Boat Detection

Miroljub Mihailovic, 2021445

July 2021

1 Introduction

This report explains how and why some choices are implemented in such a way to recognize boats found on a given image. To achieve the described goal, I have implemented a neural network approach, which I will present in the following chapters. The python part for NN is made on Google Colab since it faster way that I have found to train a model, here is the link of the training procedure: Model

2 Neural Network

All the procedures I am going to describe took more or less 10 days to fine-tune the network parameters and increase the dataset samples.

2.1 Dataset generation

The dataset that I realized required the following steps:

- Load the images provided, the number of elements chosen are approximately 1400 (link: train)
- Label each image, for this task I used the online application *Labelbox* (`https://labelbox.com/`) from which I was able to obtain a JSON file with the respective boundary boxes.
- Now we have to develop the dataset, it is based on *SelectiveSearch* algorithm, I described the procedure on Colab file.
- At the end I obtain two classes: *boat* and *no_boat*, the first has 3000 elements and the second approximately, 12000.

2.2 Model training

I have implemented a pre-trained neural network, since it is more performing than a model built from the beginning. I tried the $VGG16$, $MobileNetV2$ and $Xception$ models and from the results obtained $MobileNetV2$ is clearly faster

but is less performing than others mentioned. Therefore, I train the net using the *Adam* algorithm because it generalized better from the other implementation that I tested. The outcome is reported below, where the plot highlights the progress of training and validation.

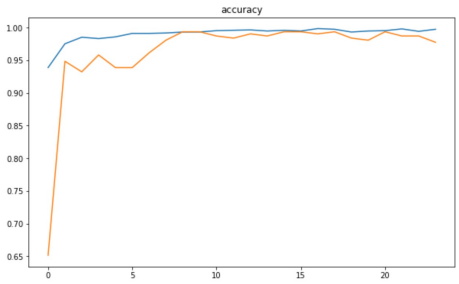


Figure 1: accuracy plot

3 Selective Search

Selective search algorithm consist of finding regions that can be hypothetical boat, the decision is made by the neural network. I have tried several secondary implementations of the original, for example *SingleStrategy*, but the best option obtained, in terms of speed and quality of computation, is $SelectiveSearchFast$. Applying the preprocessing phase, that is presented in the following section, the final result is improved in an exponential way.

4 Preprocessing phase

A preprocessing phase is a sequence of instructions in which the goals are: firstly, improving the quality of the input image and the second is to reduce the number of proposed regions. This procedure permits me to reduce the time for *SelectiveSearch* phase and therefore to analyze an amount of images (using the NN) that are exponential lower compared to a processing image without a preprocessing stage. Applying the $GaussinaFilter$ for reducing the noise and then changing the background from white to black, since that will help later to extract better results during the use of $DistanceTransform$ procedure. Therefore, I sharp the image implementing the *Laplacian* filter and then binarized the input. The last step consists of accomplishing the $DistanceTransform$. As it could be seen on the following images, the red rectangles represent the regions found by the *SelectiveSearch* with and without processing.

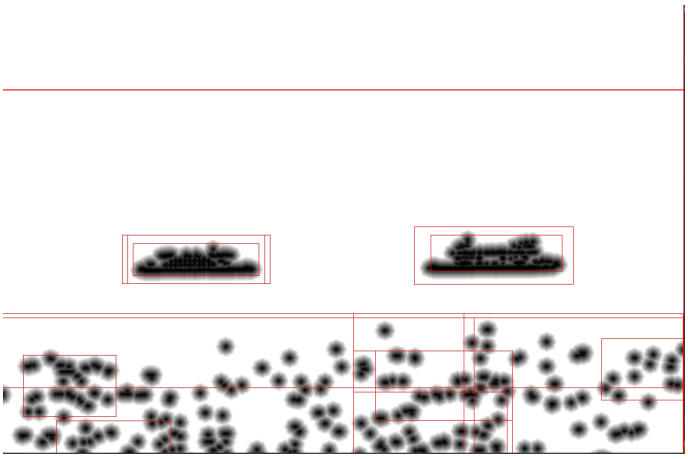


Figure 2: *SelectiveSearch* with preprocessing (20 regions found)

