AI for Coral Reefs

Group: Coral Classification

December-2021

# Final Report

Julieta Millan (julimillan6@gmail.com), Parth Dandavate (dandavateparth@gmail.com), Nathanya Queby Satriani (nathanyaqueby21@gmail.com), Kim van Asten (kimvanasten@gmail.com), Leonard Meyer (meyerle@uni-bremen.de)

---

## The challenge

**Goal**: The goal of this subgroup was to automate the classification of image content. Here specifically, images contained a range of static benthic organisms such as hard coral, algae and bare substrate that are commonly found on tropical coral reefs. Ideally, the final model would accurately differentiate between at least coral (hard and soft) and non-coral objects, or eventually among a group of several biologically functional entities. The final model should further be able to classify chosen groups with an accuracy >75% and should be presented including measures of additional evaluation against ground truth information.
An extra goal was to obtain an estimation of coral coverage of the terrain based on the model's predictions.

## Data

Image data was chosen from the SEAVIEW[1] data set, which contains mid-range resolution images for nine different ocean regions, as seen in Table 1.

| Folder | Region | Folder size | Number of images |
|---|---|---|---|
| ATL | Atlantic | 2.1 GB | 1407 |
| IND_CHA | Indian Ocean, Chagos Archipelago | 0.9 GB | 686 |
| IND_MDV | India Ocean, Maldives | 2.5 GB | 1612 |
| PAC_AUS | Pacific Ocean, Australia | 2.7 GB | 2657 |
| PAC_USA | Pacific Ocean, USA (Hawaii) | 2.9 GB | 1153 |
| PAC_IDN_PHL | Pacific Ocean, Indonesia and Philippines | 2.7 GB | 1638 |
| PAC_SLB | Pacific Ocean, Solomon Islands | 0.8 GB | 732 |
| PAC_TWN | Pacific Ocean, Taiwan | 0.7 GB | 638 |
| PAC_TLS | Pacific Ocean, Timor-Leste | 1.4 GB | 864 |

Table 1. Organization of the SEAVIEW dataset, with the name of the folders, the oceanic regions they correspond to, size of the and number of images in each folder (Courtesy of ReefSupport).

Whereas each of the images of the global survey is peppered with 40 - 50 pixel points, which are labeled within the appended .csv files as one of several semi-uniform groups. The .csv files come in a format seen in Figure 1.

| | quadratid | y | x | label_name | label | func_group | method | data_set |
|---|---|---|---|---|---|---|---|---|
| 0 | 17001738002 | 180 | 183 | Fish | FISH | Other | random | test |
| 1 | 17001738002 | 199 | 162 | CCA | CCA | Algae | random | test |
| 2 | 17001738002 | 197 | 160 | CCA | CCA | Algae | random | test |
| 3 | 17001738002 | 88 | 561 | Turf | Turf | Algae | random | test |
| 4 | 17001738002 | 195 | 714 | Ocomplex | OCOM | Hard Coral | random | test |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

Figure 1. Example of annotation csv file.

[1] XL Catlin Seaview Survey (http://data.qld.edu.au/public/Q1281/)

From those files, we focused on four columns: ‘quadratid’, ‘x’, ‘y’ and ‘func_group’.
The column ‘quadratid’ refers to the image name or id; and the ‘x’ and ‘y’ columns, to coordinates corresponding to pixels in those images that were previously labeled by experts. The ‘func_group’, meaning functional group, correlates the pixels with the labels that we used to train our models. Those labels were 'Algae', 'Hard Coral', 'Other', 'Soft Coral', 'Other Invertebrates', 'Sponge'. The ‘Sponge’ label was only present in the ATL folder, which must be taken into account at the time of making predictions.
One of our main problems was the fact that the classes are highly unbalanced (Figure 2), and every attempt to fix this issue would have a large impact on the dataset.

