Shore Livecams: A Maritime Dataset for Deep Learning based Object Detection

Daniel Ahlers
Dept. of Signal Proc. and Comm.
Helmut Schmidt University
Hamburg, Germany
daniel.ahlers@hsu-hh.de

Purbaditya Bhattacharya
Dept. of Signal Proc. and Comm.
Helmut Schmidt University
Hamburg, Germany
bhattacp@hsu-hh.de

Patrick Nowak
Dept. of Signal Proc. and Comm.
Helmut Schmidt University
Hamburg, Germany
patrick.nowak@hsu-hh.de

Udo Zölzer
Dept. of Signal Proc. and Comm.
Helmut Schmidt University
Hamburg, Germany
zoelzer@hsu-hh.de

***Abstract*—This paper introduces a maritime dataset for object detection named as the shore livecam dataset. The dataset is a collection of high definition (HD), full high deifinition (FHD), ultra high definition (UHD) images captured from live video feeds recorded accross various port based areas of Germany. These images contain multiple instances of objects which are primarily classified into three different classes and annotated accordingly. The widely varying object sizes contribute to the uniqueness of this dataset whose content is thoroughly analysed and described. Finally, a selection of deep learning models is used on this dataset for evaluation.**

***Index Terms*—Maritime dataset, object detection, deep learning, convolutional neural network, image processing**

I. Introduction

The development of surveillance systems with cameras towards autonomous vision systems is going on rapidly. With deep learning and convolutional neural networks (CNN) a breakthrough has been achieved in recent years. This has led to an explosion in the field of object detection and machine learning. But at the core of these technologies an incredible amount of data is needed. In the domain of object detection, among others the datasets of PASCAL VOC [?], ImageNet [?], COCO [?], Open Images [?] and VisDrone [?] are jointly responsible for the success and still set the benchmarks for new object detectors today to compare against. However, to achieve satisfying results on more specific tasks a well picked and prepared dataset is needed. Our task is to detect all kind of moving vessels in the maritime environment as well as humans to automate surveillance tasks by creating notifications. It could also be used for collision avoidance for water transport and identification for vessels which are not equipped with Automatic Identification System (AIS) transceivers. Datasets that are closes to our task and created in a maritime environment are the Singapore Maritime Dataset [?], that solely contains boats, and the dataset SeaDroneSee [?] that is focusing more on search and rescue with drone videos. Therefore such datasets do not completely represent the object classes required for our task. This is why an own dataset is created, named ”Shore livecams”, which can be used as a benchmark on real world images in the maritime environment.

Special characteristics of this dataset are:

- Instead of capturing the images or using images from image libraries, public livestreaming Internet Protocol (IP) cameras are used.
- The number of annotated objects per image is very high.
- The range in size of the annotated objects is very large and there is a high number of very small objects.

This paper first explains the method used for data collection and the annotation process by addressing the challenges that came with it. In the second section, the dataset will be analysed by object count, size and distribution, followed by a section where the dataset is split and trained with different object detectors. There, the results are discussed based on the mean average precision (MAP) metric. Finally, the findings are concluded and possibilities for improving the results in the future are discussed.

II. Dataset collection and labelling

To collect real world images quickly, on a variety of locations at different daytimes and weather conditions, public IP camera streams are chosen, that are often used by municipality, cities, or companies as part of their external presentation. The cameras are in operation 24/7, close to the environment we are looking for (harbour and sea), and most are pan-tilt-zoom (PTZ) cameras so they also move around to different areas. The cameras used are positioned at the North Sea, Baltic Sea and Lake Constance. To record the camera streams a recording software is developed with python that is based on FFmpeg [?]. Ten IP cameras streams are recorded from 16.07.2022 till 11.08.2022. The streams are provided as H.264 HTTP Live Streaming (HLS) video streams with bit rates between 750 and 12000 kbit/s and resolutions of 720p, 1080p and 2160p at mostly 25 fps. So they are very heavily compressed to get transmitted over the internet. One stream was offline and only showed a static image, but in total 7.52 TB of video are recorded during this period. The videos are stored in half hour video snippets, and for the dataset an arbitrary single image is extracted per video clip. After removing the images during the night time, around 30 images per day per video stream are taken.

Because the labelling is the most time consuming task and by reviewing the recordings, only three days are labelled in order to reduce the redundancy of the same scenery. The labelling is done by hand using the software labelImg [1]. The large difference in the scenery and therefore in the number of objects per image made it impossible to estimate the time to label an image in advance.

