2022

Aquatic Species Detection and Classification using Deep Learning

CAPSTONE 3

RUSHIKESH BATTULWAR

SPRINGBOARD

1. Introduction

There are several types of marine species living in aquarium across the United States. Some of these species have found their new home at these aquariums because of pollution, overfishing, and habitat destruction result in population decrease, extinction, or replacement of species. Monitoring the frequency and abundance of these species is therefore necessary to inform aquarium authorities so that they can ensure healthy ecosystems and fish stocks. Moreover, the number and distribution of different fish species can provide useful information about ecosystem health and can be used for tracking environmental change. Techniques such as fish tagging, catch-and-release fishing, and video and image analysis can determine relative abundance and track population marine species. Sonar and video data combined with recent advances in machine learning provide a potential alternative to invasive fish tags for monitoring at aquatic life health. Visual classification of fishes can also aid in tracking their movements and revealing patterns and trends in their behavior, allowing for a more in-depth understanding of the species. However, there are numerous challenges including luminosity variation, fish camouflage, complex backgrounds, water murkiness, low resolution, shape deformations of swimming fish, and subtle variations between some marine species.
There have been many attempts to classify aqueous species using convolutional neural networks, often with image processing techniques to filter the initial images. Deep learning classification models require large amounts of labeled training data, that is images with bounding boxes or regional "masks" drawn around objects of interest with labels for the object type. Models trained on large general object datasets such as ImageNet for general classification tasks can be leveraged through transfer learning, the process of providing smaller amounts of domain specific labeled training data, to reduce the amount of data that must be captured and manually annotated by experts. Large-scale distributed computing resources and reusable analysis pipelines are important for training deep learning models. Fortunately, the resulting trained models can often be run in real-time on smaller embedded devices for continuous monitoring. The YOLO (you only look once) (Redmon et al., 2016) family of object detection models is designed to process video in real time and has been shown to reliably detect fish in noisy, low light, and hazy underwater images (Redmon et al., 2016; Sung et al., 2017; Jalal et al., 2020). YOLO models place a "bounding box" around detected objects.
In this project, deep neural network algorithm is applied to aquarium images to develop an aquatic species detection and classification system. The main aim of this project is to test the performance of a deep neural network model on aquarium images to detect and classify different marine species and deploy it into to a web app.

The EfficientDet D0 model from the Tensorflow model zoo has been tested for the detection and classification of 7 aquatic species. The detection and classification were tested on a public dataset consists of 637 images collected by Roboflow from two aquariums in the United States: The Henry Doorly Zoo in Omaha (October 16, 2020) and the National Aquarium in Baltimore (November 14, 2020). The custom trained model was deployed in the form of a web application using Steamlit API.

The contributions include:

- Applying deep learning models, EfficientDet D0 for marine detection and species classification, demonstrating that deep learning models can indeed detect and correctly classify species from high quality aquarium images,
- Custom training a deep learning model already trained on COCO dataset
- Implementing the trained model in form a web application

2. Dataset

To custom train the model, a public dataset available on the website of Roboflow was used. This dataset has a total of 637 images collected by Roboflow from two two aquariums in the United States: The Henry Doorly Zoo in Omaha (October 16, 2020) and the National Aquarium in Baltimore (November 14, 2020). The data in this dataset are stored in image format format, specifically .jpeg. The raw images have been used in the analysis. Each image is 768 x 1024 pixels size. There are 7 species available in the dataset: Fish, Jellyfish, Penguin, Shark, Puffin, Stingray, and Starfish. Fig. 1 shows the distribution of different species in the dataset. Note that the species distribution of this data is very unbalanced over 50% of samples with fish class and less than 5% belonging to stingray and starfish class.

