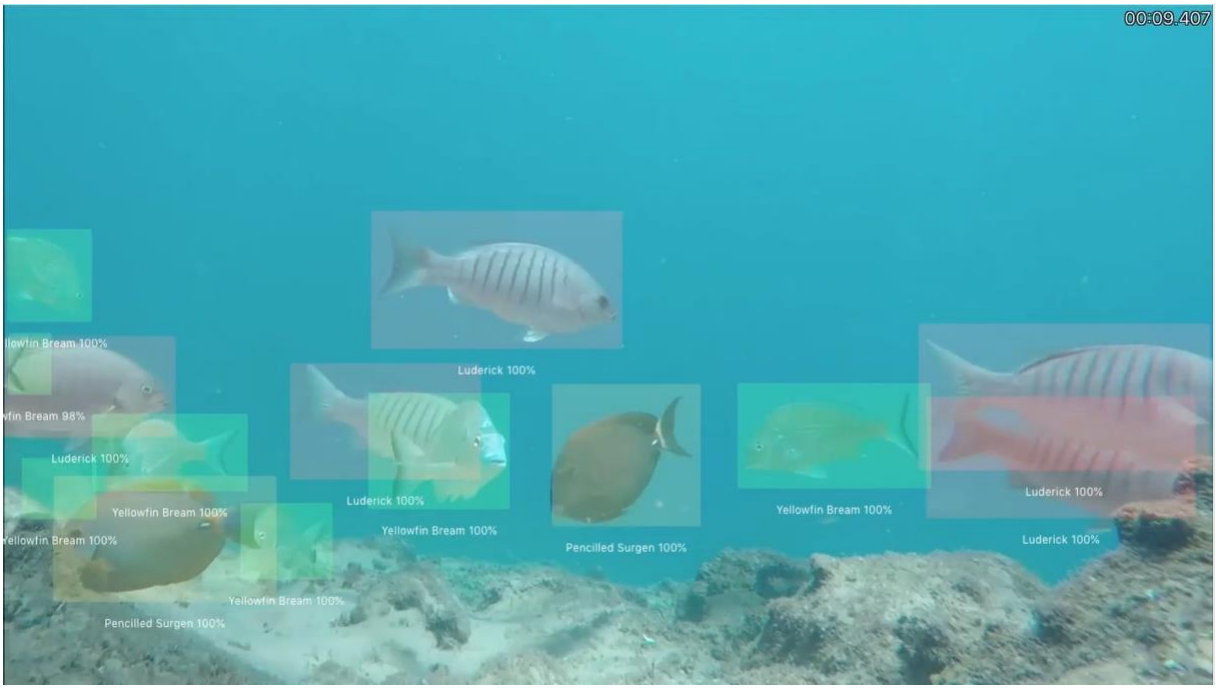
TER 2021-071 – Development

Title:

Deep Learning System for animal recognition in underwater images and videos from Mediterranean Sea



Students:

Giuseppe Alessio Dimonte	Master 2 EIT Digital (DSC)
Arturo Adan Pinar	Master 2 EIT Digital (DSC)
Yueqiao Huang	MAM5

Supervisors:

Frederic Precioso

Diane Lingrand

Date: 14/02/2022

1. Presentation of the subject

1.1 Problem

One of the greatest humans' abilities is the detection and classification of objects in visual scenarios (real life, images, and videos) with a natural accuracy, simplicity, and velocity. Humans preserve this ability also for underwater ecosystems, being able to easily recognize the position of fishes, corals, and marine-related objects. Beyond that, humans with a certain expertise in the marine field are capable to distinguish different species, classifying each object in the correct way.

What is required in the development of the project is to automatically detect and recognize fishes in the marine ecosystem without passing through experts each time it is necessary to perform this kind of task. In fact, in the case the process needs to be done at a high scale and considering many species, it might become an extensive process requiring a notable amount of time. Fish experts should spend a substantial amount of their time drawing the shape and assigning the label for each fish. On a practical side it is clearly an ineffective use of time, energy, and money. Considering an average time of 2 minutes for a single image annotation performed by an expert, the annotation of 1000 images translates into more than 33 hours to perform the task. This amount of time becomes a waste of the costly experts' working hours.

To tackle this problem a machine learning solution can be deployed to accomplish the same duty in an automatic way and saving in this way time for experts, who can thus dedicate themselves to other activities.

1.2 Context

The players involved are the Mercantour Park's experts, whose are focusing on the monitoring and safeguarding of the Mediterranean area. This park is one of the nine French national parks located in the Alpes-de-Haute-Provence and Alpes-Maritimes departments, and its aim is to protect the nature, the landscapes, and the biological diversity of the territory, promoting the discovery and the knowledge for its public[1].

This is the reason why the tool for the analysis and monitoring of the Mediterranean Sea area can be considered as a solid starting point to pursue Mercantour's mission.

[1]Park National du Mercantour, mercantour-parcnational.fr/fr/le-parc-national-du-mercantour/les-missions-du-parc

1.3 Users

The main outcome of the project is to help the Mercantour Park's experts with an efficient tool they can exploit for fish location and class extraction, performing large-scale operations on millions of images.

1.4 Scope qualification

Considering the dataset dimension, the project aims to use images representing the underwater ecosystem within the Mediterranean Sea. However, even though the software has been developed considering this scope, at the moment it is general purpose and can equally recognize fishes from different scenarios, allowing its use in other contexts.

For the technological aspect, the software created can identify, locate, and create a mask for fishes in different images classifying them in a unique "Fish" class. Nonetheless, this decision has been taken starting from the fact that the availability of annotated images was scarce. However, the model can be significantly improved and specialized as the experts start to contribute and to expand the model with a specific set of images.

Finally, it is mandatory for the model to be trained. To do so experts need to first annotate manually a considerable number of images to "teach" the model how to automatically annotate other unseen images and reach an acceptable level of accuracy and reliability.

2. Presentations of the solutions

The first query for the fish model is to infer the position of each fish in each image. This task is harder in the case of underwater images, where fishes are difficult to distinguish due to the blurriness of the images and the numerous overlappings between different fishes.

Object detection

The main goal of object detection is to allow an IT algorithm to separate all the instances of different objects in an image or video in different classes[2]. Hence, the algorithm is able to locate the instances of each class and mark them using a square rounding them called bounding box. For some situations this method does not have enough precision[3].

Instance segmentation

To solve this problem an instance segmentation algorithm predicting the class label and the pixel-specific instance mask can be used, rather than object detection.

