ground reference bounding box.

5.2 Hyperparameters

For the experiments presented in the following section, we used YOLOv5 default hyperparameters. The default values for some of the most important hyperparameters are:

- Initial learning rate: 0.01
- Final learning rate: 0.1 (1 cycle learning rate policy is applied. The learning rate changes after every batch).
- SGD momentum: 0.937
- IoU training threshold: 0.2

These parameters were later fine-tuned using the validation set (see section 5.4).

5.3 Experiments

As noted extensively in class, the process we have followed has been iterative. The first experiment that we did was running YOLOv5 with default hyperparameters for 300 epochs and a batch size of 32 to get a baseline performance. We chose a batch size of 32 since it is the maximum value we can use without going out of memory, while we still benefit from a reduced training time. Besides, we had to train with a maximum image size of 768px due to memory limits. Given that the validation loss steadily decreased for 300 epochs, we decided to run the model for a higher number of epochs to observe how many epochs are required before overfitting occurs. After running the baseline model for 500 epochs with no observations of overfitting, we performed error analysis and discovered that nearly identical images were being included in the training set as well as in the validation set. This issue led us to cluster our images (see section 3.4 for more information).

After the previous step, we run the vanilla model again and performed error analysis on it. The main problems we detected and the solutions we found are presented in Table 2.

	Detected Problem	**Proposed Solution**
1	Sunlight and glint cause false positives	Include background images
2	Farther distances cause false negatives	Modify YOLOv5 to include a detection level for small objects

Table 2: Main problems detected using error analysis.

As we can seen in Table 2, one of the problems we detected was that images with substantial sunlight and glint had more false positives. To solve it, we gathered ocean images from online datasets to add as background images to our training set (see section 3.5 for more information). From Table 3, we can see that with the background images F1-score improved 1.8%.

After the implementation of the previously mentioned solution, we modified YOLOv5 architecture to include a detection level for small objects (see section 4.1 for more information). As seen in Table 3, this modification yielded to a increase of 8.2% and 2.8% in the F1 and $\mathrm{mAP}_{0.5:0.95}$ scores, respectively.

5.4 Hyperparameter Tuning

During the training of model 3, we recorded the validation loss for every epoch. With this information, we were able to detect that the model was overfitting when running it for 500 epochs. We observed that the validation loss decreased until 200 epochs and subsequently started increasing. Therefore, we trained it again for only 200 epochs.

Given that YOLOv5 has over 30 hyperparameters used for various training settings, we decided to use the provided genetic algorithm that searches for the best hyperparameters [4]. Specifically, using the weights from the best model as the initial weights, we performed 100 evolutions of 10 epochs each. We modified the weights of the fitness function to also consider precision and recall ($[\mathrm{P} = 0.1, \mathrm{R} = 0.1, \mathrm{mAP@0.5} = 0.1, \mathrm{mAP@0.5} : 0.95 = 0.7]$).The resulting hyperparameters were used to retrain the best model for an extra 100 epochs.

5.5 Model Results

	Training			Test		
Experiments	F1	mAP@0.5	mAP@0.5-0.95	F1	mAP@0.5	mAP@0.5-0.95
1. vanilla clusters	0.601	0.584	0.21	0.675	0.667	0.229
2. clust. + background	0.609	0.578	0.227	0.693	0.695	0.23
3. clust. + background + modified_arch	0.666	0.662	0.2505	**0.757**	0.748	0.257
4. clust. + background + modified_arch + param. tuning ★	**0.761**	**0.76**	**0.265**	0.753	**0.756**	**0.259**

Table 3: Results from the different experiments.

Table 3 includes an overview of the performance of each of the presented experiments. Moreover, in Figure 10, we can find the evolution of these performance metrics for the best model (model 4).