Figure 3: *SelectiveSearch* without preprocessing (303 regions found)

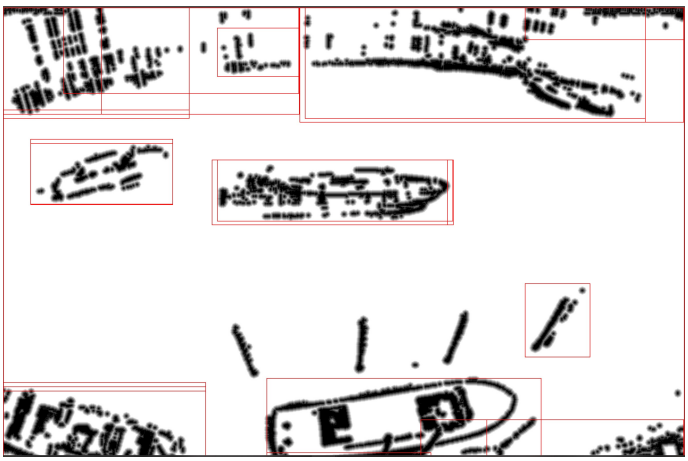


Figure 4: *SelectiveSearch* with preprocessing (26 regions found)

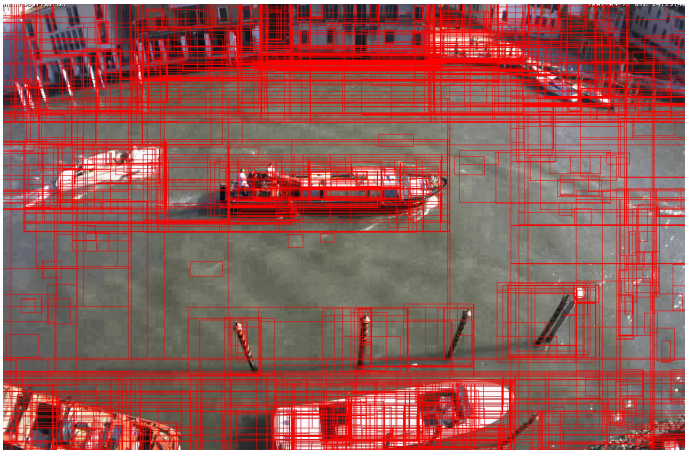


Figure 5: *SelectiveSearch* without preprocessing (2500 regions found)

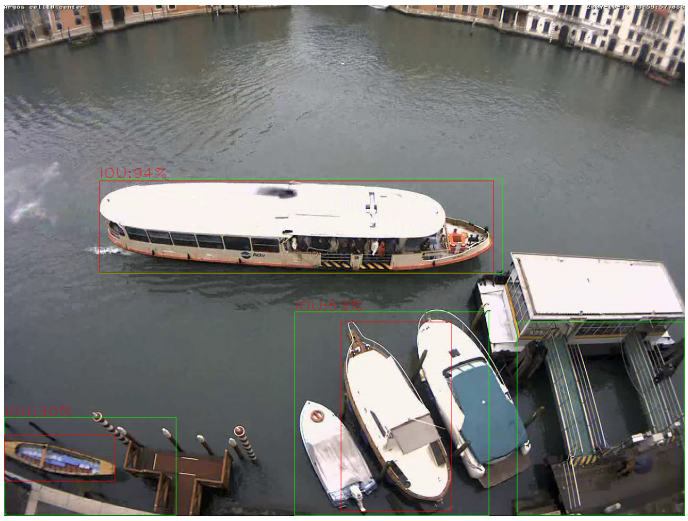
5 Detection

The detection part is the main phase inside the project in which, given in input an image, the system have to find boats and then drawn around them the boundary boxes. The steps required to achieve the goal are the follows:

1. A preprocessing phase, explained previously.
2. *SelectiveSearch* for finding the hypothetical boats.
3. Analyze each region found through Neural Network.
4. NMS (Non-Maximum Suppression) stage in which there is a grouping into only one rectangular, the same boat found.

6 Results

I report some results found with the application, the other outputs are present in the folder with the source code.


IOU:94%


IOU:36%


IOU:85%
IOU:81%


IOU:71%