After training an object detector and testing it on the dataset, several objects are found by the object detector that weren't labelled by hand, resulting in worse results. Therefore, the predictions of the detector are exported and combined with the ground truth labels in order to generate updated annotations. These annotations are checked by hand and wrong detections are removed. Additionally, labels of objects that are still missing are appended. In this way, the number of labels is increased from 6821 to 9243 labels. (see Table I).

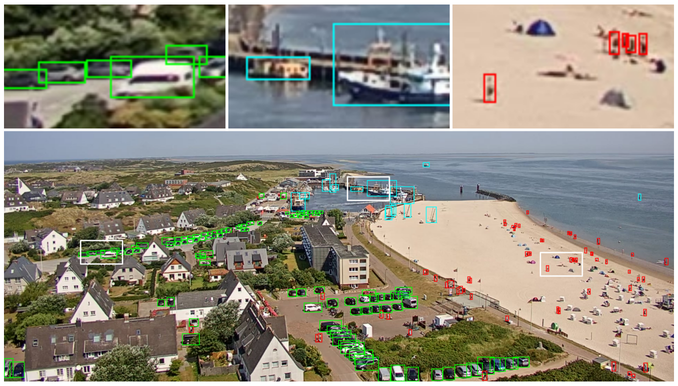
In Fig. 1, three examples are displayed to show the diversity of the dataset. In Fig. 1(a), parts of the image have been enlarged to make the very small objects visible. In this image, the harbour is quite far away, which is why all three object classes have similarly sized bounding boxes. Contrarily, Fig. 1(b) shows a very large ship object. The human object can barely be seen in the centre of the image. Another example of small objects in a far distance at the horizon level is represented in Fig. 1(c).

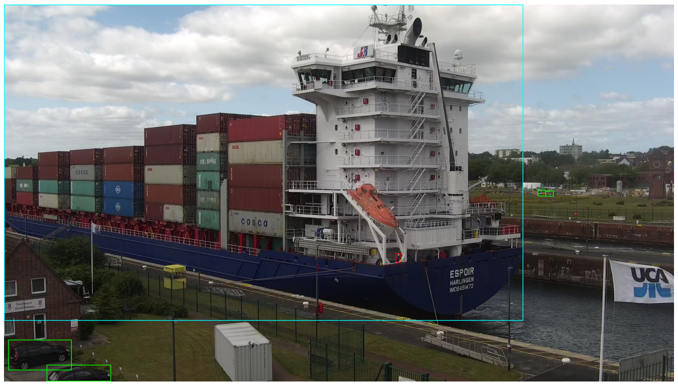
III. Dataset analysis

To get a better understanding of the dataset different python scripts are used, the software COCO-Dataset-Explorer [2] is extended to our needs and Matlab scripts are created for a better visualisation.

As can be seen in Table I the dataset consists of 525 images, which are separated in 392 at 1080p, 116 at 720p, and 17 at 3840p. All images have an aspect ratio of 16:9. In total there are 9243 objects, which results in 17.6 objects per image on average. The object classes are relatively evenly distributed showing that the set of IP cameras was well picked. In detail, 32.9 % of the objects are humans, 36.6 % are sea vessels, like cargo ships, cruise ships, small boats and sailboats. The rest of the objects (30.5 %) are land vehicles, mostly cars but also some trucks, motorcycle and exotic vehicles like excavators.

In Fig. 2, the number of objects in every image is plotted, subdivided by the object classes. The images are sorted by the nine camera streams and by time and day. The figure shows that in some camera streams there are a lot more objects than in other streams due to the different scenery. In the first camera stream (image id: 1-68), also shown in Fig. 1(a), the coast is dominant in the image with a parking space, the beach and a harbour. This results in a high number of objects per image especially of human and land objects. The second peak of objects in image id 427 to 443 can also be traced back to a


(a) Crowded scene with a lot of small objects (183)