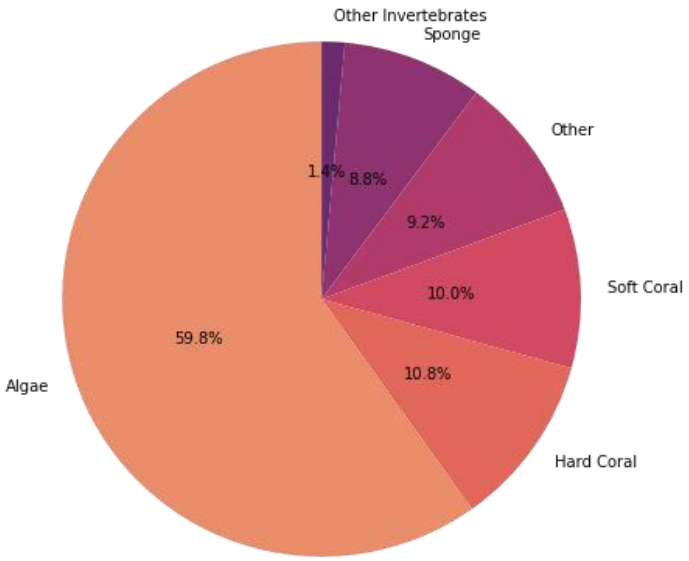


Figure 2. Percentage of class representation on the ATL dataset.
More than half of the labels correspond to the ‘Algae’ class.

**Data: Pre-processing**.

Image Cropping

The column ‘func_group’ (functional group) was chosen to include the label information for the algorithm and to provide sufficient biological information according to the goal. To overcome sparse labels (= 50 pixels in an approximately 1000 x 1000 pixel image) and achieve dense labels on each image, point labels of chosen functional groups were expanded to dimensions of 224x224 - this format is also a common input size for many image classifying algorithms. Like this, each image gave a product of 50 x 224x224 pixels smaller images (Figure 3a and 3b).

Note that point labels are not centered to their labeled organism or the original image dimensions, therefore cropping's might contain other benthic organisms (even in the majority of pixels) as well as can be sliced by picture corners as seen in Figure 3c.

Data storage

We tried different approaches for the generation and storage of these preprocessed images.

1. Our first idea was to store the already cropped images as Hierarchical Data Format (H5) files. We would create numpy arrays of 224x224x3 (width x height x 3 RGB channels) from the larger images, and save them. We soon realized that working with that amount of data would quickly fill our available RAM and would only allow us to use very small batches of images at a time.
2. Finally, we decided to use a custom Pytorch Dataset. Here, we created a class that crops the images based on the annotation file each time the model is trained. The advantage is that using the GPU allows us to work with large volumes of data at the time, but since the images are constantly being manipulated it also takes longer to run the script.

Image Normalization

For the use of pre-trained models the input images must be normalized in the same way, using a mean= [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].