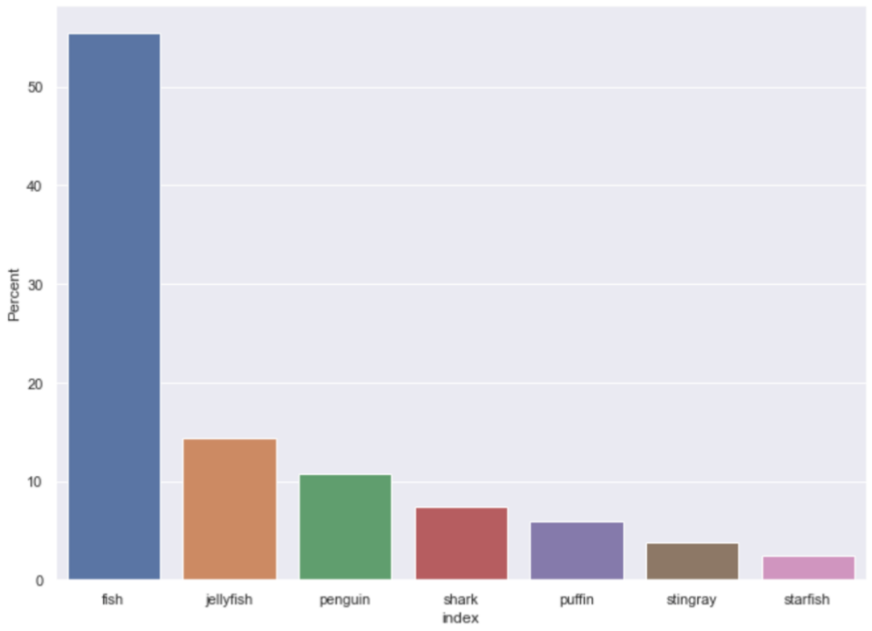


Figure 1 Distribution of different species in the dataset

The dataset is annotated by roboflow and includes spreadsheets with textual information about the filename, width, height, class, xmin, ymin, xmax, and ymax bounding box coordinates for each species present in the images. An example of the image with labels is shown in Fig. 2. As shown, there are two penguins in this image which are labelled and localized with the bounding box information. Images can have multiple different species present in them.

