AI for Coral Reefs

Segmentation team

Date: 14.12.2021

Final Report

Members:

Maks Kulicki

Mark Halverson

# 

# The challenge

## Goal

The primary objective of the image object segmentation team was tasked with locating coral in underwater coral reef photographs. Segmenting the images provides a way for researchers to compute the fraction of each image covered by coral, an important metric for coral reef ecology and conservation. Together with the colour correction and classification teams, a full suite of automated tools will allow researchers to accurately locate and identify coral species.

## Dataset

The data for this challenge consisted of thousands of ocean floor images in the SEAVIEW collection. Each image was supplied with 50 to 200 random point annotations that were generated manually. The annotations were made at the species and functional group level (e.g., hard coral, algae, etc...). The images were collected by many agencies, and together covered major tropical reef systems in the Pacific, Atlantic, and Indian Oceans.

## Modeling

There are many ways in which objects can be segmented from within an image. As mentioned above, one of the important metrics in coral ecology is the fraction of a reef covered by coral. Thus we decided to focus our effort on calculating this instead of building a system to perform tasks like semantic segmentation or object localization.

Our approach relies on the model developed by the classification team of this challenge. Their model is a fine-tuned GoogLeNet classifier which takes a 224 x 224 pixel image as input, and outputs one of six classes of benthic cover: algae, hard coral, soft coral, other invertebrates, sponge, and other.

Our algorithm first applies Simple Linear Iterative Clustering (SLIC) to a given image. This algorithm uses k-mean clustering, an unsupervised classification technique. SLIC divides the image into a given number of clusters (i.e., superpixels) based on their RGB values and proximity. Next, the full image is broken down into smaller regions of 224 x 224 pixels centered on each cluster centroid. Each sub-image is then passed to the GoogLeNet classifier to produce a label.

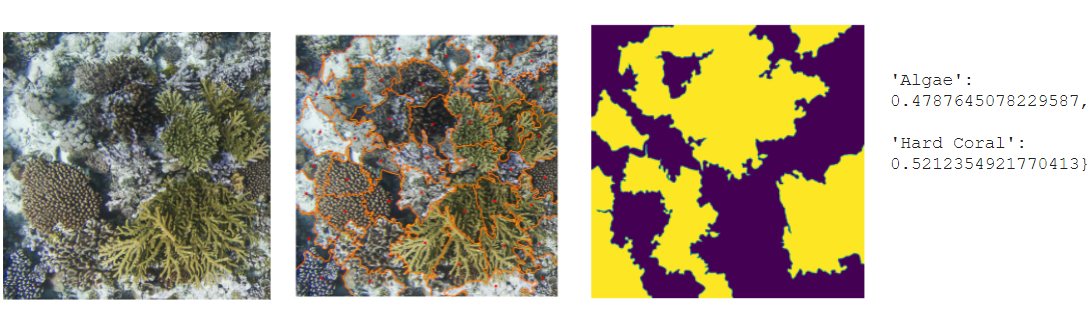
Then, the labels of the clusters are inferred - we assume that the whole cluster represents the same class as its centroid.

## Evaluation metrics

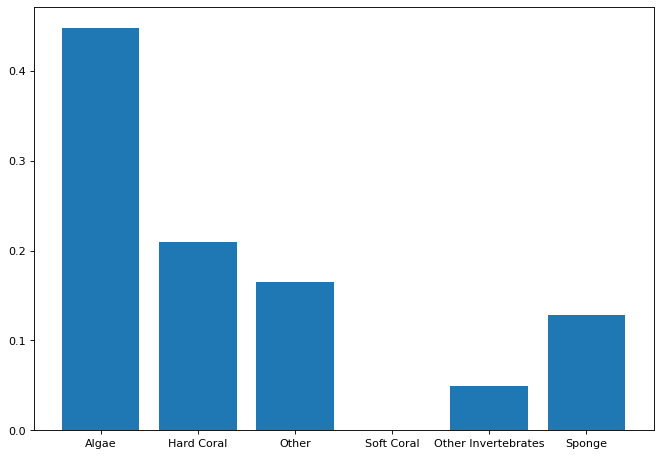
Evaluating our algorithm explicitly is difficult because there is no ground truth image-level data. The accuracy will likely depend strongly on the accuracy of the classifier, as well as the extent to which each subimage contains only a single function group.

