Home Work #2

Detection of Vegetation in Baltic Sea

Romane Lucas 214306IV

Sanan Suleymanov 213860IASM

1. Introduction

The detection of vegetation in Baltic Sea is performed by feature detectors as well as using deep learning algorithms. Firstly, the images are resized to 1024x1024 and then set of etalons are created. For task 3, SIFT, ORB and BRIEF feature detectors are used and for the improvement of baseline the object detection approach is applied because of its appropriateness for this project. The dataset is splitted to the training and testing datasets and labeled using the bounding box method. For seeing the improvement of baseline, as a metrics Confusion Matrix is implemented.

2. Classification using feature detectors

As a first approach we have done a classification of the images of our data set by means of feature detectors. We have used three different feature detectors provided by open-cv : SIFT, ORB and BRIEF. The dataset is composed of 95 images and we also have a set of etalons which is composed of 5 small images per class.

To predict the type of water plant which is in the data images we try to match the images with all the etalons. The class which obtains the most total matches is chosen as prediction. We are so able to compute the prediction for the 3 different feature detectors.

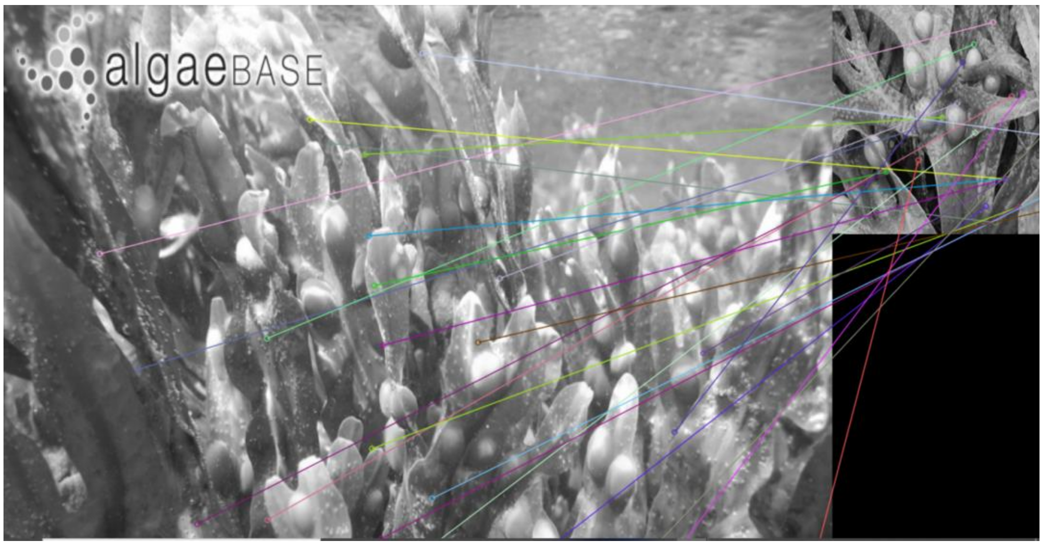
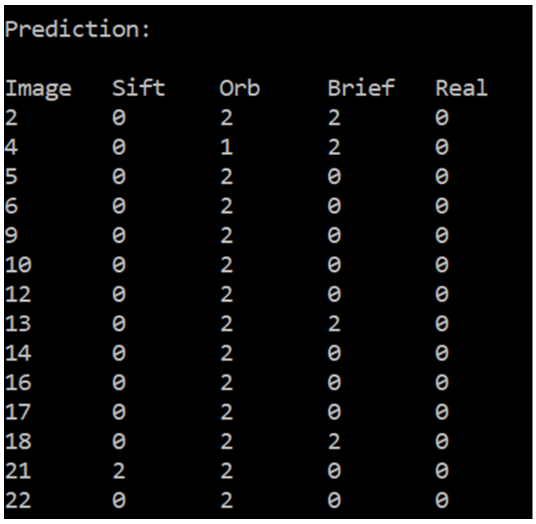


