Split port surveliance using artificial intelligence and high performance computing

1st Sven Gotovac
Dep. for Electronics and Computing
University of Split, Faculty for Electrical Engineering, Mechanical Engineering and Naval Architecture
Split, Croatia
gotovac@fesb.hr

2nd Luka Matić
Dep. for Electronics and Computing
University of Split, Faculty for Electrical Engineering, Mechanical Engineering and Naval Architecture
Split, Croatia
luka.matic.00@fesb.hr

3rd Vladan Papić
Dep. for Electronics and Computing
University of Split, Faculty for Electrical Engineering, Mechanical Engineering and Naval Architecture
Split, Croatia
vpapic@fesb.hr

4th Dunja Božić-Štulić
Dep. for Electronics and Computing
University of Split, Faculty for Electrical Engineering, Mechanical Engineering and Naval Architecture
Split, Croatia
dgotovac@fesb.hr

5th Hrvoje Turić
Dep. of Polytechnics
University of Split, Faculty of Science
Split, Croatia
hrvoje.turic@pmfst.hr

6th Vladimir Pleština
Dep. of Polytechnics
University of Split, Faculty of Science
Duvno, Bosnia & Herzegovina
vladimir.plestina@pmfst.hr

Abstract—

Keywords—Smart City, Image processing, YOLO, HPC, Google Colab, Jupyter

I. Introduction

Smart Cities use digital technology for strengthening citizens' welfare and to improve their quality of life while at the same time taking care of the preservation of the environment and natural resources in accordance with the UN Sustainable Development Agenda. Although every smart city has practically the same mission, the way city government plans to accomplish this mission and the goals that it defines depends on the particularities of the city itself and its ecosystem.

Technology plays a key role to rethink the way smart city is organized. It helps to gather information, to deploy efficient solutions and policies and to enable new communication channels relying upon broadband and mobile technology, big data, cloud and HPC services, artificial intelligence and machine learning, sensors, hyperconnectivity, etc.

The city of Split is a typical Mediterranean city, tourist centre and the main port of Dalmatia. The port of Split is a transit canter for all Dalmatian islands. The beautiful beaches of Split attract many tourists. Marinas in Split and Kaštela Bay are convenient starting point for nautical tourism along the Dalmatian coast. All that results in heavy activities in the Split harbour and coast waters. The safety of maritime traffic within Split port as well as bathers on the beaches is of great importance. Here, the technology can help a lot. University of Split, Faculty of electrical engineering, mechanical engineering and naval architecture has designed and developed a prototype for Split harbour traffic monitoring based on image processing and artificial intelligence described in this paper.

II. Proposed solution

The main idea is to use one or more high-resolution camera that supports remote control of direction (panning and tilting) and zooming, so called PTZ camera. The camera could be installed on the fixed or mobile position, such as, for example, flying Unmanned Autonomous Vehicle (UAV). Cameras are connected to the remote centre via either 4G/5G wireless or broadband optical link, depending upon the placement capabilities. Required network bandwidth is approximately 20 Mbit/s for 4k video transmission per camera. Received video is processed and stored in the HPC/Data Centre and results are forwarded to the control centre. The operators in control centre can (according to the situation) rotate camera and zoom the object or situation of the interest and take certain actions. Figure 1. presents proposed solution for Split harbour monitoring.

For a proof of concept, solution involving fixed cameras and drones has been considered but eventually, simpler solution with only one fixed camera has been chosen. The number of possible locations for the camera placement have been considered: Katalinića brig, Banovina, Harbour Master's Office, Koteks skyscraper. Due to the infrastructure capabilities for the proof of concept location we have selected Koteks skyscraper were 4k, PTZ camera has been installed and connected via optical link.

Next step is to select and adapt adequate image/video processing framework as it is described in the next section.