Instance segmentation comprehend different approaches. According to the literature[4], it is possible to have:

a. Classification of mask proposals

   This technique involves the generation of mask proposals that are later classified in different classes[5] [6]. Selective search is a first approach following this technique: it detects bounding boxes and then performs semantic segmentation[7]. However, these family of techniques were replaced after Deep Learning became more popular with solutions like RCNN[8], Fast RCNN and Faster RCNN.

b. Detection followed by segmentation

---

[2] Yali Amit, Pedro Felzenszwalb 2020, "Object detection", ReserachGate, researchgate.net/publication/339792032_Object_Detection

[3] Dengsheng Zhang, Md. Monirul Islam, Guojun Lu, 2011, "A review on automatic image annotation techniques", ScienceDirect, sciencedirect.com/science/article/abs/pii/S0031320311002391

[4] Abdul Mueed Hafiz, Ghulam Mohiuddin Bhat, 2019, "A survey on instance segmentation: state of the art", Springer, link.springer.com/content/pdf/10.1007/s13735-020-00195-x.pdf

[5] Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, Arnold W. M. Smeulders, 2011, "Segmentation As Selective Search for Object Recognition", ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=vandeSandeICCV2011&bib=all.bib

[6] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T. Barron, Ferran Marques, Jitendra Malik, 2015, "Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation", Arxiv, arxiv.org/abs/1503.00848

[7] Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, Arnold W. M. Smeulders, 2011, "Segmentation As Selective Search for Object Recognition", https://ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=vandeSandeICCV2011&bib=all.bib

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, 2013, "Rich feature hierarchies for accurate object detection and semantic segmentation", Arxiv, arxiv.org/abs/1311.2524

This approach involves object detection using a bounding box followed by the segmentation of the object within the bounding box. Some examples of this approach are Mask R-CNN[9] and Faster R-CNN[10].

c. Labelling pixels followed by clustering
It performs categorical labelling of every image pixel, followed by a grouped process of the pixels into object instances using a clustering algorithm. Deep Watershed Transform and the Instance Cut are approaches following this methodology.

d. Dense Sliding Window Methods:
These methods use CNNs for generating the mask proposals by using dense sliding window techniques. A sliding window is a rectangular region with fixed width and height that moves across an image for detecting the exact position of each object[11]. TensorMask, DeepMask or InstanceFCN are examples of algorithms using this approach.

According to the requirements of the project, on every picture was required to perform both instance segmentation and object detection. Consequently, the chosen methodology was "detection followed by segmentation". For this family of solutions, several techniques were studied.

2.1 Fast R-CNN

Fast R-CNN[12] improves the training process difficulties presented in R-CNN by implementing a simultaneous learning of the SoftMax classifier and of the class specific bounding box regression (rather than individual as it was in R-CNN[13]). With this improvement the RCNN problems are overcome, but on the other hand, it presents other downsides, because of the usage of external region proposals[14].

2.2 Faster R-CNN

Faster R-CNN[15] proves that selective search, when implemented as fully connected layers, can be replaced by a CNN producing region proposals. In fact, fully connected layers have a worst ability in

9 Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, 2013, "Rich feature hierarchies for accurate object detection and semantic segmentation", Arxiv, https://arxiv.org/abs/1311.2524
10 Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", Arxiv, arxiv.org/abs/1506.01497
11 Dapeng Wang, Yubin Zhang, Zhang Zhang, Jiang Zhu, Jun Yu, 2010, "KaKs_Calculator 2.0: A Toolkit Incorporating Gamma-Series Methods and Sliding Window Strategies", ScienceDirect, sciencedirect.com/science/article/pii/S1672022910600083
12 R. Girshick, 2015, "Fast R-CNN", IEEE, ieeexplore.ieee.org/document/7410526
13 Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, 2018, "Path Aggregation Network for Instance Segmentation", Arxiv, arxiv.org/abs/1803.01534
14 Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, Xinggang Wang, 2019, "Mask scoring R-CNN", Arxiv, arxiv.org/abs/1903.00241
15 Jonathan Long, Evan Shelhamer, Trevor Darrell, 2014, "Fully Convolutional Networks for Semantic Segmentation", Arxiv, arxiv.org/abs/1411.4038

object localization compared to CNN. Moreover, Faster R-CNN is implemented using a branch for object bounding box recognition, and this feature has been exploited to improve the Mask R-CNN[16].

2.3 Mask R-CNN

It is an improvement of Faster R-CNN, Fast R-CNN and R-CNN models for instance segmentation. RCNN was developed integrating the AlexNet along with a region proposal and using the selective search technique[17]. Although it achieves high accuracy in object detection, it presents some downsides: slowness and difficulty for the multi-stage training process, slowness of the testing process (since different features from CNN have to be extracted for each object proposal).
Mask R-CNN introduces an improvement to overcome Faster R-CNN[18], by adding a new branch to the existing one for object detection in Faster R-CNN, to perform object mask prediction in parallel. This translates into a new model, offering multiple advantages.

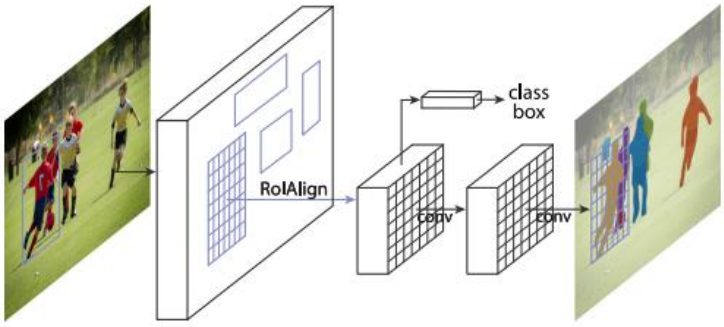


Figure 1: Mask R-CNN framework for instance segmentation, from [10]

2.4 Cascade

Cascade is an architecture in object detection, with improved performances in its tasks thanks to multi-stage architecture for object detection[19]. Nevertheless, when this approach is shifted to instance segmentation, it brings to a limited gain in the accuracy of the mask compared with the bounding box.

---

[16] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, Jitendra Malik, 2014, "Hypercolumns for Object Segmentation and Fine-grained Localization", Arxiv, arxiv.org/abs/1411.5752
[17] Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, Arnold W. M. Smeulders, 2011, "Segmentation As Selective Search for Object Recognition", ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=vandeSandeICCV2011&bib=all.bib
[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, 2017, "Mask R-CNN", Arxiv, arxiv.org/abs/1703.06870
[19] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin, 2019, "Hybrid Task Cascade for Instance Segmentation", Arxiv, arxiv.org/abs/1901.07518

2.5 HTC (Hybrid Task Cascade)

HTC is a framework for instance segmentation improving the cascade methods[20]. HTC was developed with the goal of overcoming the limited gain accuracy problem, improving the information flow by using a multitasking approach at each stage where the bounding box and the mask prediction are combined together and by introducing a fully convolution branch helping the distinction between hard foreground and cluttered foreground.

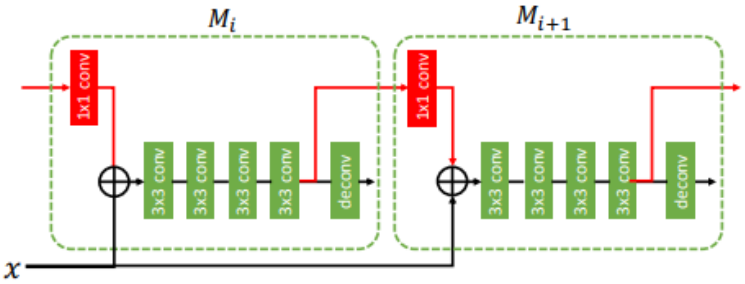


Figure 2: Architecture of multi-stage mask branches of Hybrid Task Cascade, from [13]

2.6 PANet

The PANet model focuses on the way the information propagates in deep neural networks[21]. It implements shorter information paths across the lower layers and the features at the top of the network, making more relevant information flow across the subnetworks generating the proposals. Furthermore, in the end of the network a branched segment captures some proposals view to improve the prediction of the generated masks.