Figure 1. Example of matching keypoints displayed using the sift detector

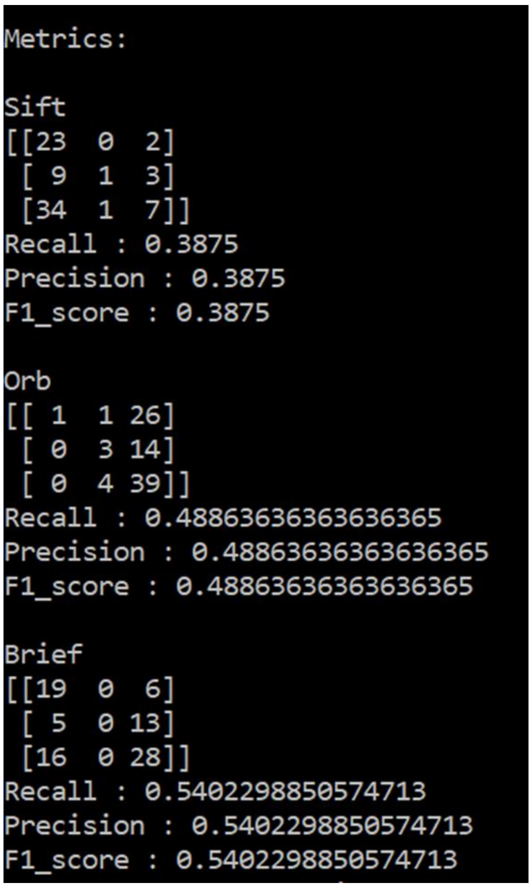
To calculate metrics we read the real classe from the image label and using these real matches and the predicted one we compute the confusion matrix, the recall, the precision and the F1-score. In the case where the standard comes from the tested image and that the prediction is good, this one is removed from the calculation of the scores to avoid overlapping. We can recognise that a etalon is a part of an image with the score obtained by SIFT detector because this is then greater than 300 which is never the case otherwise.

We have chosen to use these metrics because the confusion matrix allows us to obtain True Positive value : the value of the intersection of the corresponding column and row (the diagonal). The False Negative value : the sum of values of corresponding rows except the TP value. The False Positive value : the sum of values of corresponding columns except the TP value. And the True Negative value : the sum of values of all columns and rows except the values of that class that we are calculating the values for. The Recall value gives us the percentage of positives well predicted. If the value is high, this means that the model will not miss any positive. The Precision value gives us the number of positive predictions made well. If the value is high, the majority of the model's positive predictions are well-predicted positives. And we have also used the F1-Score, it’s the harmonic mean of recall and precision. If high, the model is efficient.



```
Prediction:

Image   Sift    Orb     Brief   Real
2       0       2       2       0
4       0       1       2       0
5       0       2       0       0
6       0       2       0       0
9       0       2       0       0
10      0       2       0       0
12      0       2       0       0
13      0       2       2       0
14      0       2       0       0
16      0       2       0       0
17      0       2       0       0
18      0       2       2       0
21      2       2       0       0
22      0       2       0       0
```

Figure 2. Prediction of classification by SIFT, ORB and BRIEF feature detector

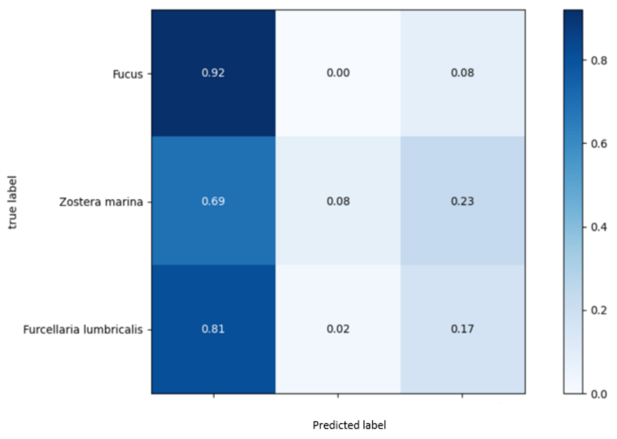


```
Metrics:

Sift
[[23  0  2]
 [ 9  1  3]
 [34  1  7]]
Recall : 0.3875
Precision : 0.3875
F1_score : 0.3875

Orb
[[ 1  1 26]
 [ 0  3 14]
 [ 0  4 39]]
Recall : 0.48863636363636365
Precision : 0.48863636363636365
F1_score : 0.48863636363636365

Brief
[[19  0  6]
 [ 5  0 13]
 [16  0 28]]
Recall : 0.5402298850574713
Precision : 0.5402298850574713
F1_score : 0.5402298850574713
```