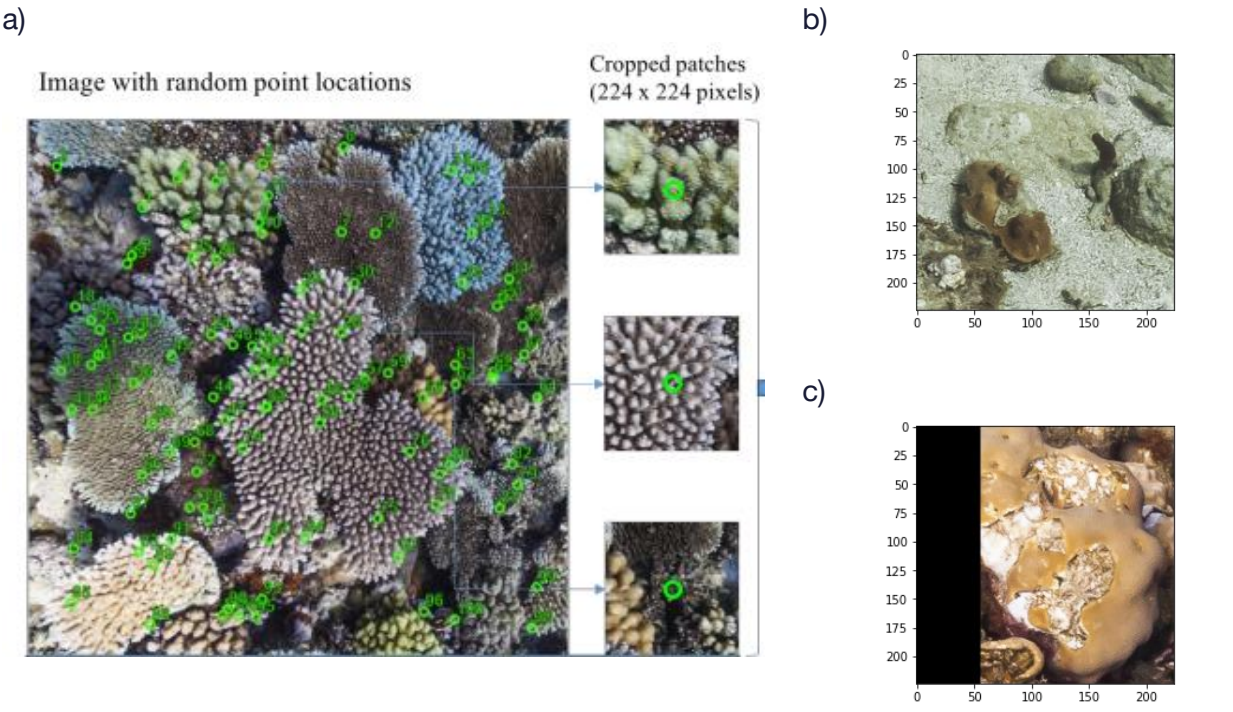


Figure 3. a) Example of image with random points and the 224x224 crop around them (Figure courtesy of ReefSupport) b) Example of a 224x224 cropped image. c) Crop of a point close to the image border.

**Data: Modeling**.

All data operations were executed in Google Colaboratory using Python and several network architectures were tested.

Vision Transformer

A Transformer is a model architecture that implements an encoder-decoder structure without the recurrence and convolutions, and relies entirely on self-attention mechanisms. It is largely used for Natural Language Processing and was introduced by Ashish Vaswani and collaborators in 2017 in the paper 'Attention Is All You Need'. In October 2020, Alexey Dosovitskiy and collaborators presented a similar architecture for image classification in 'An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale', that is now known as Vision Transformer (ViT) (Figure 4).

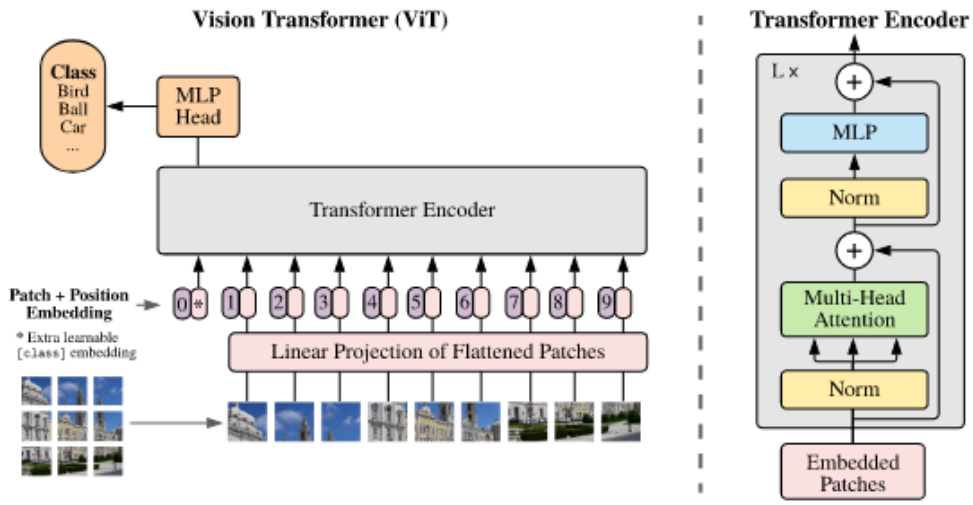


Figure 4. Vision Transformer model overview. The authors split an image into patches, linearly embed each of them, add position embeddings and feed the resulting sequence of vectors to a standard Transformer encoder. To perform classification, they use the standard approach of adding extra learnable "classification token" to the sequence (Dosovitskiy et al. 2020).