As shown in Figure 3, the architecture is based in the following steps:

a) FPN backbone: path augmentation and aggregation is performed to improve performances
b) Bottom-up path augmentation: makes low-layer information easier to propagate
c) Adaptive feature pooling: allows each proposal to access information from all levels for prediction
d) Box branch: complementary path added to the mask-prediction branch
e) Fully connected fusion: made by a Fully Connected Network to extract the final mask

[20] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin, 2019, "Hybrid Task Cascade for Instance Segmentation", Arxiv, arxiv.org/abs/1901.07518

[21] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, 2018, "Path Aggregation Network for Instance Segmentation", Arxiv, arxiv.org/abs/1803.01534

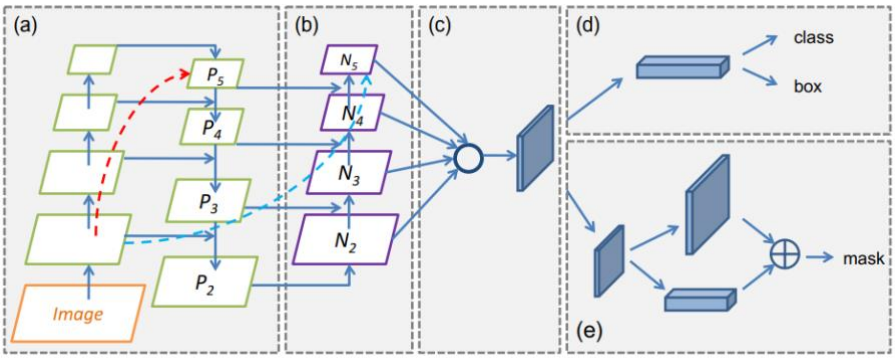


Figure 3: Architecture of PANet framework, from [14]

2.7 Mask Scoring R-CNN

Masc Scoring R-CNN was created to solve the problem presented by Mask R-CNN, which presents some loss of generality and in some situations it was not performing optimally[22] . The idea is to combine Mask R-CNN with an additional Mask-IoU module, learning the Mask-IoU aligned mask score because the quality of the mask with IoU gives very accurate results[23].

The arrangement created in MaskScoringCNN is used to predict the IoU between the input mask and the ground truth mask. The model contains a network block that learns the predicted mask quality combining the features of the instance with the predicted mask to regress the predicted mask IoU. Therefore, by analyzing and comparing each time the quality and score of the mask, the performance of the instance segmentation process is improved.

The architecture of the model is explained in Figure 4 where the input image is fed into a backbone network to generate Region of Interest (RoI) via Region Proposal Network (RPN) and RoI features via RoIAlign. Using the predicted mask and RoI features as inputs, the MaskIoU is predicted with 4 convolutional layers and 3 fully connected layers.

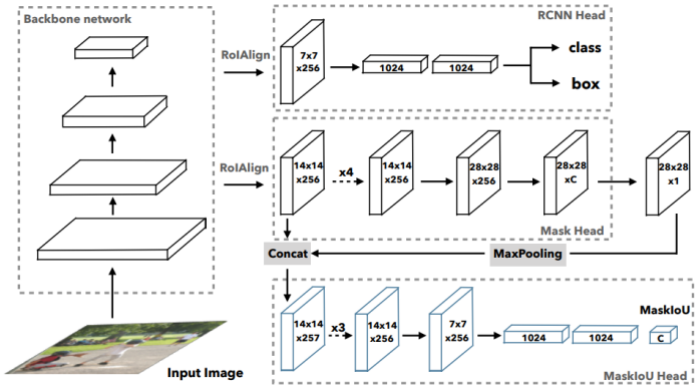


Figure 4: Architecture of Mask Scoring R-CNN, from [15]

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, 2017, "Mask R-CNN", Arxiv, arxiv.org/abs/1703.06870

[23] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, Xinggang Wang, 2019, "Mask scoring R-CNN", Arxiv, arxiv.org/abs/1903.00241

2.8 MPN (MultiPath Network)

MPN is the result of three modifications performed in the Fast R-CNN model[24]. First, by skipping certain connections, it provides the object detector with the capability of access to features of different layers what gives to the network a fast-learning process. Moreover, the second modification consisted in introducing an element exploiting the contexts of the objects to learn more specific features. Finally, it is introduced an integral loss function with the appropriate network adjustment to improve localization.

As a result, in the network created, the information can flow following multiple paths as features, multiple layers and from multiple object views. This improvements of the Fast R-CNN model yield in a better model in the COCO 2015 detection and segmentation challenges.

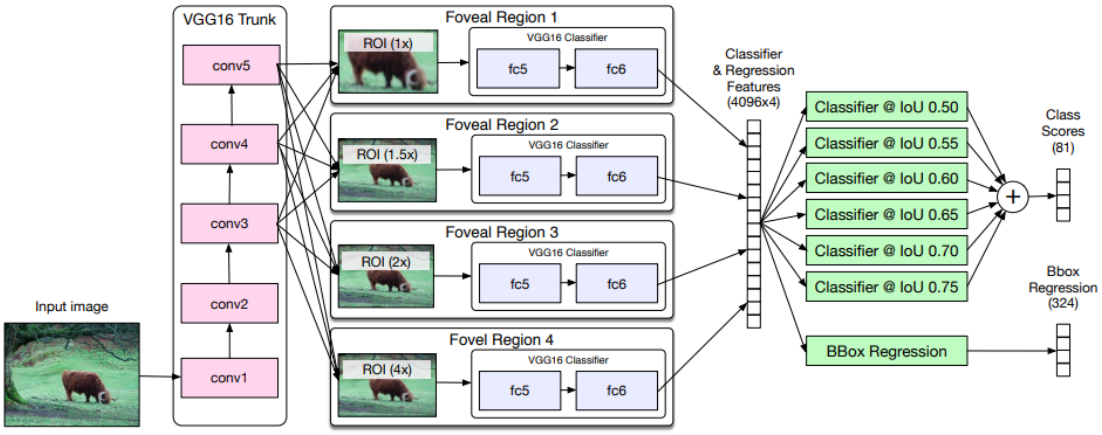


Figure 5: Architecture of MultiPath network, from [12]

2.9 YOLACT

YOLACT is an instance segmentation model focused on instance segmentation in real time[25]. It is the fastest real-time instance segmentation technique thanks to the way it is implemented. To achieve the results obtained it divides the image segmentation in two parallel subtasks:

1) Generation of the prototype masks
2) Prediction of the mask coefficients for each mask

Then, the instance masks are produced by linear combination of the prototype masks by using the coefficients of the masks. With this implementation, the network is able to learn localization.