(b) Scene combining large and small objects

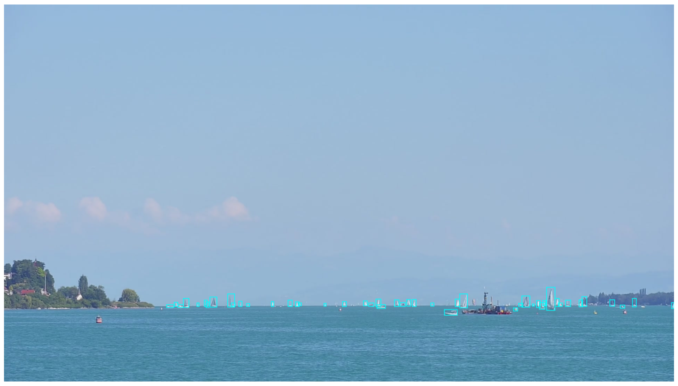

(c) Scene with small ships at the horizon

Fig. 1: Exemplary images of the dataset with bounding boxes for human (red), land (green), and sea (blue).

TABLE I: Number of images, objects and objects per class before and after verification and analysed for small ($\leq 0.33\,\%$) and medium + large ($> 0.33\,\%$) objects

	images	objects	human	land	sea
Manually labelled	511	6821	2099	2417	2305
CNN + Manually corrected	525	9243	2821	3378	3044
Small	503	8726	2818	3287	2621
Medium + large	209	517	3	91	423

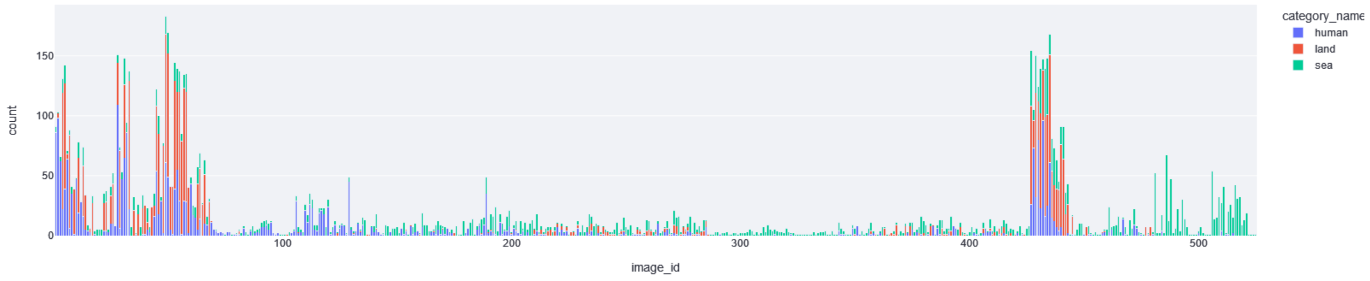


Fig. 2: Annotations per images subdivided by the object classes.

harbour and a parking space in the scenery of this camera. The different amount of objects within camera streams results of the different times of day, with less activity in the morning and evening hour, but also because the cameras are not static and the image section differs between images. The camera streams with image id 287 to 348 and 481 to 525 contain almost only sea objects. This also results in the scenery. An example of the last stream can also be seen in Fig. 1(c).

To get a better understanding of the challenges for an object detector it is crucial to analyse the size of the objects. To make it comparable across the different image resolutions we choose to use the area of an object relative to the size of the image. This is also beneficial for the quality analysis of an object detector because the images are normally resized to the input size of the convolutional neuronal network. Our splitting between small and large objects is based on the COCO [3] metric, which uses absolute pixel values. The median image size in COCO is 640×480. Converting the absolute pixel sizes to relative ones results in less than 0.33 % for small objects, greater than 3 % for large objects, in-between objects are of medium size. For comparison, the white boxes in Fig. 1(a) showing the enlarged areas have a size of 0.42 %. On average, the objects in the dataset have a size of 0.3 %. The average size per class are 0.84 % for sea objects, 0.05 % for land objects and 0.01 % for human objects. Table I also shows the distribution for small objects compared to medium + large ones. Because less than 6 % of objects where of size medium and large, both are combined. After splitting the objects in these two groups, it can be concluded, as expected, human objects are small, land objects are larger, and sea objects are huge. In total, there are a lot of very small objects in this dataset.