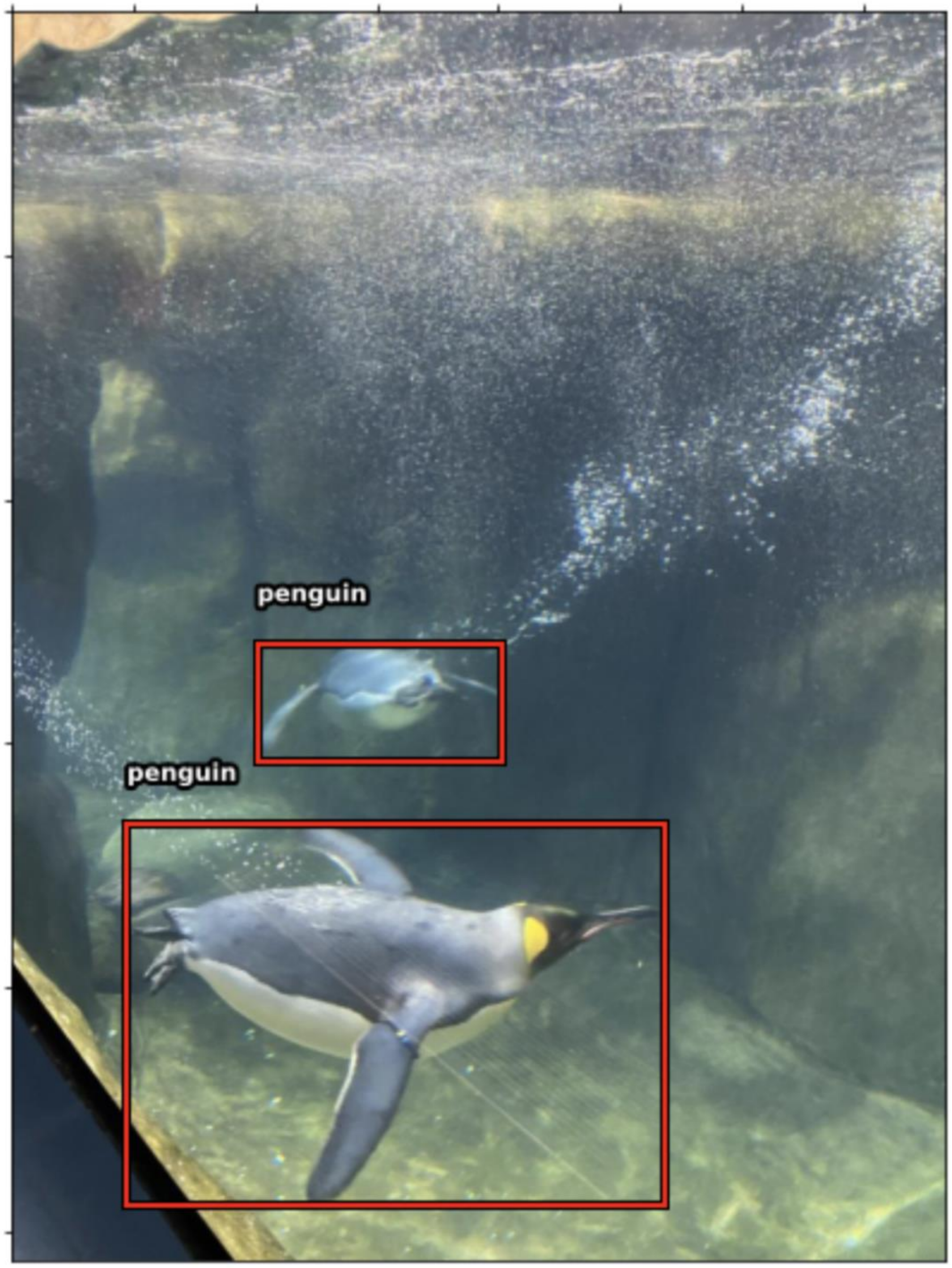
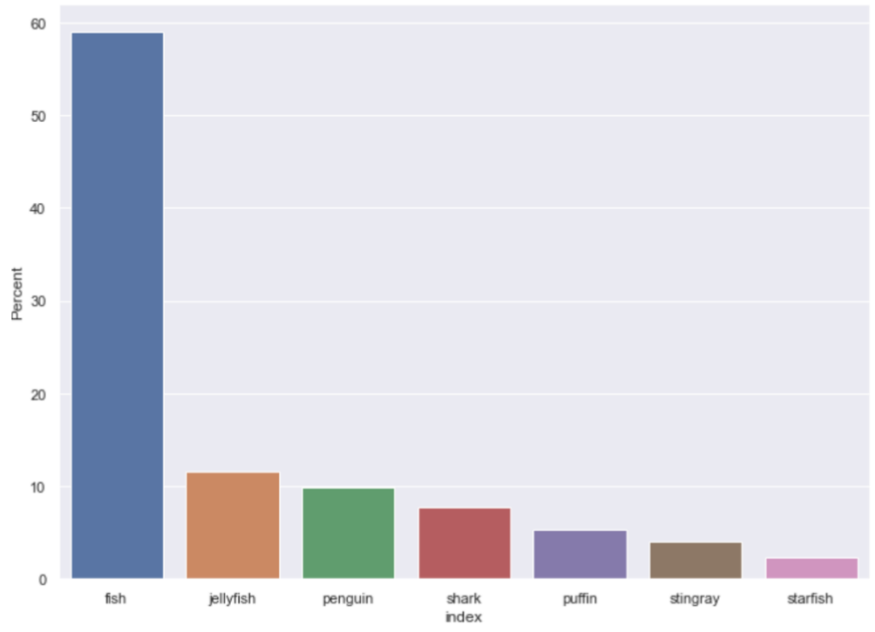
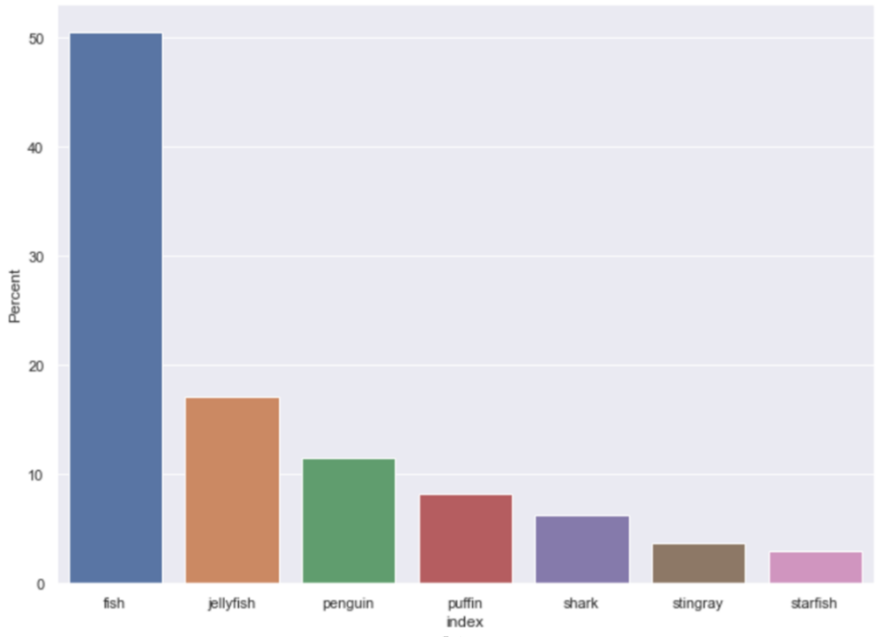