---

[24] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O. Pinheiro, Sam Gross, Soumith Chintala, Piotr Dollár, 2016, "A MultiPath Network for Object Detection", Arxiv, arxiv.org/abs/1604.02135

[25] Daniel Bolya, Chong Zhou, Fanyi Xiao, Yong Jae Lee, 2019, "YOLACT: Real-time Instance Segmentation", Arxiv, arxiv.org/abs/1904.02689

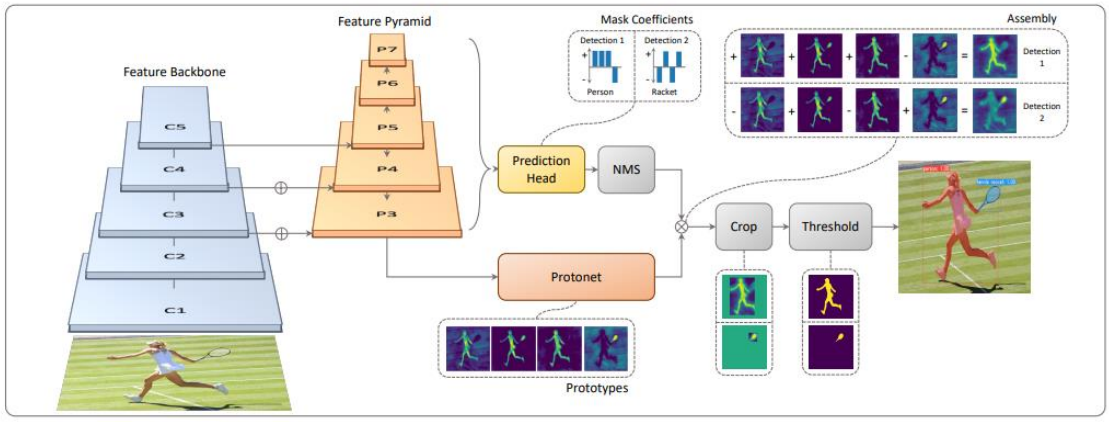


Figure 6: YOLACT Architecture, from [19]

3. Positioning of the solution in relation to the existing ones

According to the literature[26] and from the results shown in Figure 7 extracted from instance segmentation on the Microsoft COCO dataset[27], there are many models outperforming Mask R-CNN in terms of AP (TensorMask, MaskLab+, CenterMask, Mask Scoring R-CNN, Non-local Neural Networks, SOLO, BlendMask, GCNet, SOLOv2, PANet and HTC).

Method	Average precision (AP) (%)	Year
Hybrid Task Cascade (with extra training data)	43.9	2019
PANet	42.0	2018
SOLOv2	41.7	2020
GCNet	41.5	2019
BlendMask	41.3	2020
SOLO	40.4	2019
Non-local Neural Networks	40.3	2017
Mask Scoring R-CNN	39.6	2019
CenterMask	38.3	2019
MaskLab+	38.1	2017
TensorMask	37.3	2019
Mask R-CNN	37.1	2017
PolarMask	32.9	2019
YOLACT	29.8	2019
MultiPath Network	25.0	2016

Figure 7: Instance segmentation work performed on the Microsoft COCO dataset, from [23]

It is important to remark that the main task to be performed in the project is "detection followed by segmentation" approach and only the models "Mask Scoring R-CNN", "PANet", "Hybrid Task Cascade", and "Mask R-CNN" belong to this category.

The selected solution was the Mask R-CNN model because it best fits the project requirements. The main reason for its selection is the ease it provides to the training phase and its flexibility to be used for other tasks, like object detection and image segmentation. Moreover, despite being outperformed in terms of AP by different models, it gained the first position in all the 3 COCO suite 2016 challenges, in which it was considered instance segmentation and detection of bounding boxes. Finally, it is a well-known algorithm, and it is used in many implementations, making easier the development and the maintenance of the code.

---

[26] Abdul Mueed Hafiz, Ghulam Mohiuddin Bhat, 2020, "A Survey on Instance Segmentation: State of the art", Arxiv, arxiv.org/abs/2007.00047

[27] Lin T-Y, Maire M., Belongie S., Hays J, Perona P, Ramanan D, Dollar P, Zitnick L, 2014, "Microsoft COCO: Common Objects in Context", Springer, link.springer.com/chapter/10.1007/978-3-319-10602-1_48

Mask Scoring R-CNN vs Mask R-CNN

Mask Scoring R-CNN is a model based on the awareness of the quality of its own predictions by regressing the mask Intersection over Union (IoU) and prioritizing the most accurate predictions during the evaluation phase. However, the main downside presented by this model is the training time since the two tasks performed by Mask Scoring R-CNN are hard to train with a single objective function.
Moreover, the training presents a problem that Mask Scoring R-CNN model does not solve completely. The mask score learning task is decomposed as mask classification and MaskIoU regression and therefore, the problem arises when the Mask IoU needs to deal with Regions of Interest (RoI) that may contain multiple categories of objects. There are several solutions:

1) Learning the target category, while the others in the proposal are ignored
2) Learning the MaskIoU for all the categories
3) Learning the MaskIoU of all the categories appearing in the RoI

PANet vs Mask R-CNN

PANet focuses on improving the way the information propagates in the Neural Network to beat its competitors, especially Mask R-CNN, which has important points of improvement[28]:

1) There is a long path from low-level structure to topmost features, increasing the difficulty to access accurate localization information
2) Each proposal is predicted based on feature grids pooled from one features level, assigned heuristically so it can be updated since information discarded in other levels may be helpful for final prediction
3) The mask prediction is made on a single view, so it loses the possibility of gathering more diverse information.

With PANet is instead possible to:

1) Shorten information path by introducing a bottom-up path augmentation propagating low-level features to improve the features hierarchy
2) Introduce adaptive features pooling that aggregates features from all feature levels for each proposal
3) Augment the mask prediction with fully-connected layers that increase the information diversity and mask with better quality are created

Despite these solutions can improve the time of the training process, they will have a drastic impact in the performances, making the Mask R-CNN a better model to use for the project.

[28] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, 2018, "Path Aggregation Network for Instance Segmentation", Arxiv, arxiv.org/abs/1803.01534

4. Work done

All the code developed has been made available on a GitHub repository[29], which was used by the team to track the development of the project and can be used by other people as a guide to duplicate the work done.

As for every Data Science project, it is important to follow a well-defined and iterative workflow to produce a valuable final product. As shown in Figure 1, the process consists in various steps, which the team performed and adapted according to the requirements.

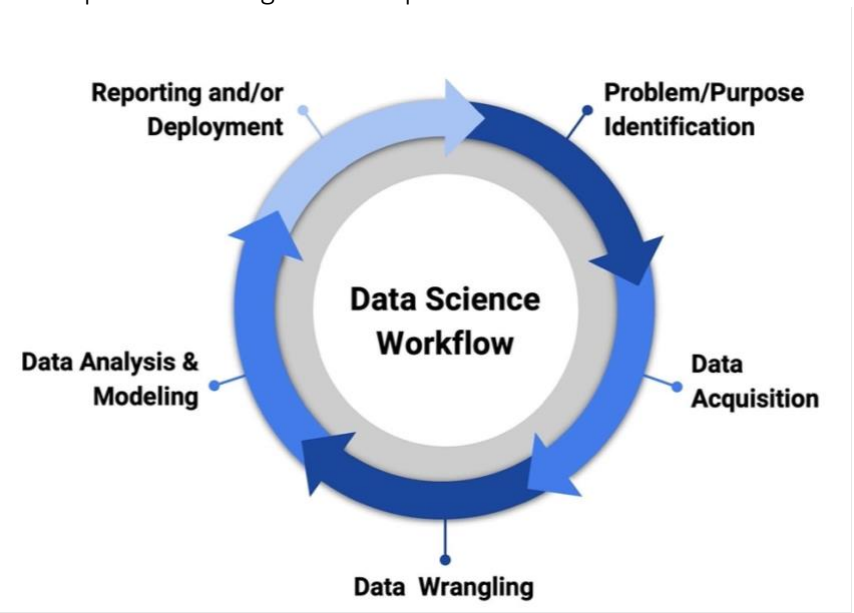


Figure 8: Data Science workflow[30]

Step 1: identifying the problem and the purpose

This first step is useful for guiding the entire project, informing how all the steps need to be executed. In this case, the commitment of the project concerns the ecological surveillance, the biodiversity monitoring, and the marine ecosystem analysis. However, this goal is extremely general and for this reason the team started from a first job regarding the monitoring of sea areas.