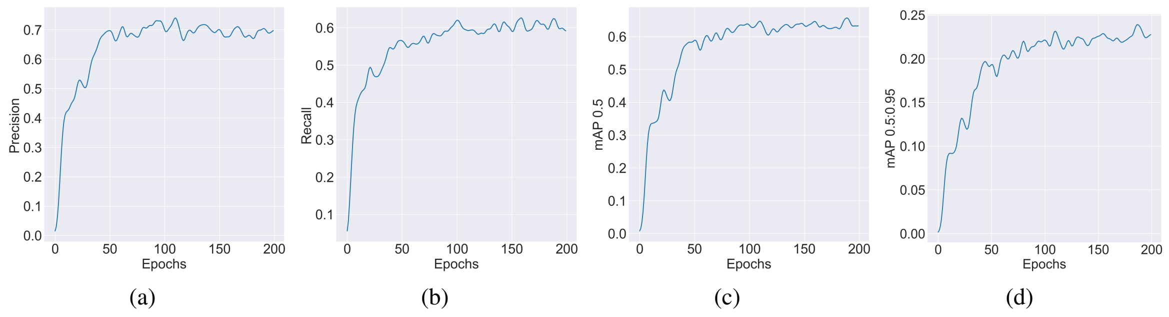

Figure 2: Performance results from the best model. (a) Precision (b) Recall (c) mAP 0.5 (d) mAP 0.5:0.95

6 Discussion

Looking at the results presented in the previous section, it is clear that we have been able to improve the model performance on every iteration. As it has been mentioned before, error analysis has been key to find the weak points of every model and define possible solutions that may improve its performance. Given the small dataset size and the variability of the images (height of drone and environmental factors), we believe our results are very promising as even when reviewing the predictions from the test set, the model found otters the labeler missed, meaning that it's currently more accurate then a human and will only improve once these are corrected.

7 Conclusions and Future Work

In this work, we have presented a modified version of the YOLOv5 model specifically crafted for small object detection in an oceanic environment. From a baseline YOLOv5 model, we have iterated to find a model with good performance metrics. With the different modifications we have introduced, we have obtained successful results. However, we think that the characteristics of our dataset hinder the performance of the presented model. If a bigger and more diverse dataset was to be collected, further experiments could be carried out. As future work, a two-stage detector could also be explored, even though its speed might not be suitable for certain applications.

As it has been mentioned in the beginning, other classes were labeled in addition to sea otters. Future work of interest would include the detection of these classes if the number of instances per class was to become sufficient in a bigger dataset. Besides, from the labeled images multiple information could be extracted: size of the sea otter population in the areas of interest, sea otter behavioral analysis from a labeled video, proximity relationships, public policy for UAV heights above marine mammals, and at-risk animals in oil spills.

8 Contributions

Given the extension and the different requirements of this project, it has been very beneficial to have an interdisciplinary team. Daly provided the sea otter data as well as oceanographic knowledge to ensure methods and analysis were logical for its application. She also labeled the images and found the background images. Parés-Morlans explored the related work on small object detection and designed the new architecture. Moreover, she performed error analysis, clustering, and hyperparameter tuning. Together with Salman, they performed image augmentation and ran the experiments. Salman also explored tiling the dataset and did all the setup for the experiments in Google Colab and AWS.

9 Code Availability

The code is available at the following GitHub repository https://github.com/carlotapares/cs230_otters.

References

[1] J. Estes, E. Danner, D. Doak, B. Konar, A. Springer, P. Steinberg, M. T. Tinker, and T. Williams, "Complex trophic interactions in kelp forest ecosystems," *Bulletin of marine science*, vol. 74, no. 3, pp. 621–638, 2004.

[2] M.-T. Pham, L. Courtrai, C. Friguet, S. Lefèvre, and A. Baussard, "Yolo-fine: one-stage detector of small objects under various backgrounds in remote sensing images," *Remote Sensing*, vol. 12, no. 15, p. 2501, 2020.

[3] M. Liu, X. Wang, A. Zhou, X. Fu, Y. Ma, and C. Piao, "Uav-yolo: small object detection on unmanned aerial vehicle perspective," *Sensors*, vol. 20, no. 8, p. 2238, 2020.

[4] U. glenn jocher, "Yolov5." `https://github.com/ultralytics/yolov5`, 2020.

[5] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, pp. 91–99, 2015.

[7] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, pp. 379–387, 2016.