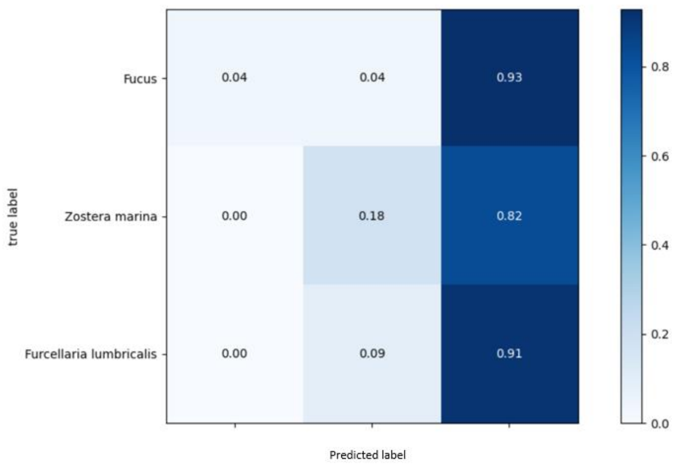
Figure 3. Confusion matrix, Recall, Precision and F1-Score for the 3 feature detectors

We also plot the confusion matrix using the mlxtend library in order to have the same calculation for both approaches.

SIFT



ORB



BRIEF

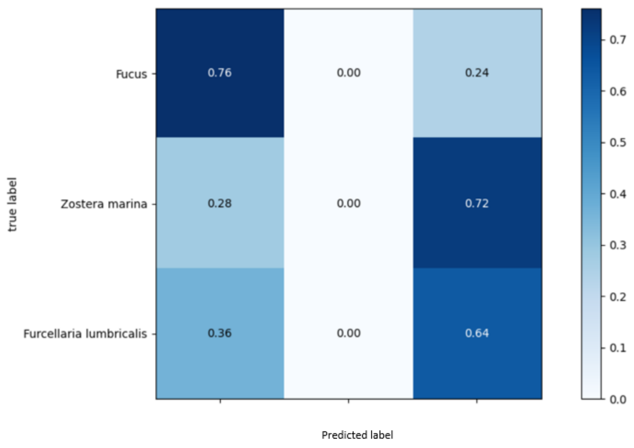


Figure 4. Confusions matrix using mlxtend library

Finally the results obtained are average but it seems normal for this type of classification. Perhaps the results could have been improved by better choosing the stallions or by using better quality images.

The folder Data and Etalons has to be in the same directory as the feacture_detectors.py file. In the Etalons folder we have one folder per class which contains the etalons for this class.

3. Detection using YOLOv5

For improvement of baseline, we have applied deep learning based object detection. For labeling data, LabelImg tool was used and they are labeled as bounding box and saved in .txt format. Detection was done for 3 classes: Fucus, Zosterina Marina and Furcellaria lubricalis. In our datasets, 111 images for training and 19 images for testing were used. As an architecture, we used YOLOv5 and resultant experience is quite high because of the fastest training process. Before YOLOv5, we used the Object Detection API of tensorflow with ResNet50 architecture while during the training process we encountered so many problems and only one time we were successful in the training and testing process.

We did many experiments for training with changing the number of the epochs, and each result of training and prediction according to the test dataset are in the yolov5/runs/train directory. The detection of aforementioned plants using YOLOv5 architecture in 5th experiment is shown in Figure 1 which is considered to give us quite normal result with 200 epochs compare to others.