We used the Huggingface ViT implementation as shown in Vision Transformer (ViT).

VGG16

VGGs are a group of convolutional neural networks with an architecture based on depth. They were presented by Karen Simonyan and Andrew Zisserman in 2014 in the paper “Very Deep Convolutional Neural Networks For Large-Scale Image Recognition”. These networks use increased depth with very small convolutional filters.

All configurations follow generic design and differ only in the depth: from 11 weight layers to 19 weight layers. We apply the most widely used, the VGG16 (Figure 5), with 16 weight layers as shown in vgg-nets.

We used a batch size of 32, Learning Rate of $1x10^{-3}$, Adam optimizer, Cross Entropy Loss and started training with 2 Epochs. Most of these hyperparameters were defined by default and changing them depended on our computational power.

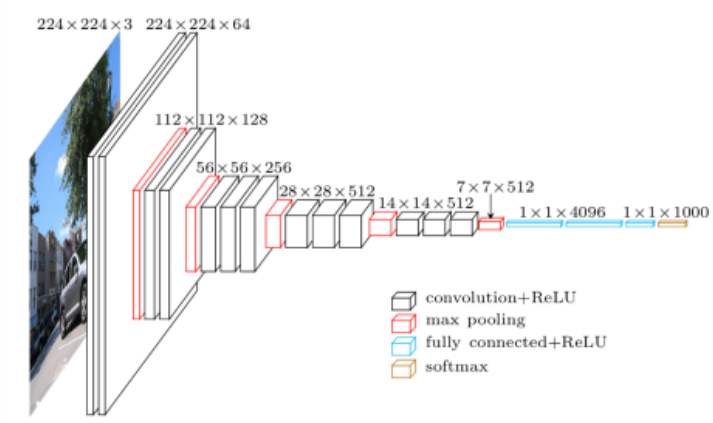


Figure 5. VGG16 Architecture.

GoogLeNet

This model (Figure 6) is a convolutional neural network based on the ‘Inception’ architecture, which was responsible for setting the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014. It was introduced in 2015 by Szegedy and collaborators in the paper ‘Going Deeper With Convolutions’ [3]. Most of our problems for the classification were based on computational power so this architecture is ideal since its main characteristic is improved utilization of the computing resources inside the network.

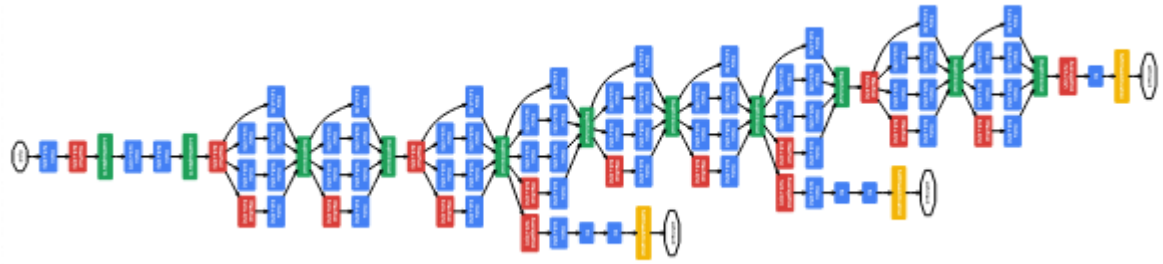

Figure 6. Full GoogLeNet architecture as shown in the paper 'Going Deeper With Convolutions' [2].