Figure 2. An example with labels from available dataset

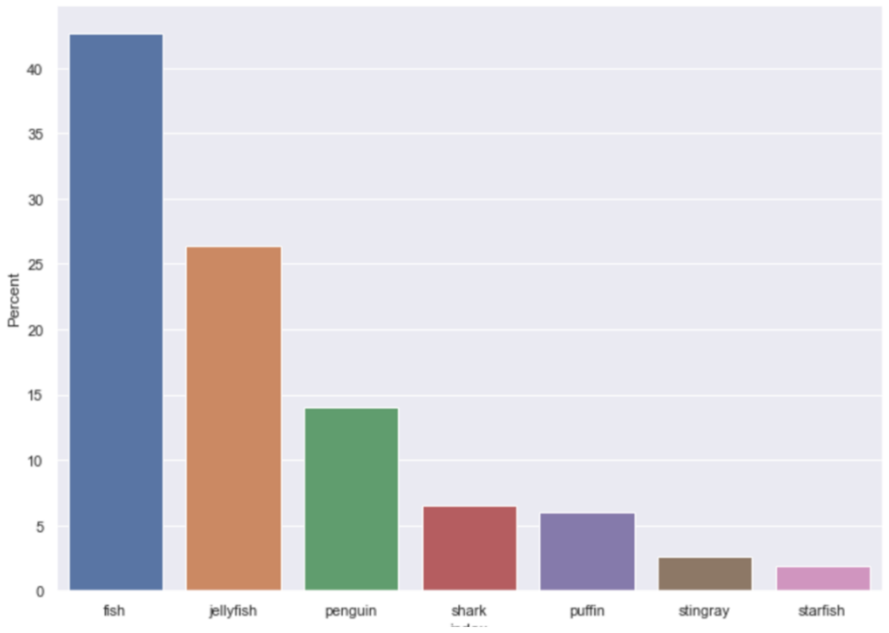
The dataset available from the website is already divided into training, validation and test sets such that there 447 images in training set, 127 images in validation set, and 63 images in the test set. The distribution of classes across these sets is shown in Fig. 3. It can be shown that classes are distributed more or less in similar to the original dataset in the training and validations sets. The test set seems to have little more labels with Jellyfish class. Regardless, all of them are dominated by the fish class. The training and validation sets were used for model training while the test set was used for model evaluation.



(a)



(b)



(c)

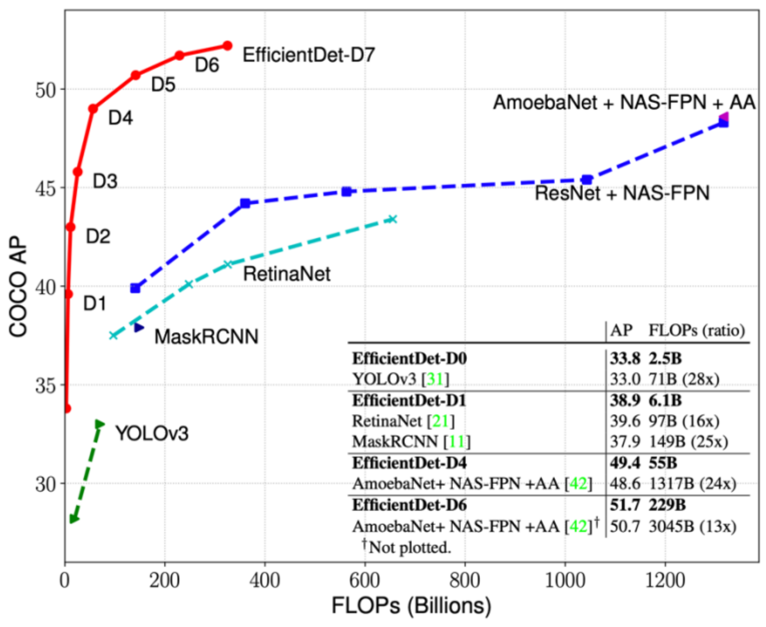
Figure 3 Distribution of classes across (a) Training set; (b) Validation set; and (c) Test set