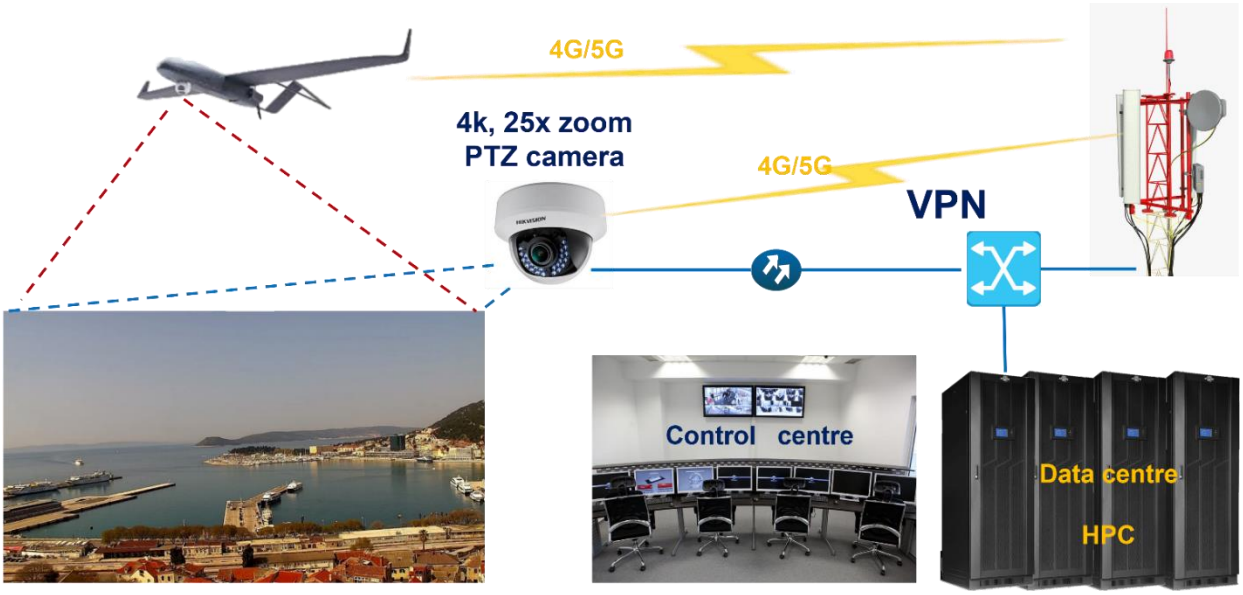


Figure 1. Proposed solution for Split harbour monitoring.

III. Object detection within video

Object detection is the process of determining areas in an image or video that have certain semantic meanings, and can be performed using a variety of methods. Each detected object of interest has a corresponding frame that delimits it and the class to which it belongs. Existing detectors can be divided into two categories: two-stage detectors and single-stage detectors [3]. The architecture of the network of single-stage and two-stage detectors is shown in Figure 2.

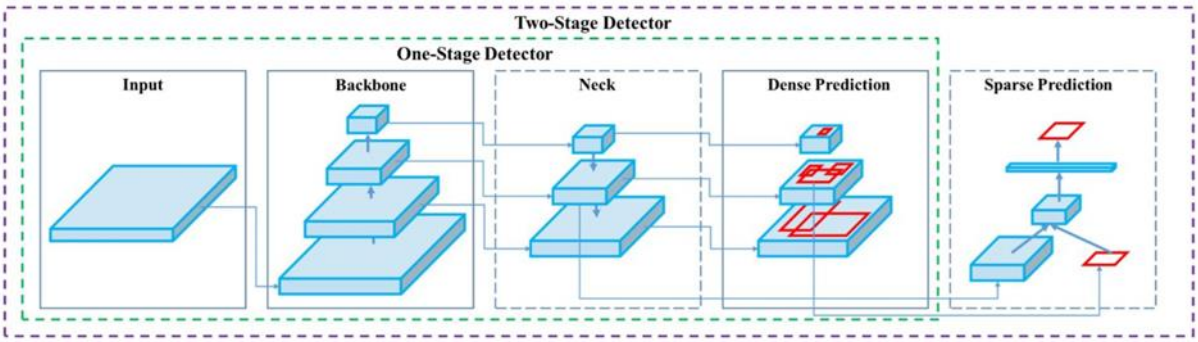


Figure 2. . Network architecture of single-stage and two-stage detectors [4], Backbone - part of the network that extracts the feature map from the input image, Neck - part of the network that collects features and combines them to prepare them for the detection step, Head - part of the network in charge of detection.

Two-stage detectors have high localization and object detection accuracy, while single-stage detectors achieve high inference speeds and are therefore more suitable for real-time detection. The speed of inference is the reason we chose to implement our solution using single stage detector.

Unlike two-stage detectors, which when detecting objects first select several suggested areas from the image, and then determine their categories and boundary frames, single-stage detectors detect objects directly from the input image, without the step of proposing regions. The most important representatives of single-stage detectors are: SSD (Single Shot Detector), DSSD (Deconvolutional Single Shot Detector), RetinaNet, YOLO (You Only Look Once). Of all the above detectors, Yolo shows the best performance, so as such it has been chosen as a solution for our problem.

The YOLO algorithm combines object detection components into a single convolutional network that can perform all the necessary tasks in a single pass [5]. This is one of the reasons that this method is fast and is often used in real-time applications. Features from the entire image are used to predict the boundary frame. In other words, the network globally considers the whole image and all the objects in the image. The input image is divided into S x S parts (cells), and passed through a convolutional neural network to extract features. Only one object is predicted for each cell, so one cell cannot predict multiple objects. After that, a linear regression process is performed, which for each location generates B boundary frames that should contain the object. Each boundary frame is defined with 5 values: x, y, w, h and confidence score, defined by equation (1):

$$Confidence\ score = probability * IoU \tag{1}$$

where IoU, Intersection over Union, is a measure of the overlap of the predicted frame and the actual (correct) ground truth frame:

$$IoU = \frac{predicted\ frame \cap ground\ truth\ frame}{predicted\ frame \cup ground\ truth\ frame} \tag{2}$$

and it is with a range [0,1]. The x, y coordinates represent the centre of the frame relative to the cell to which it belongs, and w and h are the width and height of the frame. A measure of reliability indicates the presence or absence of an object within the boundary frame. It depends on the probability that the proposed frame contains the object and the IoU that defines ratio between the predicted and the actual boundary frame. The confidence measure is equal to zero if the boundary frame does not contain an object. The aim is to ensure that the reliability measure is equal to the IoU measure. If the center of the object is not inside a cell, then that cell is not in charge for predicting the category. After all the predicted frames have been obtained, the best one has to be selected, using non-maximal suppression.