The project stakeholders have the necessity to detect fishes in the marine environment in an automatic and fast way. From here the main idea of the project: exploiting Deep Learning and its potential to simplify and speed the work of Mercantour biologists. Basically, it is needed to develop a software through which experts can start annotating fish images and train Deep Learning models over them, in order to perform object detection, instance segmentation, object tracking and counting.

[29] GitHub, github.com/alessiodimonte/TER
[30] District Data Labs, districtdatalabs.com/the-algorithm-issue-9-data-science-workflow

Step 2: dataset acquisition

Once the problem statement was fixed, the source fueling the project, namely the dataset, must be selected. The first requirement given to the team was to use the "SeaCLEF 2017" dataset, a famous visual dataset for marine organisms[31]. Unfortunately, this dataset is no longer available and, after agreeing with the supervisors to use another dataset, the team started to work with the DeepFish one.

The DeepFish dataset is containing approximately 40 thousand images and, differently from other datasets, is capturing the complexity of underwater habitats, considering also blurry pictures[32].

Step 3: Megadetector adaptation

According to the Document of Work, the initial idea was to exploit Megadetector and use it to put bounding boxes automatically on the dataset. In fact, Megadetector is a model used to detect animals, people, and vehicles in camera trap images without classifying them, but just localizing them[33].

To achieve this, the team followed the instructions given in the GitHub page covering the previous work done by Fanny Simoes[34] and, after installing all the required dependencies, some tests have been performed:

1) Change the configuration of the model "pnmdetector.config" by trial and error with different values for SoftMax, batch size, steps, metric set, data augmentation, size images and optimizer

2) Change the classes and substitute the set containing "human", "chamois", "roe deer", "doe", "deer", "fox", "badger", "wolf", "hare", "dog", "wild boar", "bicycle", "ibex" to the single class "fish"

Unluckily, results in the test phase were not adequate for the task. The blurriness of images and the difficulty of recognizing the different fish shapes lead to unsuitable outcomes. As shown in Figure 2, in some cases the detection was performed correctly (or partially), while in other cases it was completely wrong or absent.

[31] ImageCLEF, imageclef.org/lifeclef/2017/sea
[32] Arxiv, arxiv.org/abs/2008.12603
[33] GitHub, github.com/microsoft/CameraTraps/blob/main/megadetector.md
[34] GitHub, github.com/FannySimoes/Projet_PNM

Figure 9: Megadetector results on the DeepFish dataset

Since it was requested to implement both object detection and instance segmentation, the team decided together with the supervisors to avoid the Megadetector part (which only returns bounding boxes) and focus on the study of different approaches to achieve the goal.

Step 4: User Interface for annotation

At the same time, three different User Interfaces were developed, according to the different needs that came up throughout the process. Nonetheless, according to the other phases, the third version of the UI was the best suit for the project needs.

1) The first UI was considering the input images and bounding boxes coming from the Megadetector
    a. This UI was developed during the Megadetector adaptation phase, while it was not yet taken into account abandoning its use
    b. This UI was written in Python using the PySimpleGUI library
    c. It provides a way for the users to specify in the bounding box of one image is correct or not and in the first case annotate the name of the fish within the bounding box. Moreover, after the annotations are finished, they are saved in a CSV file so that they can be loaded again in the UI or used for another task.

2) The second UI was considering the annotation format for object detection and bounding boxes

a. This UI was taken and adapted starting from the "LabelImg" found on GitHub[35]
b. The UI was written in Python and through it is possible to put new bounding boxes on images, correct and delete them, choose the input images directory and the output directory where to store annotations
c. The format of annotations is YOLO, meaning that it creates a .txt file where for each box identifying an object a new line is created in the format "<object-class> <x> <y> <width> <height>"

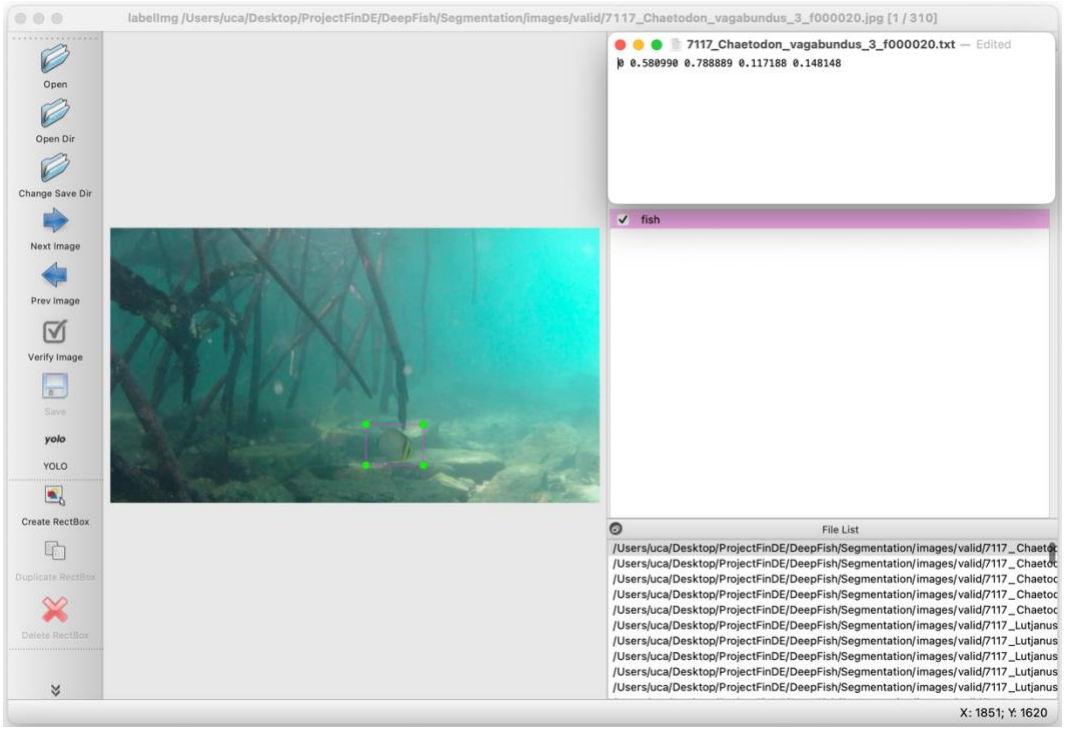


Figure 10: LabelImg User Interface

3) The third UI was considering the annotation format for instance segmentation and segmentation masks
    a. This UI was taken and adapted starting from the "MakeSense" found on GitHub[36]
    b. This UI was written in TypeScript and requires node.js version 12.x.x to work correctly
    c. The UI supports all the annotations formats, but it was adapted to the annotation format COCO JSON for segmentation
    d. Installation process:
        1. Install "node.js" version 12.x.x (https://nodejs.org/it/download/)

[35] GitHub, github.com/tzutalin/labelImg
[36] GitHub, github.com/SkalskiP/make-sense

2. Download the modified version from our GitHub repository
3. Open the terminal and enter in the just downloaded folder (e.g., if the folder is downloaded on the Desktop it is needed to write “cd Desktop/make-sense”)
4. From the terminal write “npm install” and wait for the installation
5. From the terminal write “npm start” and wait for the UI to be opened
6. Now, each time the UI is needed perform step 3 and 5 (the installation process of step 4 needs to be done just the first time)