3. Modeling

3.1 Detection and classification model

The EfficientDet architecture was written by Google Brain. EfficientDet is built on top of EfficientNet, a convolutional neural network that is pretrained on the ImageNet image database for classification. EfficientDet pools and mixes portions of the image at given granularities and forms features that are passed through a Bi-FPN feature fusion layer. The Bi-FPN combines various features at varying granularities and passes them forward to the detection head, where bounding boxes and class labels are predicted. An EfficientDet model trained on the COCO dataset yielded results with higher performance as a function of FLOPS. EfficientDet is a family of models expressing the same architecture at different model size scales. The EfficientDet model is implemented within the TensorFlow 2 Object Detection API.

We implement EfficientDet here within the TensorFlow 2 Object Detection API. The TensorFlow 2 Object Detection API allows you to quickly swap out different model architectures, including all of those in the EfficientDet model family and many more. TensorFlow 2 provides 40 pre-trained detection models on the COCO 2017 Dataset. This collection is the TensorFlow 2 Detection Model Zoo and can be accessed here. Every model has a Speed, Mean Average Precision (mAP) and Output. The EfficientDet D0 model has a speed of 39 ms and 33.6 mAP for COCO dataset. A comparison between different model and the EfficientDet architecture is shown in Fig. 4. Lower the FLOPs, faster the model runs. It can be seen that EfficientDet D0 provides reasonable AP for minimum FLOPs (Floating Point Operations per Second).



	AP	FLOPs (ratio)
EfficientDet-D0	**33.8**	**2.5B**
YOLOv3 [31]	33.0	71B (28x)
EfficientDet-D1	**38.9**	**6.1B**
RetinaNet [21]	39.6	97B (16x)
MaskRCNN [11]	37.9	149B (25x)
EfficientDet-D4	**49.4**	**55B**
AmoebaNet+ NAS-FPN +AA [42]	48.6	1317B (24x)
EfficientDet-D6	**51.7**	**229B**
AmoebaNet+ NAS-FPN +AA [42]†	50.7	3045B (13x)

†Not plotted.

Figure 4 COCO mAP vs FLOPs for different models [1]

Besides the Model Zoo, TensorFlow provides a Models Configs Repository as well. There, it's possible to get the configuration file that has to be modified before the training. The brief summary of the architecture is provided below but full details of the model are available in the original paper [1].

Fig. 5 shows the overall architecture of EfficientDet, which largely follows the one-stage detectors paradigm. The ImageNet-pretrained form the backbone network. The BiFPN serves as the feature network, which takes level 3-7 features {from the backbone network} and repeatedly applies top-down and bottom-up bidirectional feature fusion. These fused features are fed to a class and box network to produce object class and bounding box predictions respectively. The class and box network weights are shared across all levels of features.

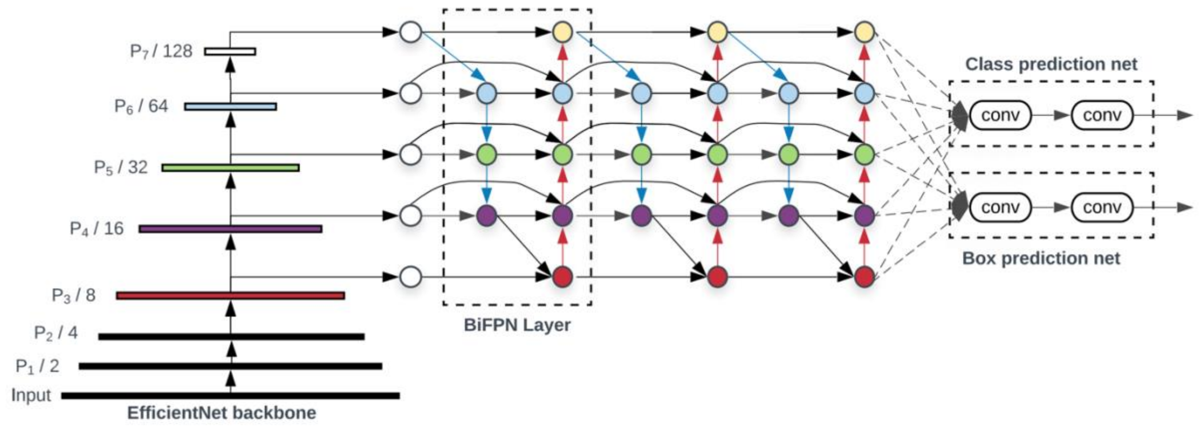


Figure 5 EfficientDet model architecture [1]

In this project, we’re going to use Tensor Flow 2 Object Detection (TF2OD) API and train the model using a Google Colaboratory Notebook.