[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[11] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[12] P. C. Gray, A. Fleishman, D. J. Klein, M. W. McKown, V. S. Bézy, K. J. Lohmann, and D. W. Johnston, "A convolutional neural network for detecting sea turtles in drone imagery," in *British Ecological Society*, 2018.

[13] N. Sharma, P. Scully-Power, and M. Blumenstein, "Shark detection from aerial imagery using region-based cnn, a study," in *AI 2018: Advances in Artificial Intelligence*, 2018.

[14] P. C. Gray, K. C. Bierlich, S. A. Mantell, A. S. Friedlaender, J. A. Goldbogen, and D. W. Johnston, "Drones and convolutional neural networks facilitate automated and accurate cetacean species identification and photogrammetry," in *Methods in Ecology and Evolution*, 2019.

[15] Boudaoud, L. Ben, F. Maussang, R. Garello, and A. Chevallier, "Marine bird detection based on deep learning using high-resolution aerial images," in *OCEANS 2019-Marseille*, pp. 1–7, IEEE, 2019.

[16] Tzutalin, "Labelimg." `https://github.com/tzutalin/labelImg`, 2015.

[17] G. Flomo, "How to cluster images based on visual similarity." `https://towardsdatascience.com/how-to-cluster-images-based-on-visual-similarity-cd6e7209fe34`, 2020.

[18] J. T. Ridge, P. C. Gray, A. E. Windle, and D. W. Johnston, "Deep learning for coastal resource conservation: automating detection of shellfish reefs," *Remote Sensing in Ecology and Conservation*, vol. 6, no. 4, pp. 431–440, 2020.

[19] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

Appendices

9.1 Labeled examples

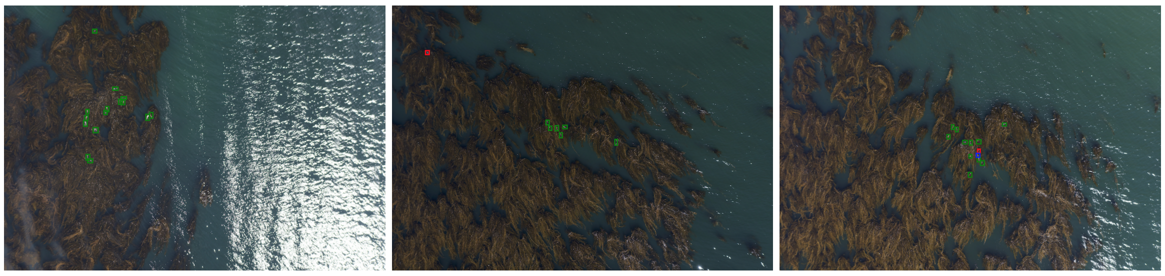

Figure 3: Labelled examples. Green indicates True Positive. Red indicates False Positive. Blue indicates False Negative. (a) Example with all labels correct (b) Example with a detected otter that remained undetected by a human while labeling (c) Example with errors.

9.2 Hyperparameter Tuning

In Figure 4, we can see the results of running the genetic algorithm for 100 generations. Each subplot shows fitness (y axis) vs hyperparameter values (x axis). Yellow indicates higher concentrations. Vertical distributions indicate that a parameter has been disabled and does not mutate.