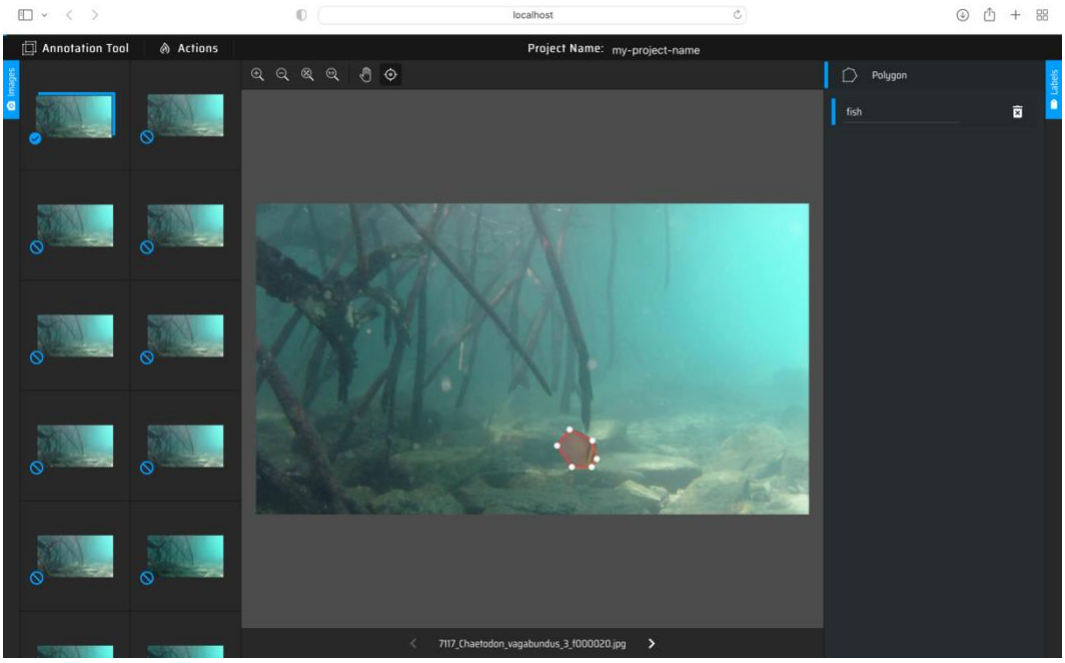


Figure 11: MakeSenseModified User Interface

Step 5: model research

In parallel to the development of the UI, to decide the best techniques to exploit for the task, the team started to study and test different methods involving object detection, instance segmentation and object tracking.

Step 5.1: Mask R-CNN on a custom dataset

The first option studied was to perform Mask-RCNN on a custom dataset. Mask-RCNN is an approach that efficiently detects objects in an image and simultaneously generates a segmentation mask for each instance[37].
In order to test it on the DeepFish dataset and see how well it performs a well-defined process was followed.

The first step was the creation of the annotations file for the selected 310 images, using the ready-to-use masks in the DeepFish dataset. The format compatible with Mask-RCNN, describing instance segmentation, is either "VGG JSON" or "COCO JSON". For the task, it was decided to use the "COCO JSON" format and consequently it was necessary to find a way to switch from a mask in black and white to a COCO JSON annotation file. To do so a script found on GitHub[38] was adapted according to the needs to return a "COCO JSON" file storing the coordinates of each of the 310 masks.

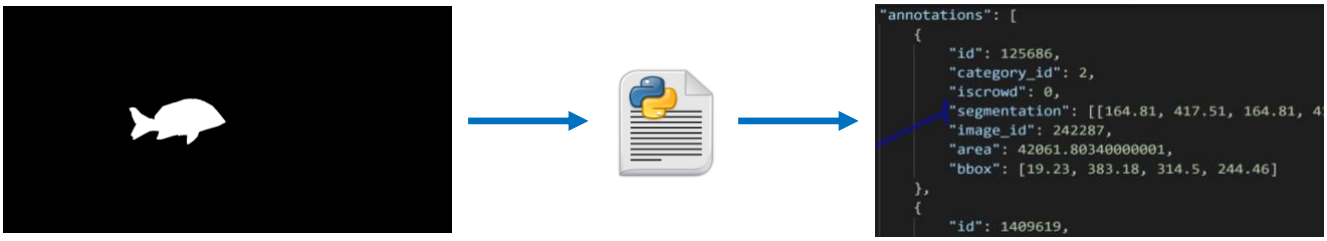


Figure 12: From Mask to COCO json process

After having the annotation file, it was possible to train the Mask-RCNN on the custom dataset. Following the instructions made available on GitHub for Mask-RCNN[39], the code was adapted as needed.

First of all, the model was not trained from scratch, but transfer learning was employed and, starting from the pre-fit Mask-RCNN model on the MS-COCO dataset, the model was tailored on the custom dataset by training only the heads for 5 epochs.
As shown in Figure 3, the results generally provide a correct detection and segmentation of fishes with a higher level of accuracy. Nonetheless, the model is not at its finest because it still fails in certain cases, detecting fishes in a wrong way.

Anyhow, considering the limited number of images taken into consideration for the training (only 310), it is sure that, incorporating in the training phase more images can significantly improve the precision of the detection and segmentation.

[37] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, 2017, "Mask R-CNN", Arxiv, arxiv.org/abs/1703.06870
[38] GitHub, github.com/chrise96/image-to-coco-json-converter
[39] GitHub, github.com/matterport/Mask_RCNN

Figure 13: Mask-RCNN on DeepFish dataset

Step 5.2: object tracking with OpenCV

Even though a dataset containing videos on which we could test object tracking was not available, the team analyzed the DeepFish dataset and found that some images were temporally taken one after the other. Thus, multiple frames were put them together to create a video and try to test this first approach on the DeepFish dataset.

In this first methodology only OpenCV was used, which has useful built-in functions for detection and tracking. For detection a naïve technique based on image processing was used, while for tracking it was utilized a mathematical function measuring the distance between centroids in two subsequent frames.

First, the "object_detector" was created with the "createBackgroundSubtractorMOG2()" function, which creates a mask in black and white for the moving elements. This is one kind of background subtraction methods offered by OpenCV[40].

After that, boxes around the detected objects were drawn, assigning a unique ID to each box, and the "EuclideanDistanceTracker()" was employed to perform object tracking.

Results from this technique are very poor, as shown in Figure 4. This is mainly because of the input video, which is hard to follow even for a human eye due to the large distance between fishes in two subsequent frames.

[40] OpenCV, docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html

Figure 14: Object tracking with OpenCV on DeepFish dataset

Step 5.3: object tracking with DeepSORT

The second approach studied for object tracking was DeepSORT. Unluckily, it was not not possible to test it on the DeepFish dataset because of the dataset's nature, which has not video content. Nonetheless, for personal interest the team tried it on a sample video, and results showed a significant potential for this algorithm if applied in the project context, as shown in Figure 5 (video: https://youtu.be/XaObp35HTKU).