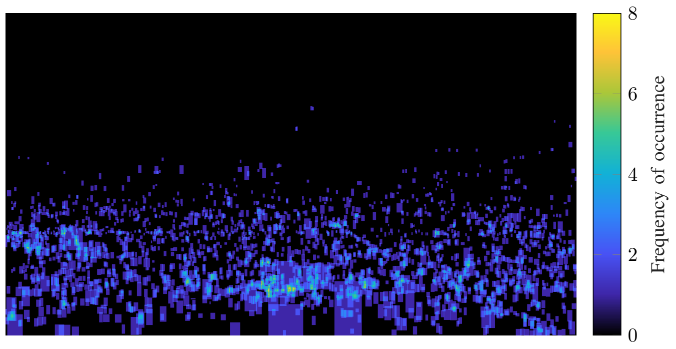
After analysing the amount and size of the objects, a closer look at the distribution of the objects in the images is taken. To visualize this, heatmaps are created showing all object annotations in one singe image depicted in Fig. 3 subdivided by classes. In Fig. 3(a), it can be seen that human objects are very small and therefore their bounding boxes do not overlap much. They accrue mostly in the lower half of the images, this can be explained by the fact that people only walk below the horizon level. This also applies to the land objects in Fig. 3(b), with the difference that the objects are larger and a few dominant objects are in the centre in some images. The hotspot at the lower left is a parking space in the foreground on one camera stream. In contrast, the sea objects are located everywhere on the image shown in Fig. 3(c), with a dominance at the centre line which coincides with the horizon in a lot of images. An example is shown in Fig. 1(c).

IV. Deep Learning based Object Detection

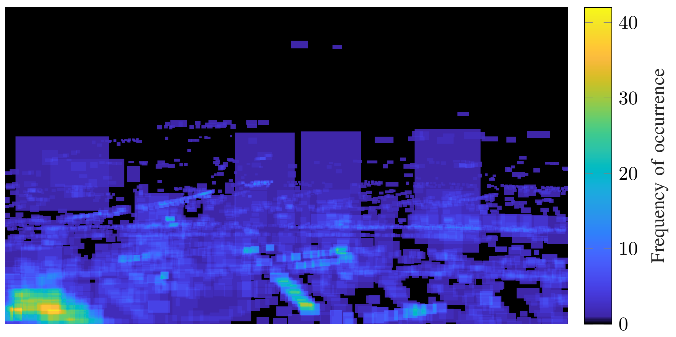
In this section, the dataset is prepared in order to train with different deep learning models. To train the models, two different dataset types are prepared. The first approach is to split the dataset into training and test images based on the camera streams. Images captured by six cameras are selected for training while the remaining cameras are selected for testing. The split is done in such a way that the distribution of objects from different classes does not differ massively in the training or test sets. However, it is possible that objects of certain dimensions are less frequent in either of the sets. Table II shows the corresponding split across all classes (All^{C}), where almost 75 % of the images are selected for training while the remaining images are used for testing.

The second approach to dataset preparation is to split the dataset based on the date of video recording. Images captured on two days are used for training while the images captured on the remaining day are used for testing. Such a split ensures that images from all cameras are included in both of training and test set. Table II shows the corresponding distribution of data (All^{D}), where nearly 75 % of the images are used for training. This approach was also used to analyze the adaption of an object detector solely trained on small objects (Small^{D}) and another object detector solely trained on large objects (Large^{D}), to observe the difference in performance. While the number of training and test images corresponding to small objects is quite similar to that of all objects, the corresponding numbers are reduced in case of large objects.

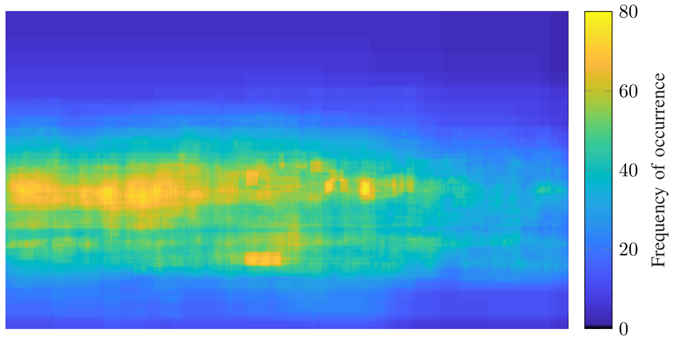
Three deep learning models from the YOLO family - YOLOv5 [4], YOLOv6 [5], and YOLOv7 [6] are used in the evaluation. The models are very similar in architecture containing a backbone, neck, and a head structure. The backbone is a pyramidal structure which produces feature maps of multiple dimensions to generate multiresolution contexts. The neck of such architectures contains bottom up and top down structures. It is responsible for the combination of low-level structural features and high level semantic features generated in the backbone. The features from the branches of the neck are processed further in the head and are split into a classification and a regression branch. At the end of these branches the



(a) Human objects



(b) Land objects



(c) Sea objects

Fig. 3: Heatmap of the distribution of the objects in the image. Black denotes areas with no object.