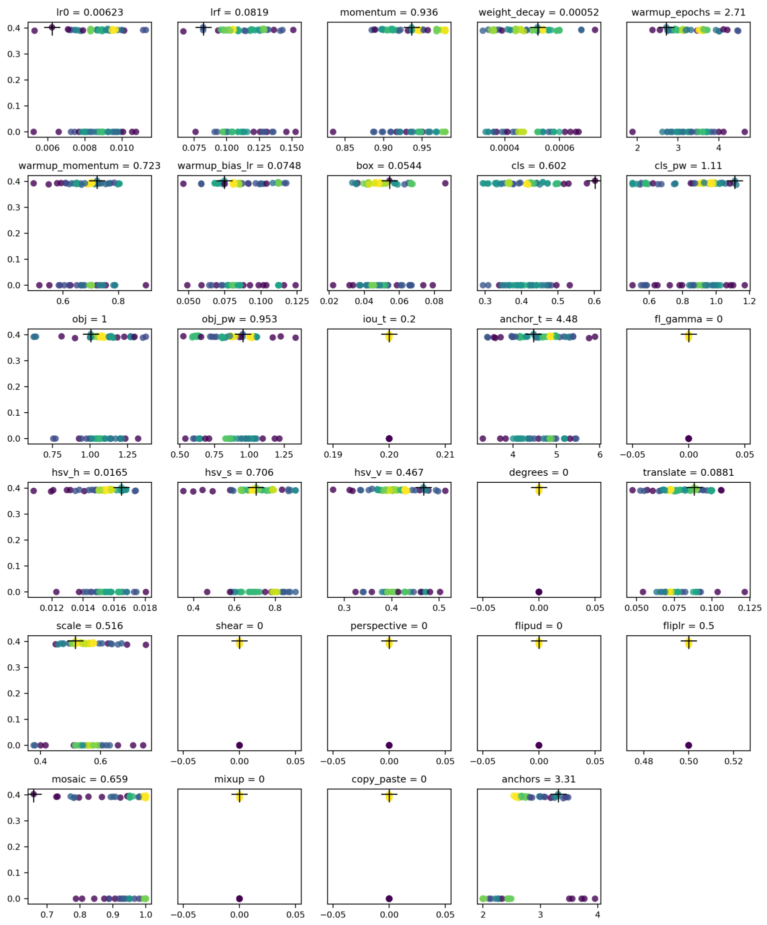

Figure 4: Hyperparameter value scatter plot from the different generations.

9.3 Sample Raw Images

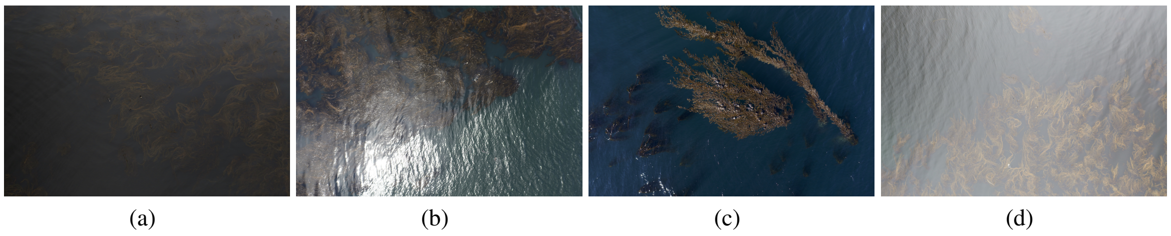
Figure 5: Sample images from the Mavic2Pro DJI Drone Imagery Dataset.

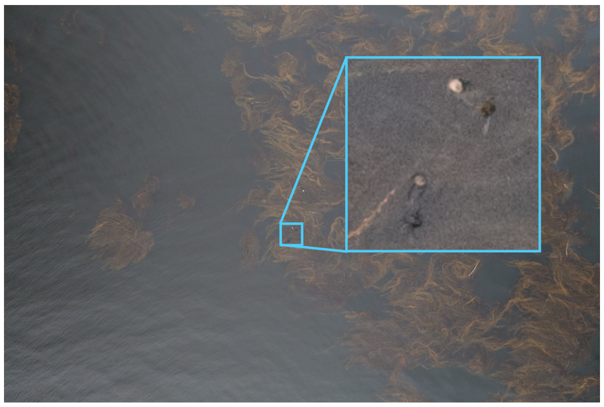
Figure 6: Example of large-scale image and sub-image.

9.4 Sample Background Images

Figure 7: Sample background images.