The first version of the YOLO network consists of 24 convolutional layers and 2 fully connected layers. Next versions of the YOLO introduced enhancements to further improve inference performance.

Some optimizations and architecture changes have been made to the YOLOv2 model, such as the use of batch normalization and high-resolution input images. Like the Faster R-CNN, the YOLOv2 model uses anchor boxes, predefined boundary frames of useful shapes and sizes that are customized during training. Instead of directly predicting position and size, shifts and reshapes of predefined anchor frames relative to the network of cells are predicted. In YOLOv2, the network structure was changed, using 19 layers, and then another 11 layers specifically designed to detect objects.

YOLOv3 introduced further improvements. The logistic regression substituted softmax function and was introduced to predict the objectivity of the results for each boundary frame. In addition, significantly larger feature extraction networks with 53 convolutional layers are used respectively Darknet-53 replaces Darknet-19.

YOLOv4 was introduced after years of cumulative improvements of YOLOv3, taking advantages of recent advances in deep learning algorithms. Technical improvements built into YOLOv4 are classified into two categories. The first category is called Bag of Freebies (BoF) and denotes improvements that can be made during training without affecting the inference speed. These include CutMix and Mosaic data augmentation techniques, DropBlock regulation, class label smoothing, Complete IoU (CIoU) loss, Cross mini-Batch Normalization (CmBN), Self Adversarial Training (SAT), multiple anchor frames for one object, cosine annealing schedule and obtained optimal hyperparameters through genetic algorithms.

The second category is called Bag of Specials (BoS) and represents improvements that have little effect on the inference time, while at the same time significantly increasing accuracy. These include a mish activation function, Cross Stage Partial Connection (CSP), Multi-input Weighted Residual Connection (MiWRC), Spatial Pyramid Pooling (SPP), Spatial Attention Module (SAM), Path Aggregation Network (PAN) and DIoU-NMS (Distance Intersection over Union Non-maximum suppression) step. YOLOv4 architecture consist of CSPDarknet53 Backbone, Neck that incorporate Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PAN) and finally YOLOv3 Head.

A month after YOLOv4, YOLOv5 was released with a very similar architecture and performance. The key difference is that YOLOv4 is written in C using the DarkNet framework while YOLOv5 is written in Python using the PyTorch framework which has a significantly larger user community. This is the reason why the YOLOv5 architecture has been chosen for the implementation of this project.

Before proceeding to the next step, it is necessary to highlight the metrics that are commonly used to evaluate the effectiveness machine learning algorithms.

Intersection over Union has been already explained. Next metrics is Precision. It is the ratio of correctly detected objects, True Positives (TP) and the total number of objects predicted by the classifier, True Positives and False Positives (FP):

$$precision = \frac{TP}{TP+FP} \tag{3}$$

True Positives object is the one that has been positively detected and has IoU greater than some predefined, threshold value.

Recall represents the ratio of correctly detected objects (TP) and the total number of objects in the data set (TP + FN) (FN-False Negative is not detected object):

$$recall = \frac{TP}{TP+FN} \tag{4}$$

This measure shows what proportion of actual positive results has been accurately identified.

From the previous definitions of precision and recall, it can be concluded that the higher the precision, the model more accurately classifies and localised positive samples. The higher the recall, the model has correctly classified and localised more positive samples.

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells how precise classifier is (how many instances it classifies and localised correctly), as well as how robust it is (it does not miss a significant number of instances).

High precision but lower recall, gives an extremely accurate model, but it then misses many instances that are difficult to classify. The greater the F1 Score, the better is the performance of the model. Mathematically, it can be expressed as:

$$F1 = 2 * \frac{precission*recall}{precission+recall} \tag{5}$$

Whether an object will be TP or FP depends on the IoU threshold. In principle, if the IoU threshold is increased number of TP will be decreased as well as the number of FP. In this case precision will be increased, while recall will be decreased. By changing the IoU threshold a dependence of the precision vs. recall can be plotted. Average Precision (AP) is the area under the Precision-Recall curve:

$$AP = \int_0^1 precision(recall) drecall \tag{6}$$

If n different classes have to be detected and k class has AP_k, then the new measure for detector, Mean Average Precision (mAP) is defined as:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{7}$$

IV. IMPLEMENTATION OF THE SYSTEM

After the system architecture has been defined, the YOLOv5 algorithm for object detection and localization selected, and the algorithm performance measures have been defined, system implementation will be described. As already mentioned, a 4-megapixel DS-2DE4425W-DE PTZ camera with a 25x optical zoom has been installed on top of the Koteks skyscraper. The Split Harbour has been monitored and recorded continuously for 30 days to cover different weather conditions as well as part of the day: sunny, rainy, windy, sunset, night. Some of the images are presented in the Figure 3.