TABLE II: Number of images, objects and objects per class for different train test splitting scenarios. ^{C} denotes train and test split by camera and ^{D} denotes train and test split by day.

		images	objects	human	land	sea
All^C	train	392	6495	2004	2418	2073
	test	133	2748	817	960	971
All^D	train	394	5643	1624	1976	2043
	test	131	3600	1197	1402	1001
Small^D	train	376	5252	1621	1903	1728
	test	127	3474	1197	1384	893
Large^D	train	158	391	3	73	315
	test	51	126	0	18	108

final features are sent to the corresponding loss layers for error minimization during training. However, certain individual modules are different between the three models which result in a difference in their performances.

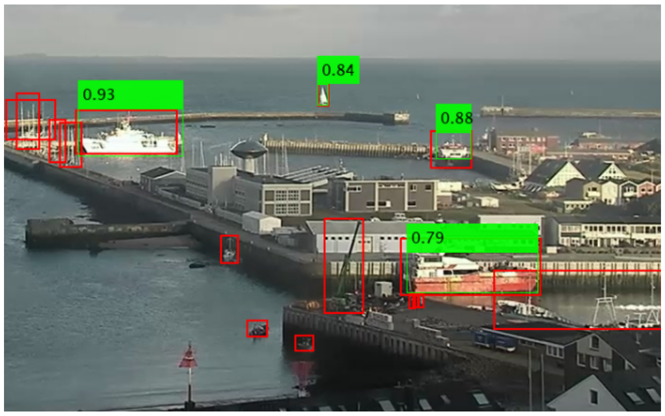
YOLOv5 uses the CSPDarknet53 [?] as its backbone - a variation of the Darknet structure introduced in YOLO. On the other hand, YOLOv6 uses an EfficientRep backbone [?], which contains re-parametrizable modules for an efficient inference. Such modules are also present throughout the neck of YOLOv6. YOLOv7 also uses efficient re-parametrizable modules in multiple forms. Additionally, it employs efficient training methods where model weights are averaged across multiple training runs or across multiple epochs to improve its generalization capability.

The pre-trained models are initially retrained on the dataset denoted by All^C for the input resolutions of 1280×1280 and 1920×1920, separately. The corresponding results on the test dataset are provided in Table III where the mean average precision (MAP$^{IOU=0.5}$, MAP$^{IOU=0.5:0.95}$) values are used as performance metrics. The results indicate that a higher input resolution produces better scores likely due to increment in dimensions of the small objects. It can also be seen that YOLOv6 outperforms the other networks in terms of MAP scores but is slower than YOLOv7. Generally, the MAP scores are quite low which can be attributed to the large difference in scenery between cameras, where certain objects and backgrounds in the test dataset are not present in the training set. The models are then trained on the dataset All^D where images from all cameras are present in both of training and test sets. From the results in Table III, an improvement in MAP scores can be seen with respect to the previous results, across all models. An example of detection by YOLOv6 on a test image from All^C is shown in Fig. 4a and the example on the same image from All^D is shown in Fig. 4b, where the detection performance is relatively better.

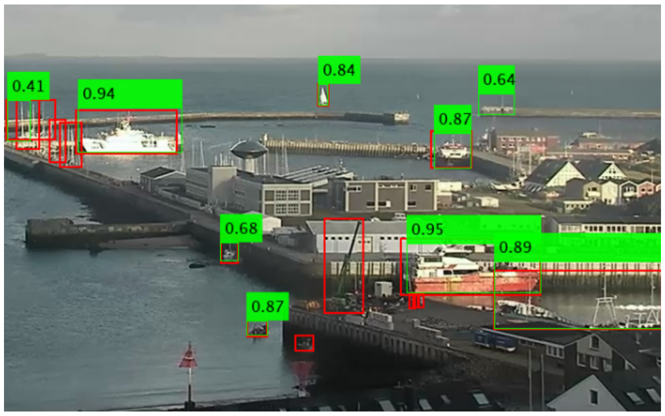
In spite of the homogeneity in dataset distribution across training and test datasets the scores have not improved massively. Hence, the models are trained separately on images containing only the small objects and images containing only large objects as defined previously. The YOLO models use an algorithm to find the initial anchor boxes based on the distribution of object sizes in the datasets. It is usually difficult to define a finite number of initial anchor boxes that range from a very small value to a very large value and make them converge to all kinds of solutions. A separation of objects based on size makes the individual distributions narrower. It was however observed that the algorithm still struggled to fit to the dataset containing only small objects. The results are shown in Table III as well. The performance of the models on the test dataset of Large^D is much improved compared to the test set of All^D, while the performance on the test set of Small^D is comparatively worse. This is primarily attributed to the resolution of small objects which result in low detection confidence scores. Lower values of confidence threshold result in many false positives in the case of small objects. As for the models, YOLOv6 and YOLOv7 perform better than YOLOv5,