9.5 Sample Image Augmentation

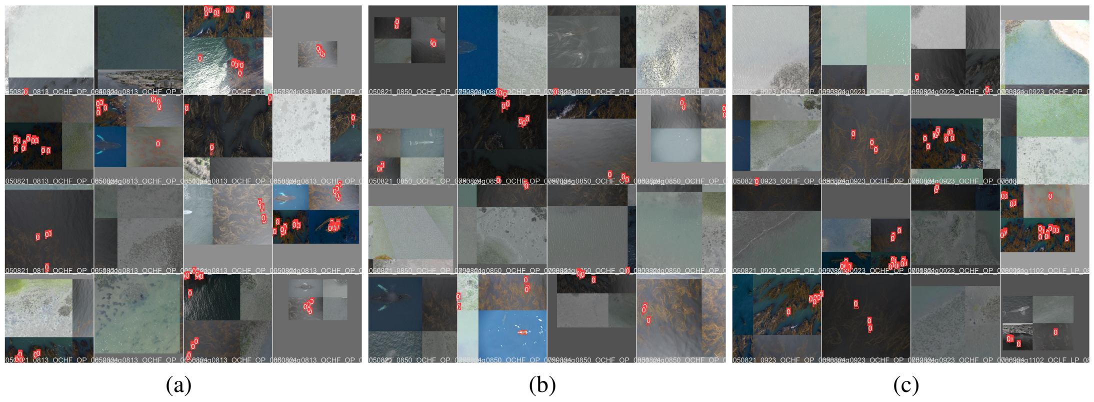
Figure 8: Sample Image Augmentation. Output generated by YOLOv5 using the user-defined hyperparameters.

9.6 Overfitting

As it can be seen in Figure 9, the validation loss started to increase after epoch 200. Consequently, we trained the model for 200 epochs to avoid overfitting and have a worse performance on the test set.

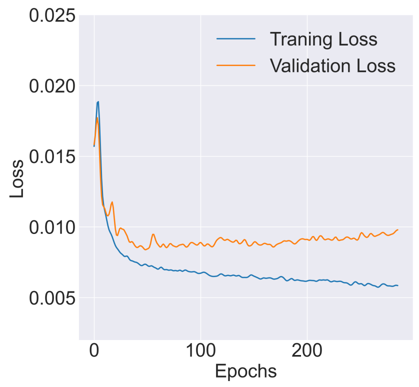

Figure 9: Training vs Validation Loss from model 3.

9.7 Tiling

As mentioned before, sea otters have a small size compared to image size. To increase the otter to ocean pixel ratio, we created a second dataset with the original images divided into 8 tiles with size 256x341 each. As the image was split into even amounts, the center of the image was split horizontally and vertically. Given that this is the area where the majority of otters reside, 50% of the otters got their body cropped. Thus, we didn't proceed further with this approach.

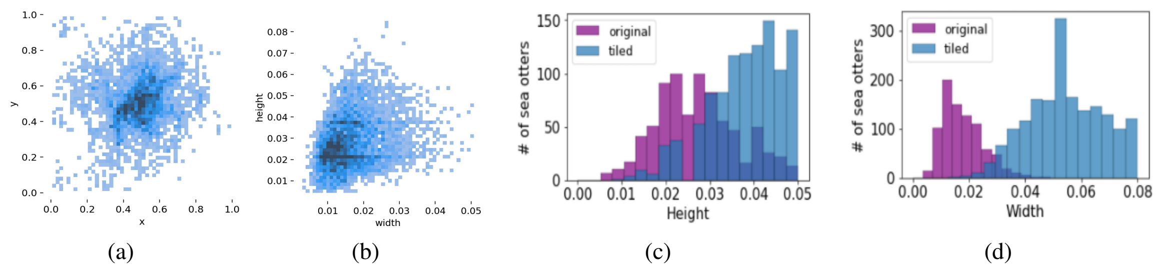

Figure 10: (a) Sea otter image position. (b) Sea otter size on image (c) Histogram of bounding box height (d) Histogram of bounding box width.

9.8 Precision-Recall curve

In Figure 11, we can see the precision-recall curve of model 4.

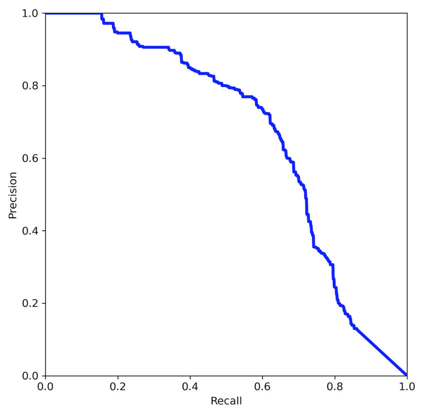

Figure 11: Precision-Recall curve