For our classification problem, we used the pre-trained model as shown in Pytorch's official GoogLeNet website[2]. We performed model fine-tuning by changing the last fully connected layer to fit our class number, and pre-processed the images as explained above. We used a batch size of 32, Learning Rate of $1x10^{-3}$, Adam optimizer, Cross Entropy Loss and started training with 2 Epochs. Most of these hyperparameters were defined by default and changing them depended on our computational power.

**Data: Evaluation metrics**

We analyze the accuracy of our model's predictions, but since the classes are highly unbalanced as explained previously we also use confusion matrices to be sure of how well the models are performing.

## Results

VGG16

For reasons we were unable to determine, the VGG performance was very low, reaching a 19% accuracy. The results are not shown here.

GoogLeNet

The first training was made for two epochs on the ATL folder. The train-test split was 0.75 and 0.25 respectively. Our model got 19478 correct predictions over a total of 23225 showing an accuracy of 83.87%. The confusion matrix is shown in Figure 7.

a)

```
[13063,    95,   208,    23,   392,    49]
[  526,  1736,    12,     6,   163,     8]
[  663,    75,  1381,     6,    94,     8]
[  123,    37,     9,   166,    22,     0]
[  320,    97,    19,     2,  1876,    19]
[  563,   118,     9,    16,    65,  1256]
```

b)

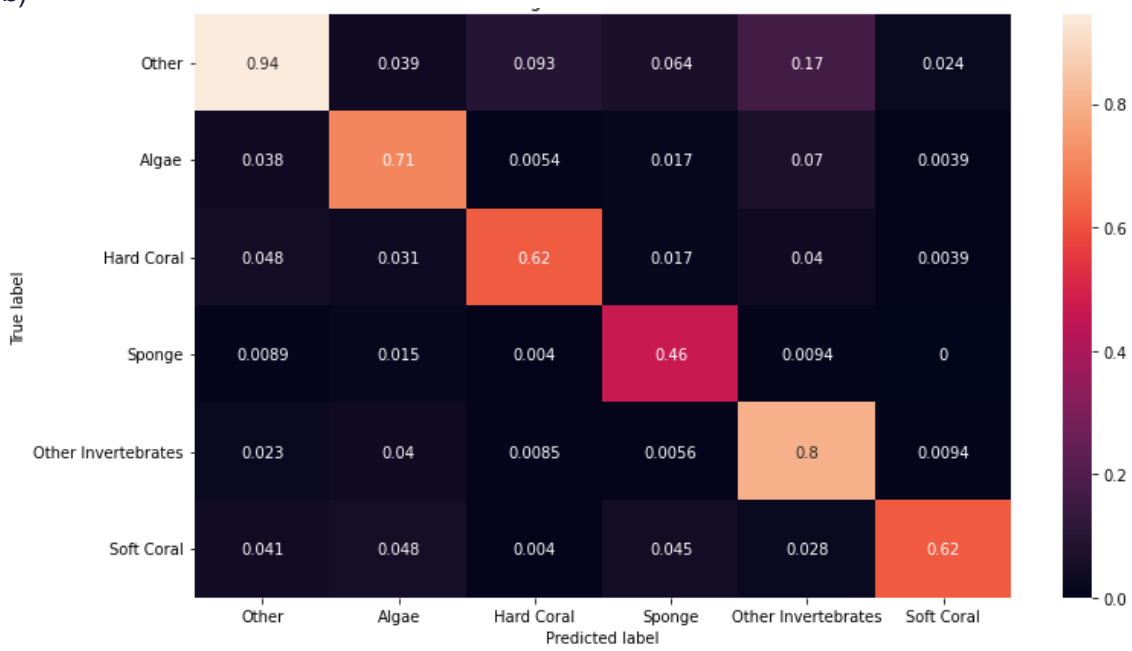


Figure 7. Confusion Matrices of the first trained GoogLeNet model. a) Matrix of true labels vs. predicted labels, shown in integer numbers. b) Given the class imbalance, each predicted class was normalized over the total number of images in that class, and not the total images across all classes.