3.2 Training the model

The model weights are TF2OD APIA are pre-trained on the COCO 2017 Dataset which has 330K images and 80 different object classes. Using a pre-trained model and applying it to a custom dataset with a different format (ex. Image dimensions, no. of classes etc.) is called transfer learning and it helps to speed up the learning process. This is done by modifying some parameters in config file for downloaded EfficientDet model.

The first change made to the config file was specifying the number of classes for classification which is 7 in this case. The batch value, which indicates how many images and labels are used in the forward pass to compute a gradient and update the weights via backpropagation, was set to 16. The width and height parameters, which indicate that every image will be resized to fit the network size during training and testing, are set to 512,512. The steps parameter was set to 40000 and num_eval_steps was set to 500. In addition to use, image augmentation methods such as random horizontal flipping, random scaling and cropping was also used.

After the dataset and configuration file were prepared, training was initiated on Google Colab with GPU enabled session. The model was trained for about 4900 steps when the classification and localization loss went below 0.2. It took approximately 1.5 hrs. The model was evaluated over the validation set, and the metrics are shown in Table 1.

Table 1 mAP after training the model

mAP @ IoU=0.5	0.67
mAP @ IoU=0.75	0.313

4. Evaluation and Discussion

The evaluation of models for object detection and classification require different metrics. The output of these model for each input image consists of detection box and a confidence score for both detection and classification. These models assign each detection a confidence score representing the confidence of both the detection and classification. The intersection over union (IOU) between two detection regions (e.g., a bounding box or mask) is the area of the intersection divided by the area of the union of the predicted region and the ground-truth region. This is used a similarity measure for the proposed bouding box and ground truth. When evaluating an image frame the proposed bounding boxes are matched to the ground truth bounding boxes. Correctly detected objects are called true positives (TP), defined as an object detected with the given IOU threshold, confidence threshold, and correct classification. Incorrectly detected objects are called false positives (FP), defined as objects detected but below the IOU threshold or with the wrong classification. Missed detections are called false negatives (FN), defined as objects with no predicted bounding box or a detection below the confidence threshold. There may be multiple or no objects in each image frame so the concept of true negatives is not typically used to evaluate multiple object detection and classification.