TABLE III: MAP results of different object detection networks. $*^C$ denotes train and test split by camera and $*^D$ denotes train and test split by day

Model	Input resolution	$\text{MAP}^{\text{IoU}=0.5:0.05:0.95}$				$\text{MAP}^{0.5}$				Average Duration (ms)		
		All^C	All^D	Small^D	Large^D	All^C	All^D	Small^D	Large^D	Proc.	NMS	Total
YOLO-v5m6 (35.30M)	1280 × 1280	0.065	0.131	0.096	0.451	0.152	0.303	0.242	0.655	27	13	40
	1920 × 1920	0.084	0.169	0.114	0.455	0.202	0.381	0.311	0.746	55	80	135
YOLO-v6m6 (79.53M)	1280 × 1280	0.094	0.178	0.154	**0.567**	0.202	0.381	0.339	0.765	34	115	149
	1920 × 1920	**0.157**	**0.221**	**0.211**	0.511	**0.308**	0.441	**0.442**	0.718	63	147	210
YOLO-v7W6 (80.94M)	1280 × 1280	0.078	0.154	0.103	0.533	0.177	0.369	0.279	0.763	26	2	28
	1920 × 1920	0.109	0.210	0.152	0.538	0.259	**0.461**	0.380	**0.785**	33	3	36



(a) Detection by YOLOv6 trained on All^C



(b) Detection by YOLOv6 trained on All^D

Fig. 4: Exemplary image from the All^C and All^D test dataset with detection by YOLOv6 denoted in green and ground truth denoted by the red boxes.

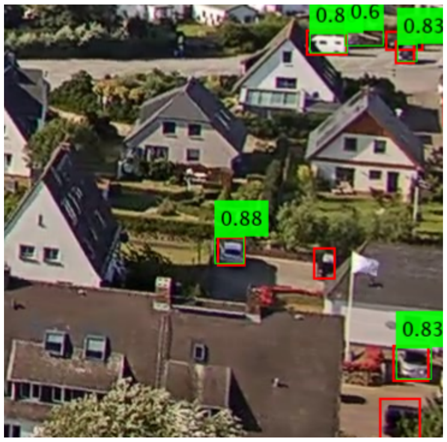
while YOLOv7 results in faster inference.

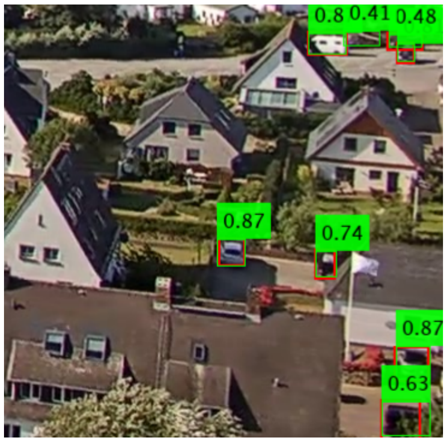
Some examples of object detection by the models on a set of image snippets from the test dataset of All^D are shown in Fig. 5. The image snippets shown in the figure are cropped from 1920 × 1920 images, where a red box denotes a ground truth object and a green box with confidence score denotes a detection. The first row of the image shows detection examples of land objects. It can be seen from the example that YOLOv5 is unable to detect two objects which are indeed relatively difficult. YOLOv6 and YOLOv7 are able to detect all objects in the particular section of the image. Additionally, YOLOv5 and YOLOv6 also detect an object which is not labelled as a land object. The second row of Fig. 5 shows a cropped section of a test image of small sea objects. In this example, the performance of the models is very similar. All of the models are unable to detect a couple of instances of a boat, while YOLOv5 and YOLOv6 make an additional detection which is not labelled. The third row of the figure shows an image containing mostly humans as the object of interest. Similar to the above examples, the models are able to detect most examples with YOLOv6 and YOLOv7 performing relatively better than YOLOv5. The final row of Fig. 5 shows an example of detection of large objects. In this example, YOLOv7 is able to detect all the large objects although one of the estimated bounding boxes around a ship is much larger than the ground truth bounding box. YOLOv5 and YOLOv7 are unable to detect the larger ship but YOLOv5 has generated a better bounding box to detect the smaller boat. It is noteworthy that a very small object in the background is also detected by YOLOv5 and YOLOv7