# Results

Example segmentation of one image:

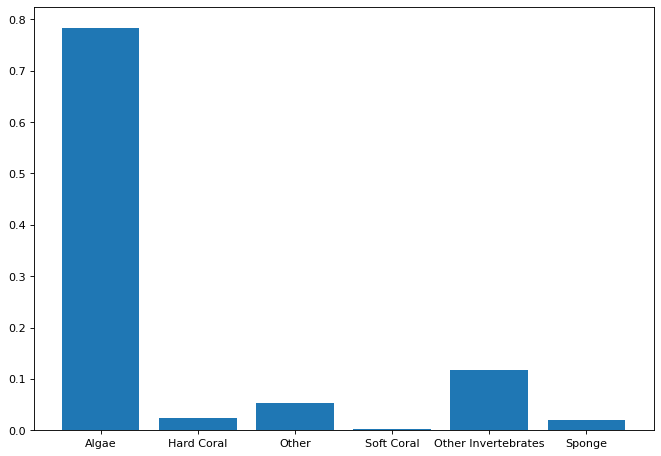


Coverage calculated from 50 first images from the IND/CHA dataset:



{'Algae': 0.44818849591785076, 'Hard Coral': 0.20903972953038108, 'Other': 0.16528110152135594, 'Soft Coral': 0.0, 'Other Invertebrates': 0.04903869989817584, 'Sponge': 0.1284519731322365}

50 first images from the ATL dataset:



We were not able to calculate coverage of the full datasets due to Colab RAM restrictions, only the first images. Therefore, these results might be skewed.

## Hurdles faced and solutions

### Team / People

The team size gradually dwindled over the course of the project. Some needed to attend to family issues, while others were too busy with work or school to contribute. Still others simply disappeared. Ultimately, only Maks and Mark made it to the end.

### Technical

#### Sparse labels for model training: software build problems

Many technical challenges were encountered. For example, one of the challenges was that the annotations were sparse. Each image contained about 1 million pixels, but only 50 - 200 were labelled. The pixel variability, whether noise or real, ultimately made it difficult to train an accurate model to classify the remaining pixels. This is a common problem in computer vision, and there exists a tool called [ML-Superpixels](https://github.com/Shathe/ML-Superpixels) to produce dense labels from sparse labels. However, we were not able to configure the software to run.

#### Existing coral segmentation models: TensorFlow dependency issues

One very promising avenue was [CoralSeg](https://github.com/Shathe/CoralSeg-Learning-coral-segmentation-from-sparse-annotations), a coral segmentation model. Unfortunately, the model was written using an old version of TensorFlow (1.x), and migrating it to TF 2.0 was not successful. We also tried to use the dataset the CoralSeg authors pro in their GitHub repo and train it using U-Net. Again, roadblocks occurred because there appeared to be problems with some annotations.

#### Assessing the accuracy of the ground cover calculation

A relative sense of the accuracy might be gained by developing other approaches and comparing the results. For example, we built a simple pixel classification model that was trained with the annotations. The RGB pixel values were extracted and compared to the labels. A random forest classifier was trained on the RGB values and functional groups, and from that model every pixel in an image could be labeled. Unfortunately this technique did not generalize well. An accuracy of about 70% was achieved using small training sets (approx. 20 images), however when trained on thousands of images the model did not generalize well. The accuracy decreased to less than 60%.

Image augmentation or blurring would probably have improved the accuracy. More advanced approaches, such as superpixel label augmentation, may have also improved the pixel classification approach.

## Conclusion

Mark’s recommendations: The groups are much too big. Next time, consider forming smaller groups that compete with another. Offering rewards will also improve people’s commitment level. The competitions organized by [ai.science](https://ai.science/) are a good template, for example.