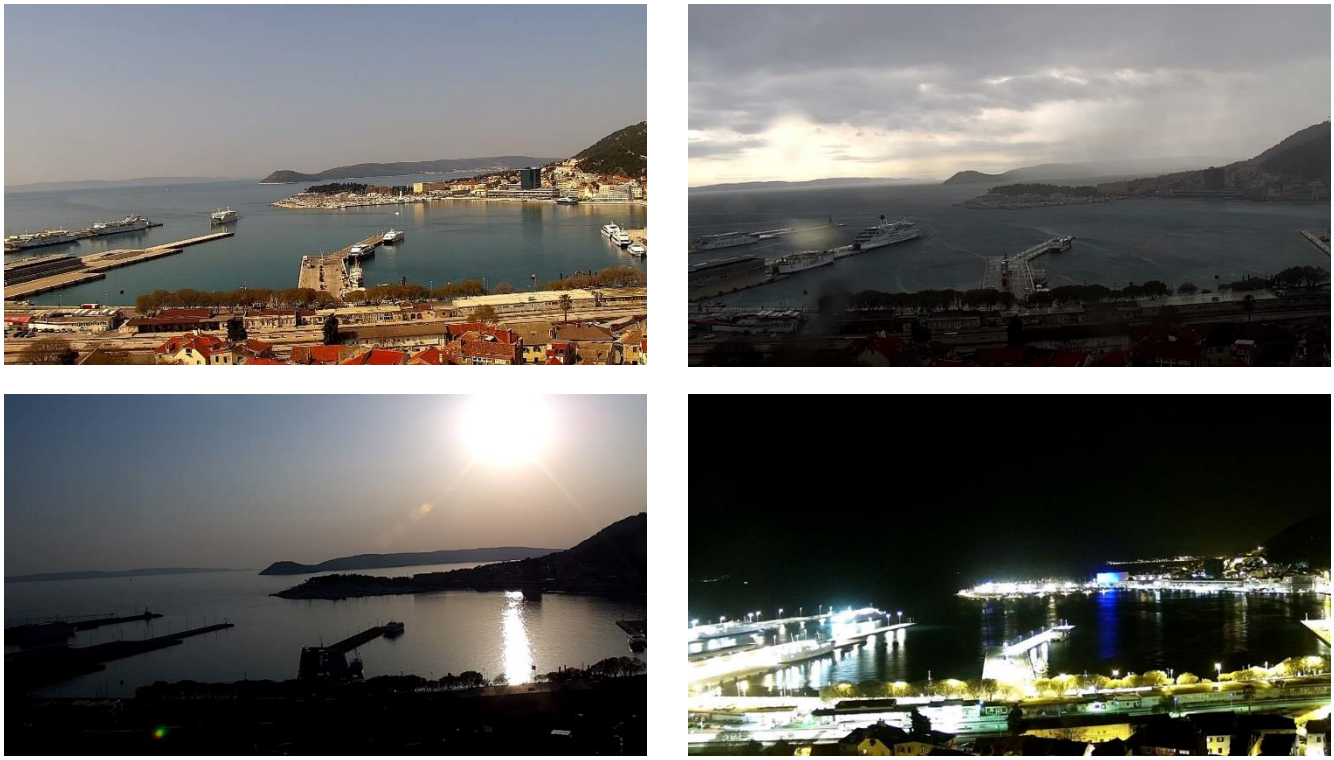


Figure 3. Images of the Split Harbour in the various part of the day and weather conditions.

The frames were taken from the videos to create an image database. It contains 324 images of which 256 are selected for training and 68 for validation, according to the common training/validation set ratio 4:1. Special care has been taken that there are no large correlation between the training and validation set. For the proof-of-concept three types of objects, ferries, catamarans and other ships, have been defined to be detected and localised.

Various tools have been considered for image labelling purposes, Computer Vision Annotation Tool (CVAT), makesense.ai, labelImg i Labelbox. Due to good and user-friendly controls, ease of use, no need for local installation, the labelling tool makesense.ai has been chosen Figure 4. shows image labelling using the makesense.ai web tool.

After the objects are manually labelled, the images are saved in the so-called YOLO format.

Training the YOLOv5 model is a very complex and demanding computing task that requires huge computer resources. Training is typically performed on processors that support parallel computing such as NVDIA Compute Unified Device Architecture CUDA. Such resources are quite expensive, and at the same time installing and setting up the necessary environment, system and application software and required library is quite demanding. In addition to the virtual environment for allocating the necessary computer resources, it is necessary to install and configure NVIDIA cuDNN accelerated library as part of NVIDIA Deep learning Software Development Kit - SDK, which optimizes the use of available hardware, what is the basis of high performance computing. Also, in order to manipulate images, it is required to install Open Source Computer Vision Library (OpenCV). In order to allow researchers to focus on the problem that solves such resources with appropriate settings are available.

So that researchers can focus on the problem they are solving and to relieve themself of the demanding and time consuming system setup the best solution is to use prebuild HPC resources. Research has been conducted on available machine learning cloud services such as Alibaba, Amazon Web Services, Google Cloud, IBM Watson Machine Learning, Microsoft Azure, Oracle, Salesforce Einstein [6].