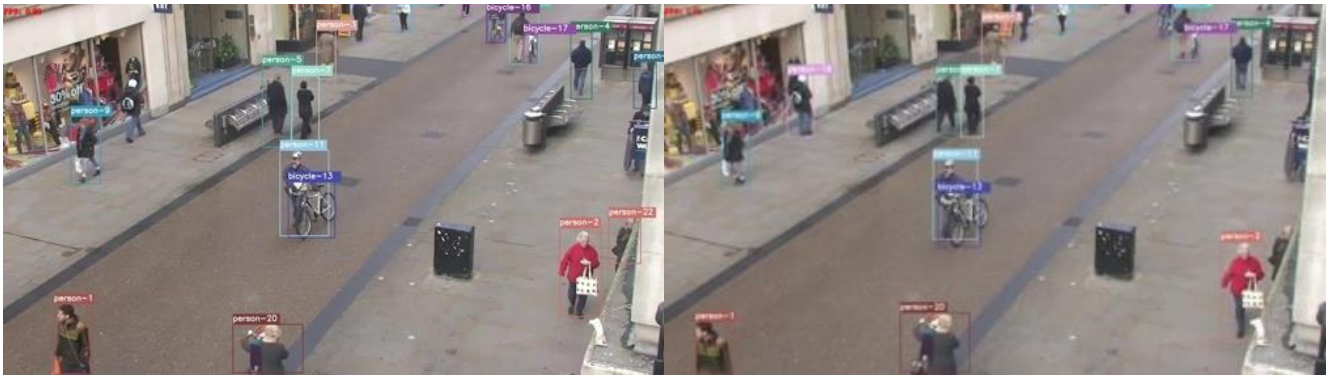


Figure 15: DeepSORT on a video sample

Step 6: project refinement

Together with the supervisors and stakeholders, the team decided to give to the project a different extent: avoid the tracking part because of the incompatible dataset and focus on enriching the user interface. The new UI should allow to start the Mask-RCNN training phase and see the results on new images directly from the User Interface.

Step 7: User Interface modification

In order to satisfy the new requirements, some adjustments to our UI needed to be performed.

Step 7.1: “Start training” button

The first change was the addition of a clickable button to start the training phase. Moreover, after the training, the model can do predictions on new images and, directly from the UI, experts have the opportunity to check and correct the predicted masks, in order to constantly ameliorate the quality of the training dataset.

This functionality was conceived such that:

1. From the UI is it possible to annotate images
2. Click “start training” and pass annotations and images to the Mask-RCNN model
3. After the training part ends, the test phase returns new images never seen by the model and the related annotations
4. From the UI it is possible to correct the annotations just made by the test phase
5. Start the training phase with the training dataset enriched by the images and annotations coming from the step 3 and 4
6. Repeat step 2, 3, 4, 5

Step 7.2: Flask

To run the training phase from the UI the team came up with the solution provided by Flask[41], which through a simple python code can start the Mask-RCNN script after clicking the button “Start training”.

Flask allows to create web-based application and performs calls to executables. In this specific case the executable to call is the script for the Mask-RCNN training and the code to do so is shown in Figure 6

diocane.py - /Users/uca/Desktop/make-sense-develop/diocane.py (3.9.7)

```
from flask import Flask
import json
import os

api = Flask(__name__)

@api.route('/start_training', methods=['GET'])
def startTraining():
    os.system("python DeepFish_MaskRCNN.py")
    response = json.dumps({'success':True}), 200, {'Access-Control-Allow-Origin' : '*'}
    return response


if __name__ == "__main__":
    api.run()
```

Ln: 9 Col: 21

Figure 16: Flask code

[41] Flask, flask.palletsprojects.com/en/2.0.x/

To make the button work correctly it is fundamental to run this Flask python script before clicking the button “Start training” on the UI. It was necessary need to pass through Flask because for safety reasons there is no way to execute a program directly from another application and thus to respect these requirements the following commands must be executed:

1. Open the terminal and enter in the location of the folder of the downloaded UI (same as step 4.3: User Interface for annotation)
2. Write the command “python run_flask.py”
3. Open the UI and click “Actions” and then the “Start training” button

5. Conclusions

As described, the project evolved throughout the process, due to the requirements imposed and the difficulties faced. In the Document of Work, it was explained the initial solution through a use-case diagram (see Annex 1), but the final product offered is a User Interface including all the functionalities.

As shown in Figure 6, the UI plays a central role in all the processes: through the UI is in fact possible to:

1. Import images: these images are the ones that experts need to annotate from scratch or to check/correct in the case the annotations are provided by the Mask-RCNN model
2. Import annotations: the Mask-RCNN model produces an annotation file in the format of "COCO JSON" for the masks related to the test images
3. Create/modify/export annotations: after creating from scratch or correcting the annotations, experts need to export the annotations in a "COCO JSON" format to pass it to the Mask-RCNN model for the training phase
4. Launch training phase: when experts finish to annotate/correct annotations they can start the training phase with the set of images and annotations they just created

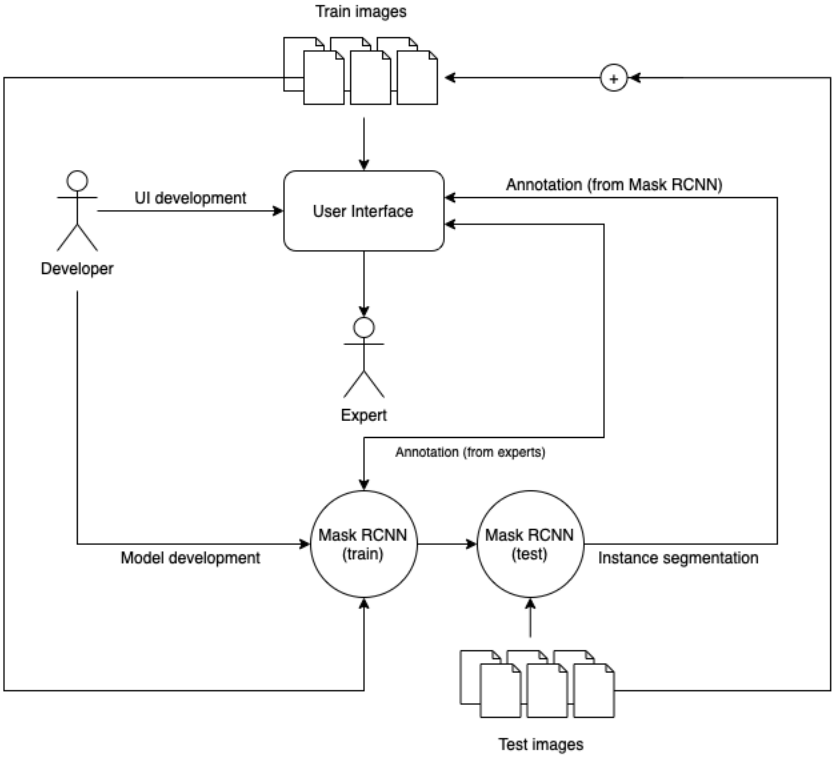


Figure 17: use case diagram

With respect to the initial goal, it was not implemented in the final solution the object tracking and the counting parts because of the dataset unsuitability and the new project constraints. Nonetheless, a study

on how to perform these tasks was made, as shown in the previous chapters, to leave them as a basis for future developments.

Further extensions of the project are numerous. First is the addition to the model of multi-class classification in order to train the model and recognize different species of fishes. Actually, the team decided to train the model to distinguish fishes because of the amount of annotated data and because of the preference to focus on the detection rather than classification.