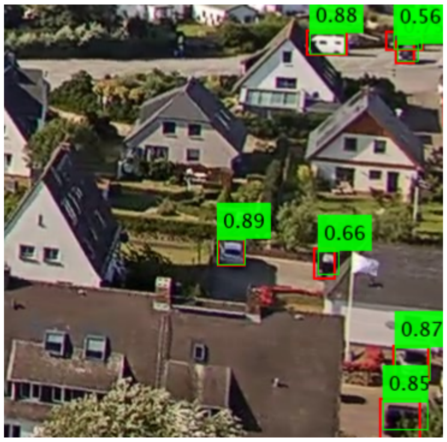
In general, the models perform relatively well on detecting medium to large objects but perform poorly in the detection of small objects, particularly in difficult backgrounds. Hence, simultaneous detection of small and large objects on images remains a challenge even when using deep learning models that are highly regarded in terms of object detection.

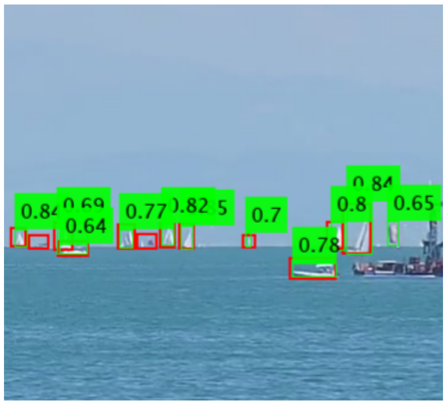
V. Conclusion

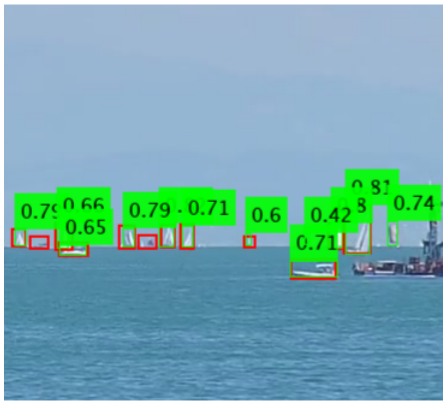
This paper introduces a maritime dataset for object detection. The dataset contains images of multiple resolutions captured from live harbour recordings in many places of north Germany. The objects in the extracted images are divided into three classes and annotated to create a dataset capable of training by deep learning models. Following the analysis of the dataset a selection of deep learning models are trained on it and evaluated. As part of the future work, the dataset can be expanded with images from more camera feeds to improve its diversity. Increment of the dataset should help improve the robustness of the detection systems to new scenarios. The evaluation of the CNN models on the dataset indicates