Google Colaboratory or Google Colab has been chosen as a good solution for training our YOLOv5 network. This platform enables the writing and execution of python code via a browser, and is especially suitable for machine learning, data analysis and education. Colab is based on Jupyter notebooks, which do not require any setup, and provide free access to computer resources, including graphics processing resources. Google Colab code is executed on virtual machines, on a private profile that is pre-created. If it is not used for a long time, all data on it at that time are deleted. The service is free, and depending on the time of use, different available types of

graphics cards are available, which include Nvidia K80, T4, P4 and P100 models. It is also not possible to run at more than 12 hours continuously, and the total memory that can be used is limited to 15GB. If you want to avoid these shortcomings, it is possible to pay for Colab Pro, and use the service without restrictions.

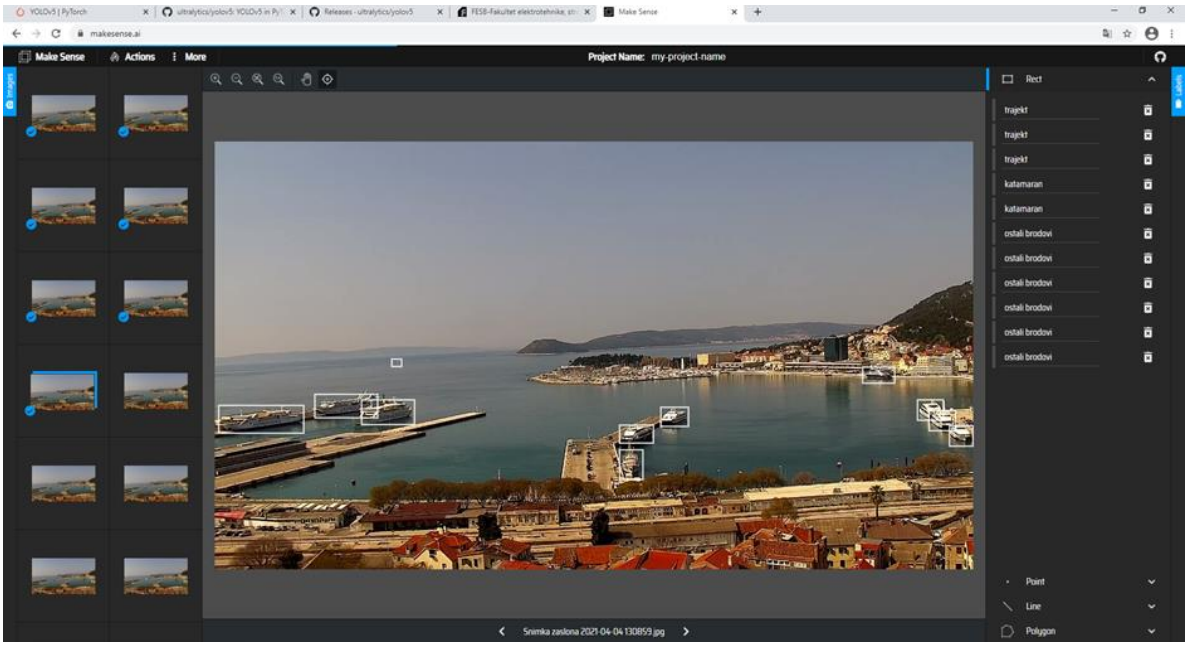


Figure 4. Image labelling using the makesense.ai web tool.

A. Model configuration and system training

As already mentioned, the Google Colab platform have been used to train and validate the model. Configuring system, running training and validation have been performed using the Jupyter Notbook platform. It is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document.

The first step is cloning the YOLOv5 to the virtual disk from the official GitHub link [10]. Afterword, the installation of the necessary Python packages is required. It is also useful to check the assigned graphics card. The next step is transfer of the labelled images, divided into train and validation set, from local disk to Google Colab. If the images are on the Google Drive, the set can be simply mounted to the Google Colab.

Before training starts, it is good to have a tool that monitors the training progress. For that purpose, Weights & Biases (WandB) python package has been installed. It allows monitoring of the training process in real-time. It can be easily integrated with popular deep learning frameworks like Pytorch, Tensorflow, or Keras.

All the information necessary for training is stored in the custom_data.yaml configuration file: paths to the labelled training and validation data set, as well as number and names of the classes.

Also, before training start, the batch size and number of epochs needs to be defined. Considering the memory size of the allocated graphics card batch size has been set to 8 images. Since training data set has 256 images, taking into account the batch size of 8 images, a total of 32 series were performed during training in each epoch. Recommended number of epochs depends on the size of the training data set and varies from 300 to 3000. For the first try a minimum number of epochs, 300, has been chosen for the training, what finally gave quite good results. For the training pretrained on the COCO image database, YOLOv5 model has been used.