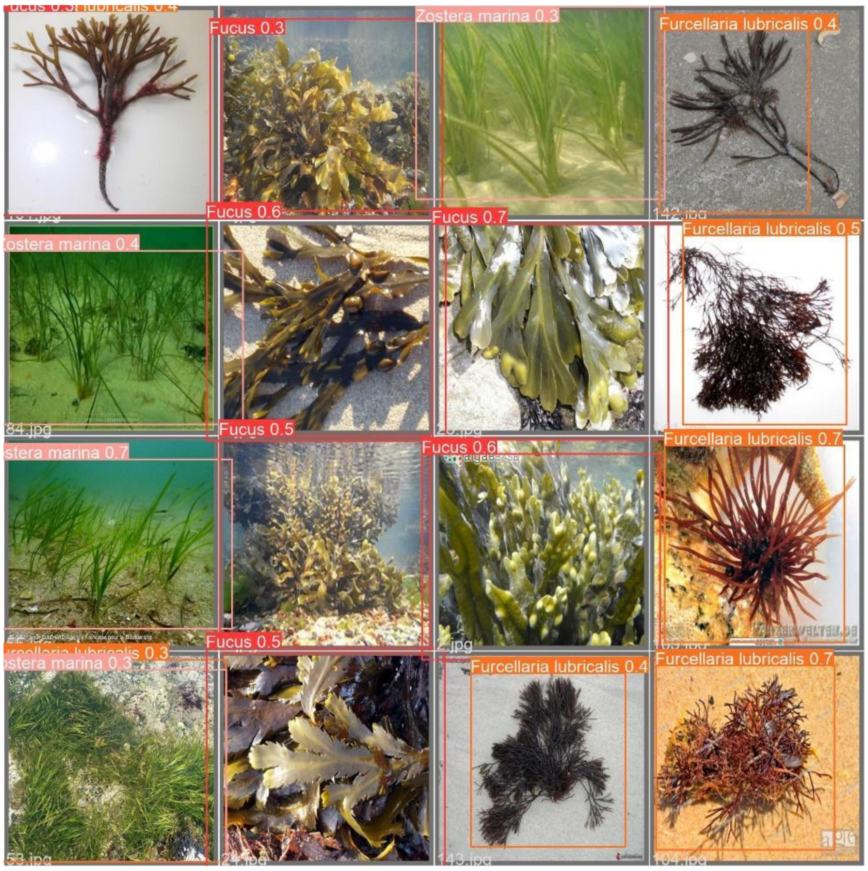


Figure 5. Detection of plants using YOLOv5 architecture.

It detects plants in each image while it detects Furcellaria lubricalis as Fucus with 0.29 and the Confusion Metrics of this experiment is described in Figure 2.