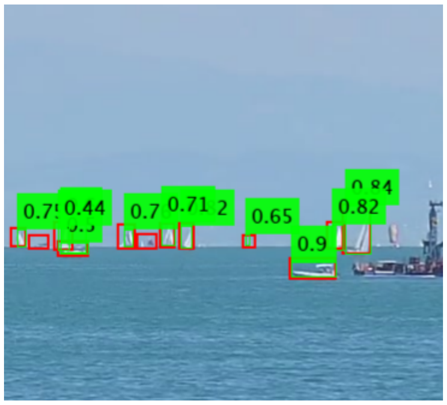

(a) YOLOv5

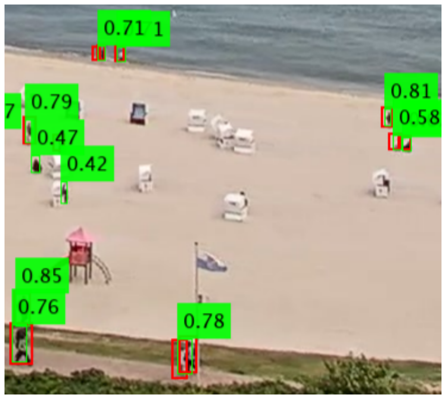

(b) YOLOv6

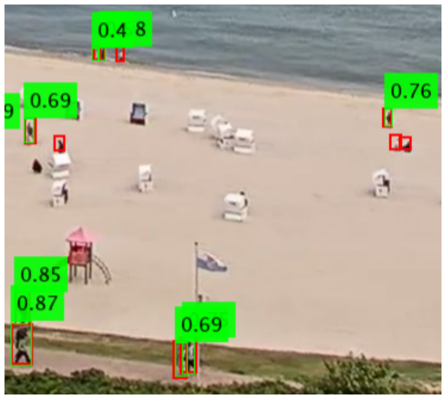

(c) YOLOv7

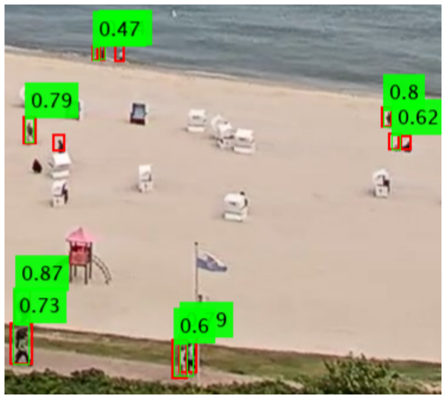

(d) YOLOv5

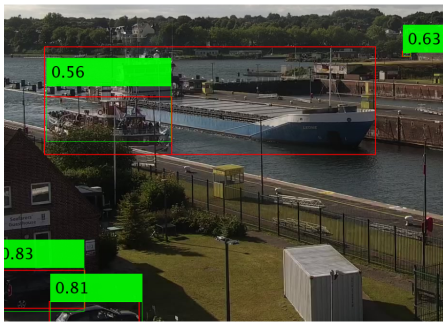

(e) YOLOv6

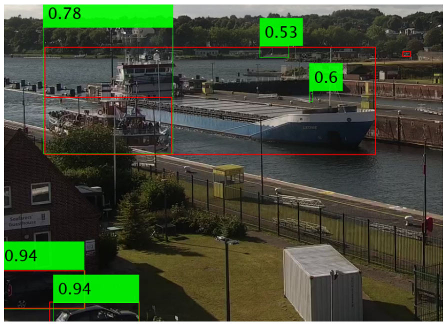

(f) YOLOv7


(g) YOLOv5


(h) YOLOv6


(i) YOLOv7


(j) YOLOv5


(k) YOLOv6

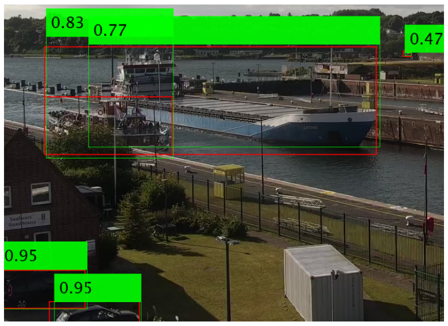

(l) YOLOv7

Fig. 5: Example of object detection on selected images from the test dataset by the YOLO models. Cropped sections of the entire image are shown for better visibility.

a poor performance in small object detection which should be addressed. One of the solutions might be an increment in the number, scale, and density of anchor boxes and initialize them properly. On the other hand, a network can be built which solely addresses the small objects and enhances the features corresponding to small objects, in terms of resolution. Additional baseline models should be experimented with and additional modules like attention, vision transformers, and its variants should be explored in order to improve detection performance.

REFERENCES

[1] T. Lin, "LabelImg," Free Software: MIT License, 2015, [Online], Available: https://github.com/heartexlabs/labelImg. [Visited on 26. August 2023].

[2] J. Cieslik, "COCO-dataset-explorer," 2022, [Online], Available: https://github.com/i008/COCO-dataset-explorer. [Visited on 26. August 2023].

[3] T. Y. Lin et al., "Microsoft COCO: common objects in context," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 740–755.

[4] G. Jocker et al., "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Zenodo, October 2020, https://doi.org/10.5281/zenodo.4154370.

[5] C. Li et al., "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," in *ArXiv*, 2209.02976, 2022.

[6] C.-Y. Wang, A. Bochkovskiy, H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464-7475