V. RESULTS

After the model training, performance measures of the obtained results, precision (P), response (R) and mean average precision (mAP), are presented in Figure 5.

```
     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   297/299     9.56G   0.01979   0.02535  0.001087   0.04623        71       640
                Class    Images    Labels         P         R    mAP@.5
                  all        68       702     0.983      0.94     0.951     0.623

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   298/299     9.56G    0.0202   0.02429  0.001126   0.04562        97       640
                Class    Images    Labels         P         R    mAP@.5
                  all        68       702     0.986     0.944     0.953     0.625

     Epoch   gpu_mem       box       obj       cls     total    labels  img_size
   299/299     9.56G   0.02107   0.02539  0.001272   0.04773       118       640
                Class    Images    Labels         P         R    mAP@.5
                  all        68       702     0.986     0.942     0.951     0.619
              trajekt        68       187     0.985         1     0.996      0.75
            katamaran        68       230     0.994         1     0.996     0.642
       ostali brodovi        68       285     0.979     0.825     0.861     0.465
300 epochs completed in 2.022 hours.
```

Figure 5. Summary of the performance measure after the training.

For one epoch, the average training time with 32 batches is approximately 25 seconds on the CUDA T4 graphic card. The total training time, for 300 epochs, lasted about 2 hours. In the last epoch the mean average accuracy, for all classes together is mAP_0.5 = 95.1%, the average accuracy for ferries is 99.6%, the average accuracy for catamarans is 99.6%, and the average accuracy for other ships is 86.1%.

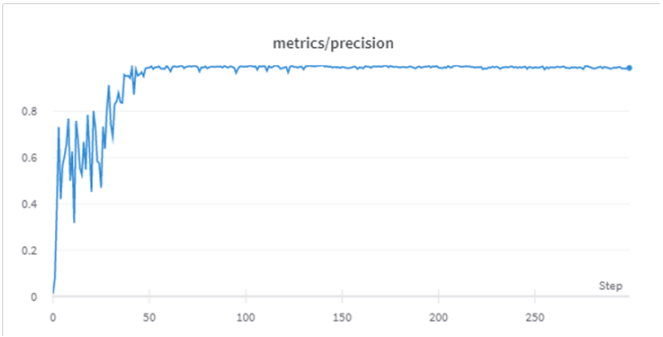
Figure 6. shows two images from validation set with labelled ground truth frames and the same images with predicted boundary frames.

The results of success measures obtained by individual epochs using the wandb tool are shown in the graphs in theFigure 7

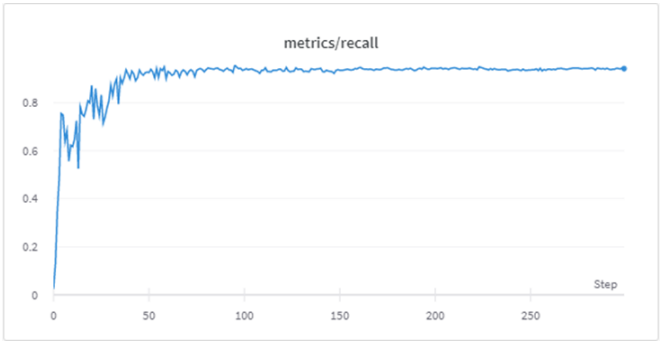
a) Two labelled images from validation set with ground truth frames

b) Predicted frames for the two images from validation set

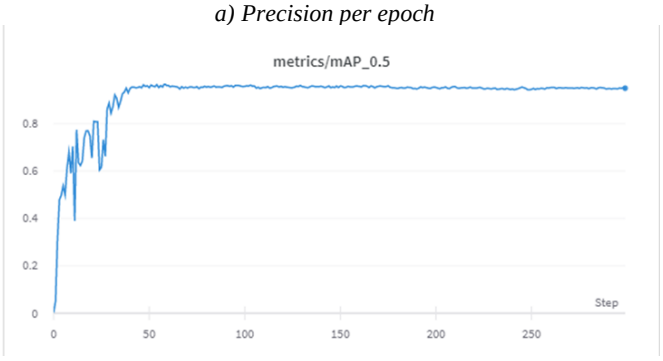
Figure 6. Detection results after model training.