The second functionality to the project is the object tracking and counting parts: starting from a different dataset which includes videos, train the DeepSORT model on it.

6. References

[1] Yali Amit, Pedro Felzenszwalb 2020, "Object detection", ReserachGate, researchgate.net/publication/339792032_Object_Detection

[2] Dengsheng Zhang, Md. Monirul Islam, Guojun Lu, 2011, "A review on automatic image annotation techniques", ScienceDirect, https://www.sciencedirect.com/science/article/abs/pii/S0031320311002391

[3] Abdul Mueed Hafiz, Ghulam Mohiuddin Bhat, 2020, "A survey on instance segmentation: state of the art", Springer, https://link.springer.com/article/10.1007/s13735-020-00195-x

[4] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin, 2019, "Hybrid Task Cascade for Instance Segmentation", Arxiv, https://arxiv.org/abs/1901.07518

[5] Abdul Mueed Hafiz, Ghulam Mohiuddin Bhat, 2019, "A survey on instance segmentation: state of the art", Springer, https://link.springer.com/content/pdf/10.1007/s13735-020-00195-x.pdf

[6] Dapeng Wang, Yubin Zhang, Zhang Zhang, Jiang Zhu, Jun Yu, 2010, "KaKs_Calculator 2.0: A Toolkit Incorporating Gamma-Series Methods and Sliding Window Strategies", ScienceDirect, https://www.sciencedirect.com/science/article/pii/S1672022910600083

[7] Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, Arnold W. M. Smeulders, 2011, "Segmentation As Selective Search for Object Recognition", https://ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=vandeSandeICCV2011&bib=all.bib

[8] Jordi Pont-Tuset, Pablo Arbelaez, Jonathan T. Barron, Ferran Marques, Jitendra Malik, 2015, "Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation", Arxiv, https://arxiv.org/abs/1503.00848

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, 2013, "Rich feature hierarchies for accurate object detection and semantic segmentation", Arxiv, https://arxiv.org/abs/1311.2524

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, 2017, "Mask R-CNN", Arxiv, https://arxiv.org/abs/1703.06870

[11] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, Yichen Wei, 2016, "Fully Convolutional Instance-aware Semantic Segmentation", https://arxiv.org/abs/1611.07709

[12] Sergey Zagoruyko, Adam Lerer, Tsung-Yi Lin, Pedro O. Pinheiro, Sam Gross, Soumith Chintala, Piotr Dollár, 2016, "A MultiPath Network for Object Detection", Arxiv, https://arxiv.org/abs/1604.02135

[13] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin, 2019, "Hybrid Task Cascade for Instance Segmentation", Arxiv, https://arxiv.org/abs/1901.07518

[14] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, Jiaya Jia, 2018, “Path Aggregation Network for Instance Segmentation”, Arxiv, https://arxiv.org/abs/1803.01534

[15] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, Xinggang Wang, 2019, “Mask scoring R-CNN”, Arxiv, https://arxiv.org/abs/1903.00241

[16] Jonathan Long, Evan Shelhamer, Trevor Darrell, 2014, “Fully Convolutional Networks for Semantic Segmentation”, Arxiv, https://arxiv.org/abs/1411.4038

[17] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, Jitendra Malik, 2014, “Hypercolumns for Object Segmentation and Fine-grained Localization”, Arxiv, https://arxiv.org/abs/1411.5752

[18] Sean Bell, C. Lawrence Zitnick, Kavita Bala, Ross Girshick, 2015, “Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks”, Arxiv, https://arxiv.org/abs/1512.04143

[19] Daniel Bolya, Chong Zhou, Fanyi Xiao, Yong Jae Lee, 2019, “YOLACT: Real-time Instance Segmentation”, Arxiv, https://arxiv.org/abs/1904.02689

[20] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, Arxiv, https://arxiv.org/abs/1506.01497

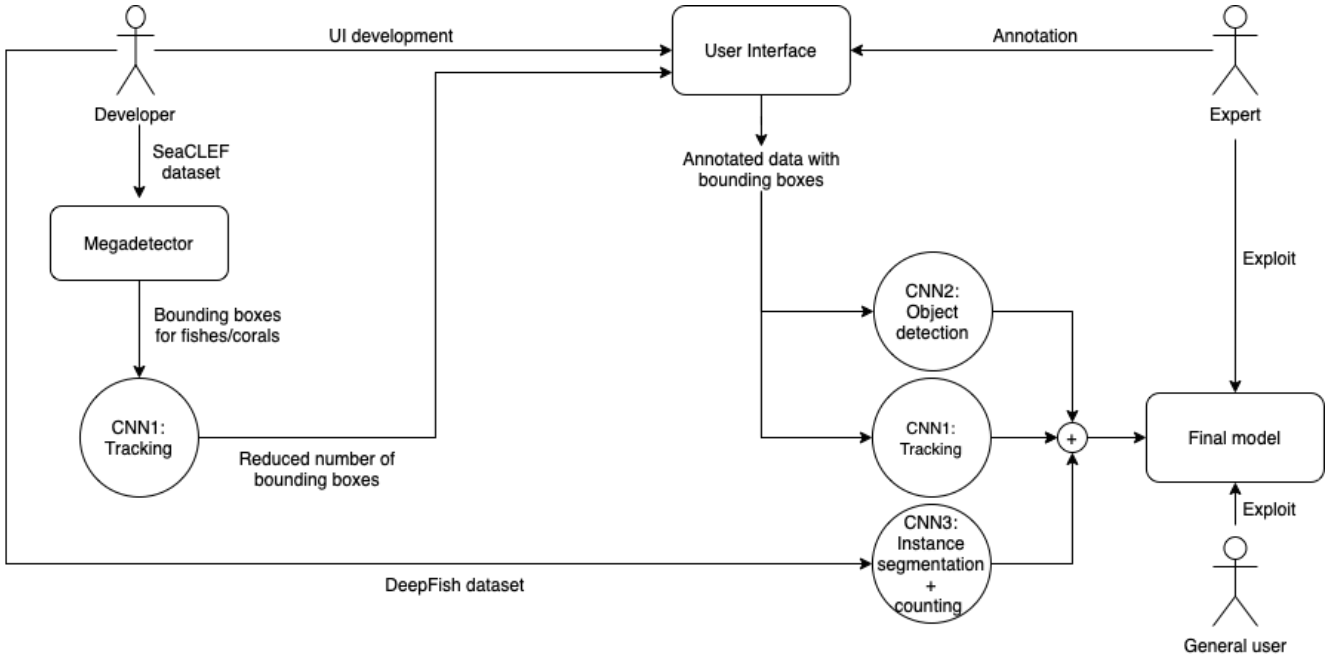
[21] Park National du Mercantour, http://www.mercantour-parcnational.fr/fr/le-parc-national-du-mercantour/les-missions-du-parc

[22] R. Girshick, 2015, “Fast R-CNN", IEEE, https://ieeexplore.ieee.org/document/7410526

[23] Abdul Mueed Hafiz, Ghulam Mohiuddin Bhat, 2020, “A Survey on Instance Segmentation: State of the art”, Arxiv, https://arxiv.org/abs/2007.00047

[24] Lin T-Y, Maire M., Belongie S., Hays J, Perona P, Ramanan D, Dollar P, Zitnick L, 2014, “Microsoft COCO: Common Objects in Context”, Springer, https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48

7. Annex



Annex 1: use-case diagram presented in the DoW