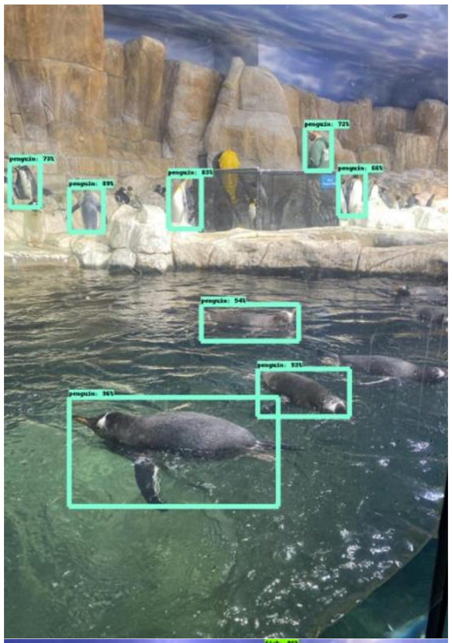
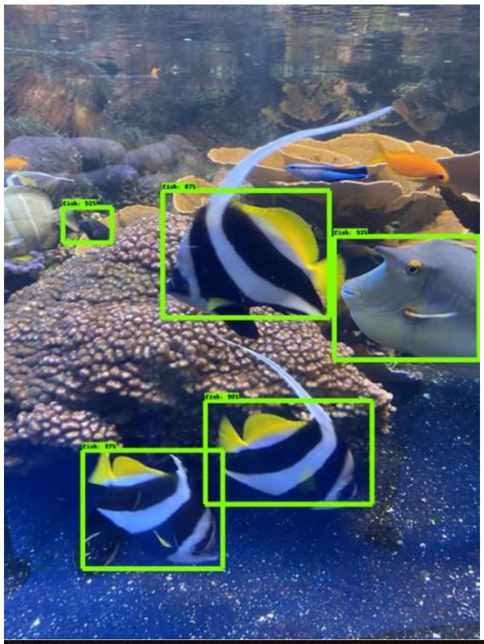
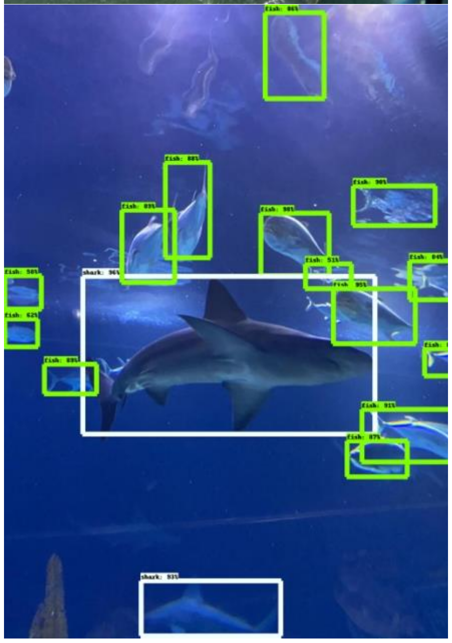
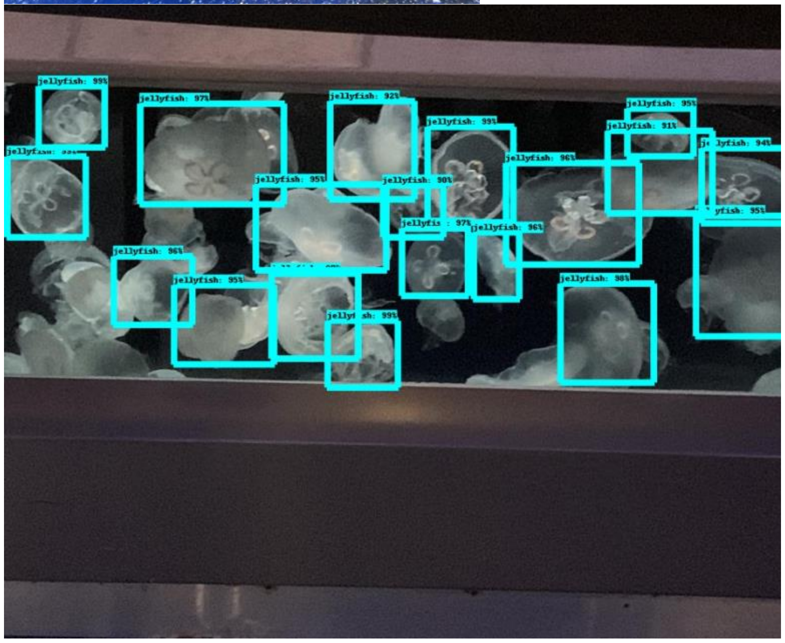
Precision is the number of true positives divided by the sum of true positives and false positives. Detections in precise models are likely to be correct detections. Recall is the number of true

positives divided by the sum of true positives and false negatives. Models with high recall tend to find most objects. There is a trade-off between these two concepts. Detections are likely to be correct from precise models at the expense of potentially missing low confidence detections. On the other hand, models with high recall tend to find most objects but may have a large number of false positives. Precision and recall may be analyzed using a precision-recall graph, with various model parameters resulting in different balances, but this can be difficult to optimize. A single metric is also considered which considers both precision and recall and is called the F1 score. It is the harmonic mean of precision and recall.
The trained model was applied to the test dataset and precision and recall values were calculated from the confusion matrix. The confidence and IoU threshold were equal to 0.5. These results are shown in Table 2. The average F1 score for the model is 0.14 which is very low. The model in general has high recall for all the classes compare to precision which are very low. The model performed will on some species with a small number of training samples.

Table 2 Precision and recall values for each class @ IoU=0.5

category	precision_@0.5IOU	recall_@0.5IOU	F1
puffin	0.08	0.26	0.12
starfish	0.24	0.82	0.38
stingray	0.06	0.80	0.11
fish	0.05	0.44	0.08
shark	0.07	0.61	0.12
jellyfish	0.03	0.80	0.06
penguin	0.06	0.32	0.10

Some of results from the test set are show in Fig. 6. It was observed that the model is predicting multiple bounding boxes with the same coordinates and scores. This was also deployed in the form of web app using Streamlit where a user can upload an image and the model will detect aquatic species and visualize the class, confidence score and bounding box.