We saved the weights and optimizer's state dictionary of this training and used them to train again the same model with the ATL folder for another two epochs in order to improve performance. The results were 19967 images correctly predicted, with an accuracy of 85.97% (Figure 8).

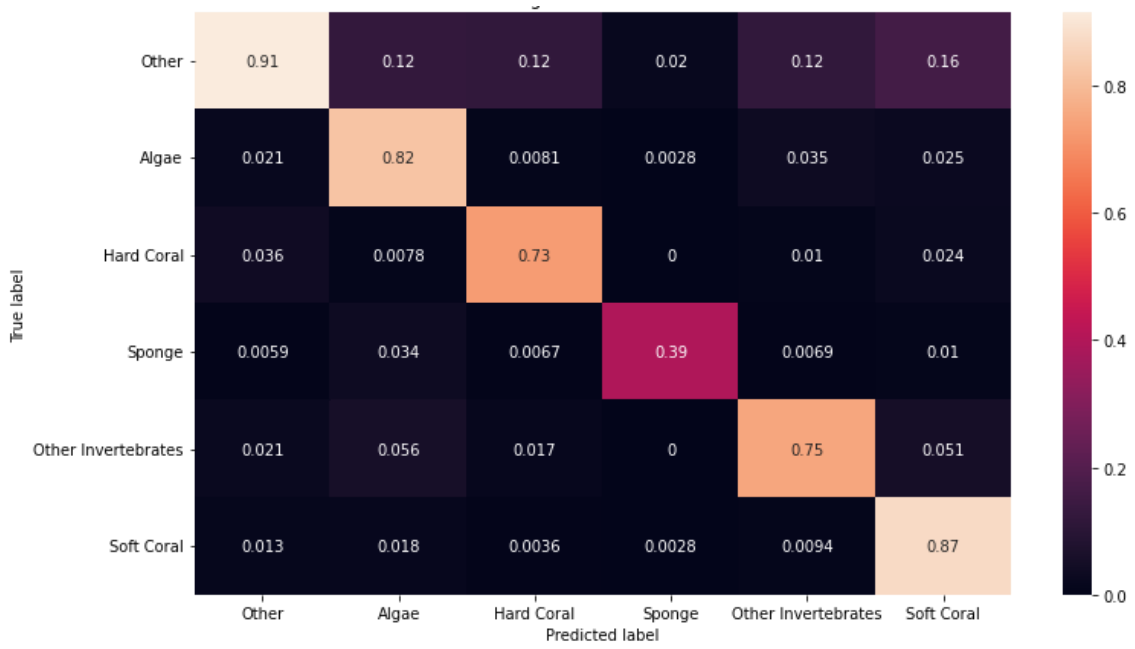


Figure 8. Confusion Matrix of the second trained GoogLeNet model. Given the class imbalance, each predicted class was normalized over the total number of images in that class, and not the total images across all classes.

Up to this point, the model has been trained for four epochs, reaching an accuracy of approximately 86%. Training for more epochs could increase that accuracy even further. The model performs similarly in different folders.

Coral coverage

Finally, for estimating the amount of coral coverage, we used the predictions from our best performing model.

We assume that the total labeled points in each dataset are a representative sample of the environment. Considering this, we say that the coral coverage is given by the points predicted as coral over the total number of points.

For the ATL dataset, the ground truth for hard coral coverage is of approximately 10% while our predictions show a 34.59% coverage. The ground truth for total coral (hard + soft corals) is approximately 20%, and our model's is 39.96%. We believe these values could be improved just by increasing the model's accuracy.

Bibliography

1. XL Catlin Seaview Survey (https://www.catlinseaviewsurvey.com/)

2. https://pytorch.org/hub/pytorch_vision_googlenet/

3. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

https://huggingface.co/docs/transformers/model_doc/vit

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

https://pytorch.org/hub/pytorch_vision_vgg/