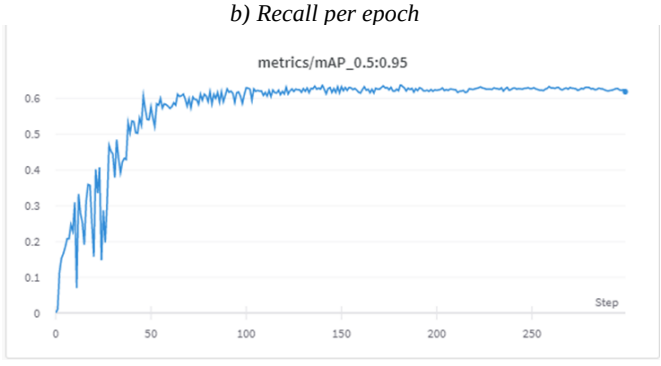
a) Precision per epoch



b) Recall per epoch



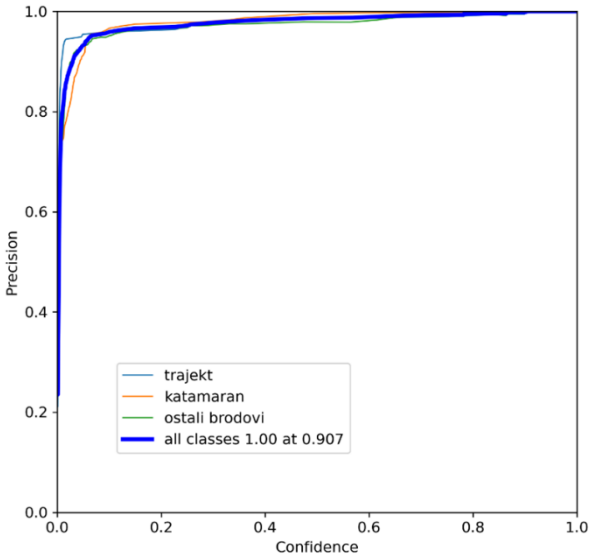
c) Mean average accuracy mAP_0.5 per epoch



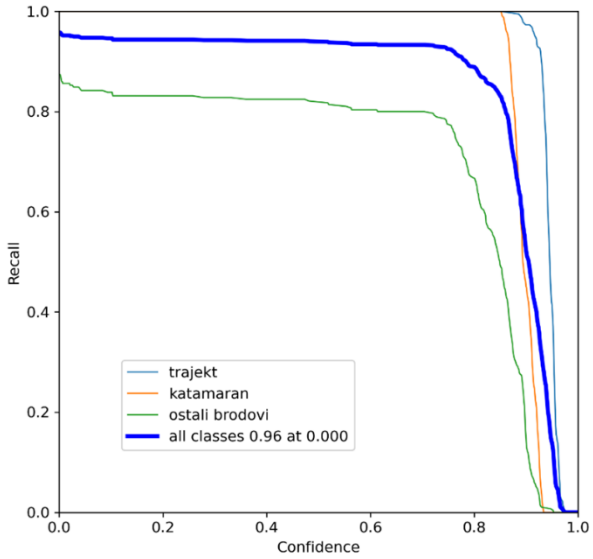
d) Mean average accuracy mAP_0.5: 0.95 per epoch

Figure 7. The results of success measures obtained by individual epochs using the wandb tool.

Graphs on the Figure 7. a) b) and c) shows that the stable, maximal precision of almost 95%, recall of 94% and mean average precision mAP_0.5 of 95% is achieved already after 50 epochs, while stabile mAP_0.5:0.95 of 62 % is achieved after 150 epochs. The conclusion is that only up to 150 to 200 epochs are enough for the model training.



a) Precision in dependence to confidence



b) Recall in dependence to confidence

Figure 8. Precision and recall in dependence to confidence.

Figure 8. shows the dependence of precision and recall to the confidence threshold. It can be seen from the graphs that as the confidence decreases, the recall increases (more ships are detected), but the precision decreases (the possibility of incorrect detection is higher). Due to the importance of both parameters for successful detection, and in order to be able to select the best confidence threshold, it is necessary to show their interdependence on a single graph.

The precision-recall graph shows different pairs of precision and recall, depending on the used confidence threshold, presented in Figure 9.

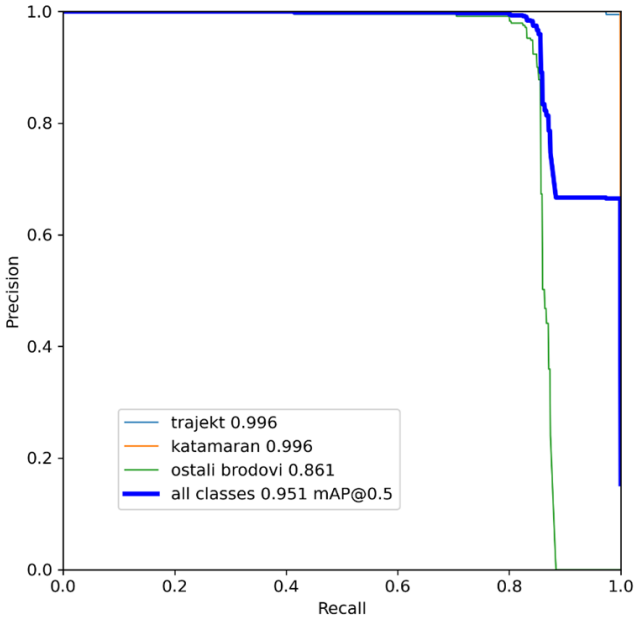


Figure 9. Precision in relation to recall.

From the graph presented in Figure 9. It could be observed that up to the recall of 80% maximal precision of 95% is obtained. Next step is to determinate optimal confidence threshold level at which the stated values of precision and recall are achieved. This value can be obtained from F1 score, presented at the Figure 10.