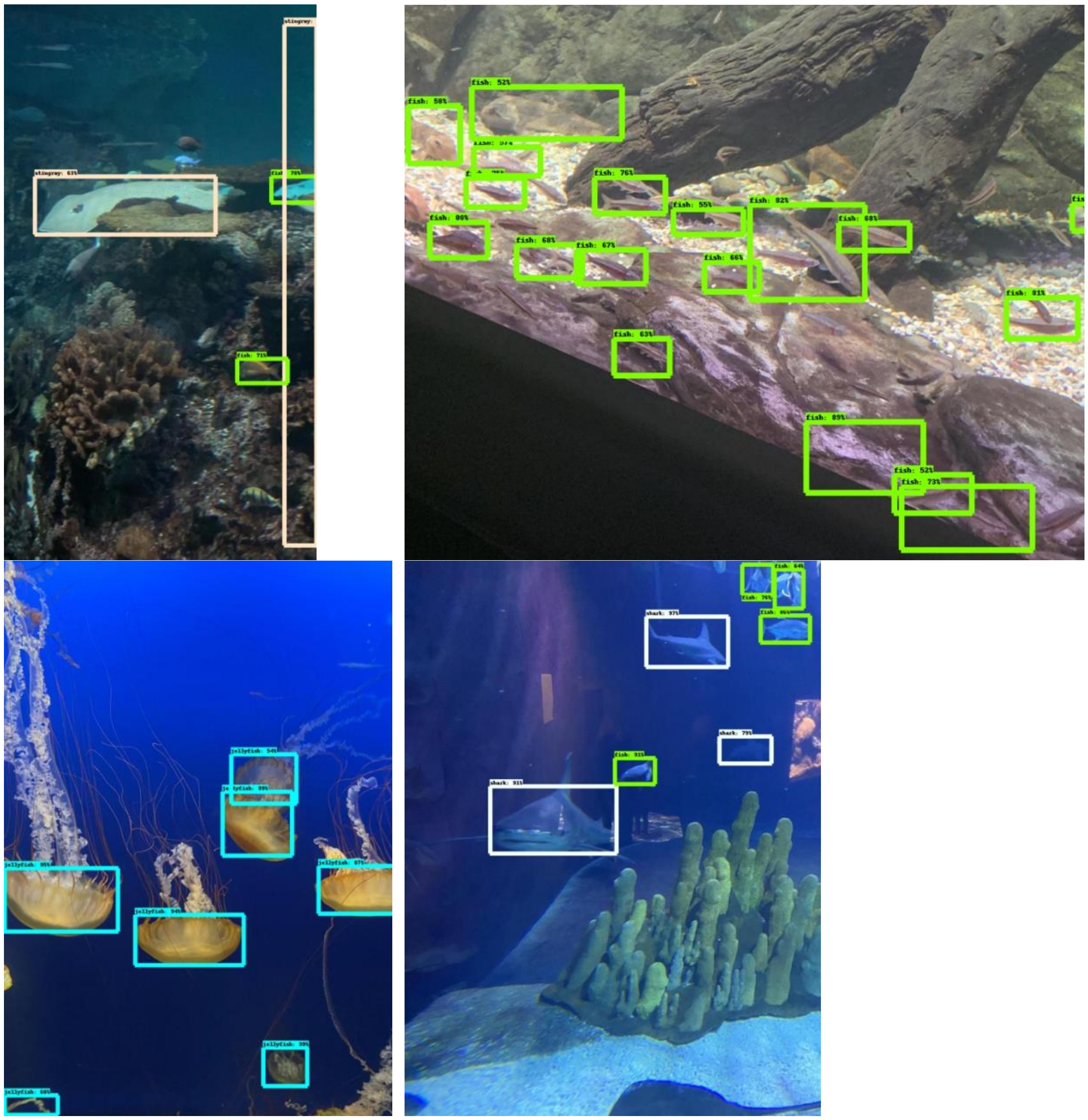


Figure 6 Predictions on test set

5. Conclusion

In summary, it was demonstrated that deep learning models are now a viable approach for detecting and classifying different aquatic species from aquarium images. One of the main challenges to solve is to increase the precision of the model. The trained model also takes about 30-40 secs to load into the website. However, the predictions are fast. The loading time which happens once needs to be reduce.

Reference

[1]https://openaccess.thecvf.com/content_CVPR_2020/papers/Tan_EfficientDet_Scalable_and_Efficient_Object_Detection_CVPR_2020_paper.pdf