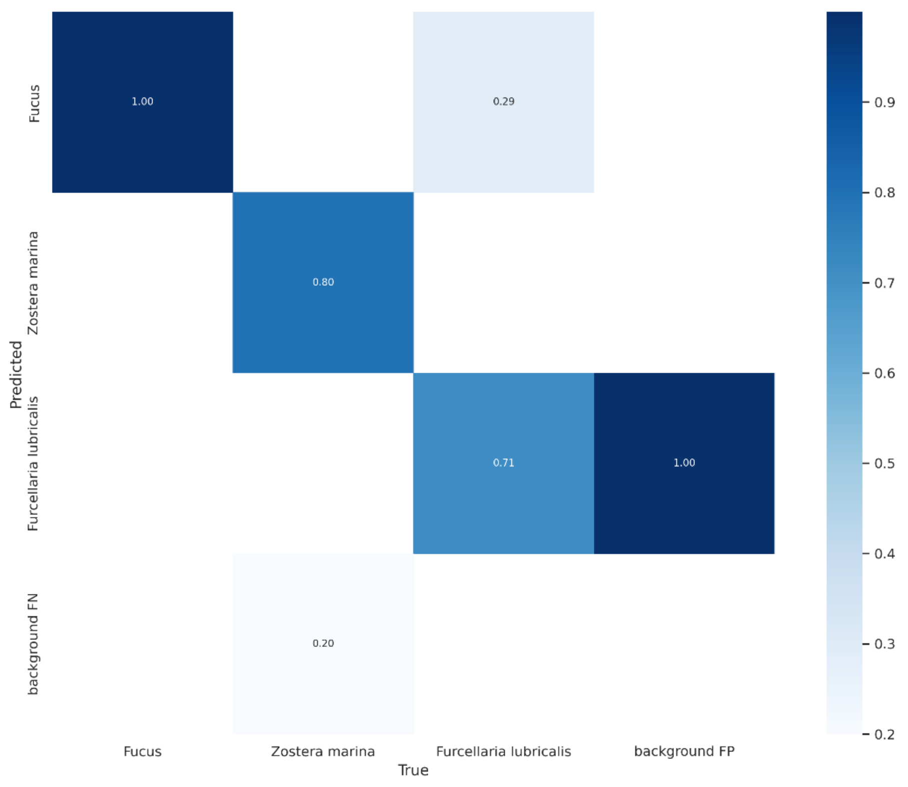


Figure 6. Confusion Matrix of 5th experiment using YOLOv5

The directory of images and the information about our classes are denoted inside of custom.yaml which is in the yolov5/data. For performing the training and prediction, it is needed to run main.ipynb.

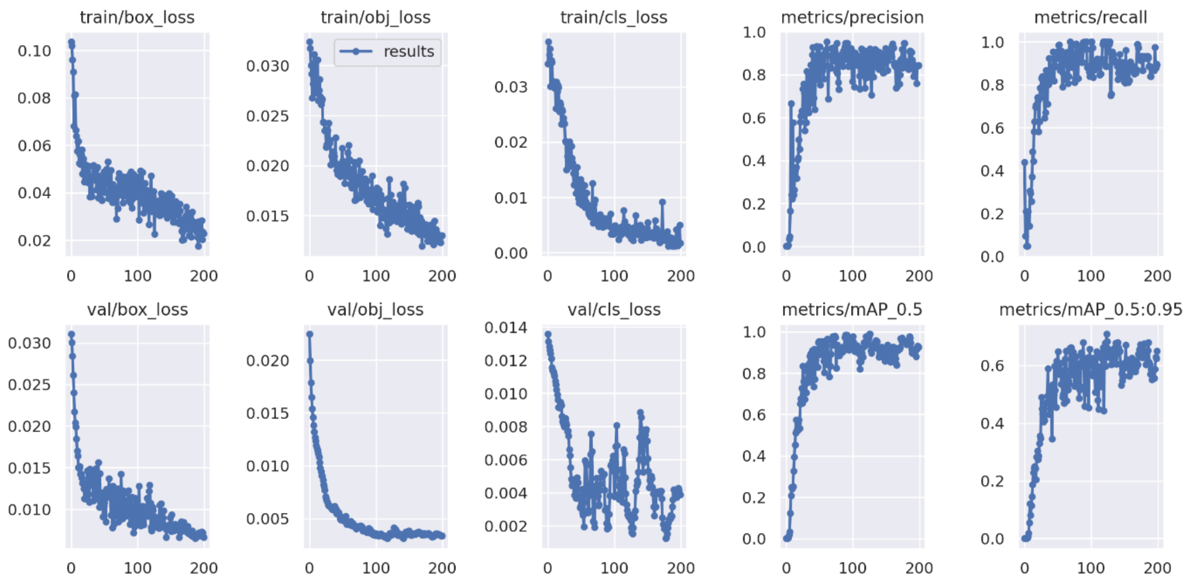


Figure 7. The results of training for 5th experiment

The results of the training during each epochs are described in the results.csv and its graphical represntation is shown in Figure 7.

4. Conclusion

As a conclusion, we consider that the reason for not getting high results is because of the dataset. We had more than images which were used for training and testing. However, some of them had low quality, maybe it happened after resizing of images and also, some of them were not appropriate for labeling and we had to filter images. Even after filtering, we again were not so sure about appropriateness of datasets for the object detection. Moreover for the feature detectors approach the results depend a lot on the type of detectors used. Certain classes are more favored by certain feature detectors than others.

For investigating the effect of using approaches to our baseline, confusion matrix was used and from the aforementioned Confusion Matrixes, it is clearly seen that the application of deep learning based Object Detection improved our baseline.