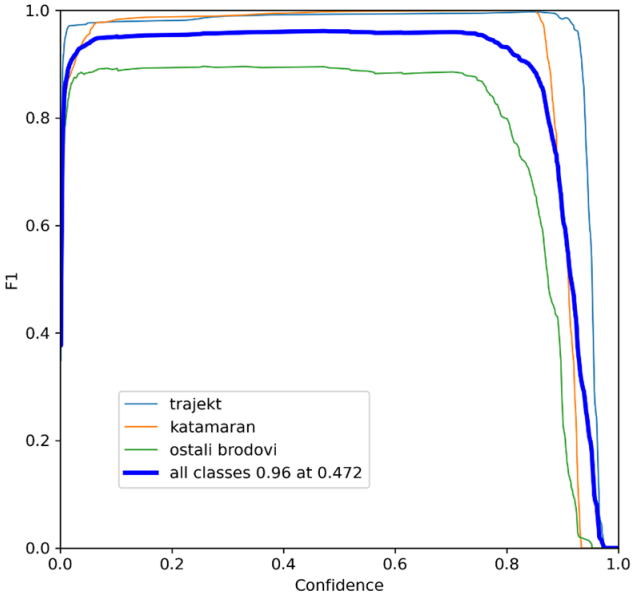


Figure 10. F1 score graph.

The graph in Figure 10. shows that the F1 score measure falls sharply for confidence values greater than 80%, while detecting the class "other ships", a slight decrease is visible already at 60%. Therefore it can be concluded that is optimal to set the confidence threshold parameter to 60% for our model.

A. Application of the Split port monitoring system

Split port monitoring system application can be started on any computer with Linux operating system and installed all the necessary packages listed in section IV. The obtained trained weights on the Colab platform has been used. Detection and localization of the objects within the source video.mp4 file is started with the command *python3 detect.py –-weights best.pt –-conf 0.6 –-source video.mp4*, where the parameter –-conf 0.6 determines the display of only those

detected objects that have a confidence level greater than 60%.

In case of detecting ships on video obtained via webcam or live broadcast, the previous command must be modified as follows:

```
$ python detect.py --source 0  # webcam
                           file.jpg  # image
                           file.mp4  # video
                           path/  # directory
                           path/*.jpg  # glob
                           rtsp://170.93.143.139/rtplive/470011e600ef003a004ee33696235daa  # rtsp stream
                           rtmp://192.168.1.105/live/test  # rtmp stream
                           http://112.50.243.8/PLTV/88888888/224/3221225900/1.m3u8  # http stream
```

With this command it is possible to start the detection of ships on video from remote surveillance cameras, with a known IP address of the camera, using RTSP (Real Time Streaming Protocol), RTMP (Real Time Messaging Protocol) or HTTP (Hypertext Transfer Protocol)) protocol.

VI. CONCLUSION

Technology, broadband and 5G network, HPC and Data Centers, Edge Computing, sensors, especially high-resolution cameras play important role in achieving Smart City vision. In this paper we have presented a solution that uses high resolution cameras, either stationary or on a drone where the video is streaming to the HPC/Data Centre via broadband or 5G network for processing. The traffic in the harbor is monitor using trained YOLOv5 deep convolution network. The process of training the YOLOv5 network is described in detail as well as the selection of all key training parameters.

The final testing of the trained model was conducted on a number of videos. The video database contains the recorded Split harbor in different weather conditions and at different times of the day. Testing has determined that the trained model detects all ships on the videos, and correctly classifies them as ferries, catamarans or other ships. Yet, occasional inaccurate classifications have been observed in some videos. Thus, at some point, a bird in flight, part of a cloud or part of an island, would be classified as "other ships."

The implemented system can be expanded and improved in various segments. Thus, it is possible to create a system that would have the ability to monitor traffic at night, which would require the use of night cameras or thermal cameras. Increasing the base of images used to train models would lead to further improvements in model accuracy, primarily eliminating, albeit very rare and sporadic, misclassifications.

ACKNOWLEDGMENT

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks ...”. Instead, try “R. B. G. thanks...”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

[1] A.N. Sarkar, Smart Cities: A Futuristic Vision, The Smart City Journal, https://www.thesmartcityjournal.com/en/articles/smart-cities-futuristic-vision, 03. July 2021.

[2] Donato Toppeta, The Smart City vision: How Innovation and ICT can build smart, “liveable”, sustainable cities, THINK! REPORT 005/2010, https://inta-aivn.org/images/cc/Urbanism/background%20documents/Toppeta_Report_005_2010.pdf, 03. July 2021.

[3] L. Jiao, F. Zhang, S. Yang, L. Li, Z. Feng i R. Qu, A Survey of Deep Learning-Based Object Detection, IEEE Access, vol. 7, str. 128837-128868, 2019.

[4] A. Bochkovskiy, C.-Y. Wang i H.-Y. M. Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, from Internet, https://arxiv.org/abs/2004.10934, 03. July 2021.

[5] J. Redmon, S. Divvala, R. Girshick i A. Farhad, You Only Look Once: Unified, Real-Time Object Detection, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, 2016.

[6] Sean Michael Kerner, Top Machine Learning Services in the Cloud, Datamation, January 29, 2020., https://www.datamation.com/artificial-intelligence/top-machine-learning-services-in-the-cloud/, 03. July 2021.