***FINAL PROJECT – BOAT DETECTION***

In this report I will present my approach to the boat detection problem and the obtained results; for compiling instruction and all other technical information regarding the execution of the code, please refer to the README file provided with the source files. The only thing I would like to highlight is that, when running the code, you can decide either to execute all the code and retrain from scratch the model, or just test the saved and already trained one: this is provided for convenience, since retraining the entire model and testing it, will take a considerable amount of time (about 20 minutes in my PC).

My approach is based on Histogram of Gradients features and Support Vector Machines classifier. The task is quite difficult, since there are lots of problems in the images, like reflections, water wags, different scales and orientations for the ships, in addition to many different possible types and shapes of boats. Moreover, boats can appear at different distanced, and may be overlapping each other.

Considering also that the two test datasets are really different one from each other, all the choices regarding the various parameters are a compromise between the results in each of them: this means that, in general, a parameter value which led to good results on Venice dataset, gave bad one on the Kaggle images, and viceversa. So, I had to accept a tradeoff, in order to get quite satisfactory output on both the datasets.

Now, I will describe in details the procedure I adopted, and also some choices I made.

First of all, I decided to use classical computer vision technique to carry out the project, avoiding deep learning techniques, even if models like Convolutional Neural Networks or Single-Shot Detector are the state of the art in this field.

So, reading some papers, I found out that the combination of HoG features and SVM classifier gave really good results in object detection, also with different types of objects.

The first thing I did was preparing the datasets: training an SVM requires a set of positive images, and a set of negative ones. The set of positive samples have been obtained by cropping the Kaggle and Mar datasets’ images in a way so that only the boats remained, with the minimum possible amount of water or general background. The set of negative images has been obtained by taking parts of the images that were not boats and that were misclassified by the classifier, for example windows, bridges, water, and so on. Moreover, into the negative set I put the “buoy” category images from the Kaggle dataset, the “water” category images from the MAR dataset, and other negative samples provided and classified as false positive by the MAR dataset creators. In total I used about 4200 positive images and 8400 negative ones.

Then, I applied a bit of pre-processing to the obtained images:

* resizing all of them so as to be of the same dimension, as required by the Support Vector Machine model. After a lot of tries, the size has been set to 135x90, since those are the values with the best tradeoff between good performance of Venice dataset and good performance on Kaggle one.
* denoising, performed using a technique called “Non-Local Means Denoising”. This method considers a small window in the image, in searches in a small neighbourhood for similar patches, and the it replaces the pixel values with the average value computed from this set of similar prts of the image: in this way, I can gent better results with respect to simpler denoising techniques, like Gaussian blur or similar ones. Indeed, these methods are not so robust against camera and scene motions, and since most of the images in the training set were taken using moving cameras or, in any case, during scene motions like waves caused by boat movements, I think this approach is better than a simple Gaussian smoothing. The drawback of this method is the considerable computational time required, but it is still acceptable, considering the number of images to process.

The third step is the extraction of Histogram of Gradients feature from both the positive and negative training sets. The extracted gradient, moreover, need to be converted into a format that is accepted as training data by the OpenCV machine learning implementation.

After this, a SVM classifier for regression task is created using default values for parameters, and then it is trained using the data computed before. By default, the model is trained two times: the second one is a kind of hard-negative mining, since the svm is trained on negative samples.

The next phase consists in testing the trained detector into the test datasets: from the detection process, performed using detectMultiScale method of the HoGDescriptor class, I get the predicted bounding boxes, together with the relative confidence scores.

These data are used to perform the postprocessing phase, that aims to reduce the number of overlapping boxes and to evaluate the performance of the detector.

The number of overlapping boxes is reduced by using a Non-Maxima Suppression algorithm, based on confidence scores of each rectangle. The predicted bounding boxes are sorted by increasing confidence score and, at each iteration, the overlapping between the highest confidence rectangle and the other predicted boxes is computed using the IoU metric (for a deeper explanation of this concept, refer to the next section, regarding the performance evaluation). If the overlapping is above a threshold, the predicted box is suppressed (same problem as before, this value causes different behaviour in the various images, including some wrong suppression sometimes).

After this, we have the last step, that is the performance evaluation by means of the Intersection over Union metric. This value is defined between a pair of bounding boxes, and it is computes ad the intersection area of the two rectangles, divided by their union area. It is a commonly used metric in detection tasks, and it can be easily extended to other geometrical shapes. In this project, the IoU is computed between each predicted box and each ground truth box, and only the maximum iou for each detected box is kept. As ground-truth for the test images, I used the txt files provided in the forum: there’s one file for each image and, in each file, a row for every real boat. The bounding box of each ship is store as a 4-tuple composed by (x coordinate of top left corner; x coordinate of bottom right corner; y coordinate of top left corner; y coordinate of bottom right corner). The .txt files are processed in order to obtain a vector, storing the above mentioned coordinates in a easier-to-handle way in OpenCV.

In the following images, you can see the results obtained by using this approach in the images belonging to the two assigned datasets:

/\*\*

Images

\*//

As you can easily see, the technique consisting in HoG feature combined with a SVM classifier is not so good as expected, applied to this particular task: there are a lot of false positives, even if many examples of water or buildings are used to train the model, and a considerable amount of boats is not detected. Moreovere, the IoU values are quite bad, considering that a good detection would have an IoU>0.5: this can be caused either by an error in the implementation of the evaluation function, or by an high rate of false positives.  
I think these bad results are due to the particular characteristics of the two used datasets, which are really different one from the other. In Kaggle images, we have clean and easy to identify boats, in most cases just one per image, or more than one but clearly distinguishable one from the other and from the background. Moreover, most of the test images include lateral views of the ships, and quite big sizes for them. In the Venice dataset, instead, we have a lot of issues caused by buildings, bridges, wooden poles, reflects, ripples and waves, we have always many boats in each image, and often ships are really close one to each other, or even partially occluded. There are ships of a lot of different sizes, and the images are taken using camera which are in an upper position with respect to the boats: hence, we have lot of upper views of the ships.

All these differences in size, shape, environment and so on, cause also a big difference in the extracted features and have a negative impact on the potential results of the proposed method: for example, the choice of the size of the window to perform detection is not easy at all. I tried both with square windows and rectangular ones, and with multiple sizes for both cases, but I was not able to find a value which performed good in a sufficient amount of images. So, the actual parameters used in the project are the result of many tries and of a trade-off, decided in order to have satisfactory results in at least some images of each dataset.

In conclusion, I think that this approach is not really suitable for this type of problem, maybe a region-based method, using some segmentation techniques to identify regions of interest, combined with a classification model, like SVM, or a neural network, would have lead to better results.

REFERENCES

1. <https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/>
2. <https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/>
3. <https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/>
4. <https://docs.opencv.org/4.5.1/d0/df8/samples_2cpp_2train_HOG_8cpp-example.html#a68>
5. <https://github.com/Nuzhny007/Non-Maximum-Suppression/blob/master/nms.h>
6. <https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/>
7. <http://www.diag.uniroma1.it//~labrococo/MAR/detection.htm>
8. https://www.kaggle.com/clorichel/boat-types-recognition/version/1