Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-21 (2019) 64–71

Assessing Deep-learning Methods for Object Detection at Sea from LWIR Images

Frederik E. T. Schöller * **Martin K. Plenge-Feidenhans'l** *
Jonathan D. Stets ** **Mogens Blanke** *

* *Technical University of Denmark, Department of Electrical Engineering, Automation and Control Group, DK 2800 Lyngby, Denmark (e-mail: {fets,mkpl,mb}@elektro.dtu.dk)*
** *Technical University of Denmark, DTU Compute, Lyngby, Denmark (e-mail: stet@dtu.dk)*

Abstract: This paper assesses the performance of three convolutional neural networks for object detection at sea using Long Wavelength Infrared (LWIR) images in the $8-14\mu m$ range. Capturing images from ferries and annotating 20k images, fine-tuning is done of three state of art deep neural networks: RetinaNet, YOLO and Faster R-CNN. Targeting on vessels and buoys as two main classes of interest for navigation, performance is quantified by the cardinality of true and false positives and negatives in a random validation set. Calculating precision and recall as functions of tuning parameters for the three classifiers, noticeable differences are found between the three networks when used for LWIR image object classification at sea. The results lead to conclusions on imaging requirements when classification is used to support navigation.

Keywords: Object Detection, Autonomous marine crafts, Navigation, Long-wave Infra-red, Detection at sea, Autonomous Ship.

1. INTRODUCTION

Accurate detection of objects at sea is an important step towards safe autonomous navigation and collision avoidance for marine vessels. Relevant objects span from large ships and ferries to small sailing boats, kayaks and other objects without a clear radar signature. It also includes stationary objects such as buoys, land, bridge pillars and similar. The location of these objects contains essential information for safe navigation.

It is necessary to identify objects within a certain distance of own vessel when aiming at assisted or unattended navigation. Radar, ECDIS and GNNS can provide such an overview, but some leisure boats and smaller objects on the water are difficult to identify by radar. Object classification using electro-optical sensors can address this problem and be a reasonable alternative to a human lookout (Blanke et al., 2019).

Methods reported in use utilize images in the visible and infrared parts of the spectrum (Moreira et al., 2014) and (Prasad et al., 2017). The latter applied traditional computer vision methods, including different edge detection techniques and concluded that thermal imaging has distinct features but has drawbacks including sensitivity to weather change and saturation at day time. Other studies aimed at detection from the air, see (Leira et al., 2015) and (Rodin and Johansen, 2018), who found the combination of visible light and thermal range images to be very sensitive.

Both detection and classification of objects are needed for safe navigating at sea. Deep learning methods have proven to work well in a large variety of image-based object classification tasks and are popular e.g. within autonomous driving. However, for object detection at sea, comparative studies and guidance to the choice of architectures are sparse, so this paper focuses on applying these models and investigate how well they perform. We employ three state of the art deep neural networks, specifically RetinaNet (Lin et al., 2018), YOLOv3 (Redmon and Farhadi, 2018) and Faster R-CNN (Girshick, 2015).

The focus of this study is on Long Wavelength Infrared (LWIR) imaging. A dataset of LWIR images was acquired from ferries in near coastal service and objects at a large range of distances are annotated in two main classes: boats and buoys. The inputs to each of the networks are single images and the annotated data is used for training and testing. Performance indicators are calculated on the outputs of the three networks and show that they perform well on LWIR images. The study also investigates the networks' ability to detect and classify boats and buoys at far distance and provide statistics to evaluate the three networks for the purpose of object detection for navigation.

The paper is organized as follows. Section 2 provides a brief overview of object classification using deep learning and Section 3 refers to results of others. Section 4 introduces training and validation data and details on the three network architectures compared in this study. Performance is presented in Section 5 and discussion of results and

10.1016/j.ifacol.2019.12.284

of navigation perspectives in Sections 6 and 7. Section 8 presents the conclusions.

Table 1. Acronyms and Abbreviations

abbreviation	meaning
CNN	Convolutional Neural Net
COCO	Microsoft image dataset for deep learning
CPA	Closest point of approach
DTU	Technical University of Denmark
FP, FN	False Positives / False Negatives
FPN	Feature Pyramid Network
GNSS	global navigation satellite system
IoU	Intersection over Union
LWIR	Long Wave InfraRed $8-14\mu m$
R-CNN	Region-based CNN
ROI	Region of Interest
RPN	Region Proposal Network
small object	occupies an area less than 81 pixels
minibatch	number of images inferred before changing weights
TP, TN	True Positives / True Negatives
YOLO	You Only Look Once CNN
mAP	Mean Average Precision

2. OBJECT DETECTION AND CLASSIFICATION

This study uses image-based object detection and classification to determine which objects are present in the relevant surroundings of own vessel and their approximate positions. Data-driven solutions to object classification, such as deep neural networks, have proved to give robust and accurate results but these require large sets of annotated training data. Techniques that require less or no prior data also exist but tend to be less generalisable than a learning-based approach.

The task for navigation includes not only to detect the presence of a particular object, say a sailing boat, in an image but also to provide a bounding box around the object and thereby make a localisation in the image. The localisation is needed for two reasons. One is to track an object between consecutive images frames, another is to estimate approximate bearing and distance using the geometry properties of the imaging (pinhole model) and knowing, by calibration or estimation, the optical axis of the camera. The relative position of other vessels is also useful for subsequent sensor fusion with data from Radar and AIS when these are available.

The tracking and sensor fusion parts of the data processing needs accurate bounding box location information in the image plane, and needs to know the class to which a detected object belongs. The focus of this paper is on obtainable quality of this object detection and classification. We use Convolutional Neural Networks (CNN) to perform this task.

A CNN consists of several layers of convolutions, and multiple filters are trained for each layer. After training the network, the filter output in each layer of the CNN are features in the input images. These features are very general in the top most layers of the network, e.g. lines, corners, but become more specific in the deeper layers of a CNN.

A fully connected neural network is used to classify the features generated by the CNN, mapping the features to a vector that contains the predictions for each class in the dataset. Fully connected means that the mapping from input to output of each layer is a linear combination of all the inputs instead of a convolution.

3. RELATED RESEARCH

Several previous works address object detection, classification and tracking in a maritime environment. Surveys including (Moreira et al., 2014) list the generic approaches and in particular show results with infrared and visible light images. Challenges include waves that can cause a rapid change in the frame of reference (Fefilatyev et al., 2012), sudden change of illumination and unwanted reflections from the water (Bloisi et al., 2014), and the possibility of poor weather conditions that reduce the range of sight. The thermal range $8-14\mu m$, referred to as the LWIR range, has salient features in avoiding these artifacts and has the ability to provide clear images without daylight. LWIR imaging has, however, been quoted to be difficult to analyse (Prasad et al., 2017).

Other studies have covered images of the marine environments, seen from a ship or from an aircraft and deep learning methods were found effective by (Leclerc et al., 2018) and (Bousetouane and Morris, 2016). Other methods, in particular edge detection, was emphasized for airborne use in (Rodin et al., 2018) and (Rodin and Johansen, 2018). While deep learning methods have been applied, a comparison of performance of CNN networks' ability to detect and classify objects at sea has not been reported in the open literature for LWIR imaging. This has been the motivation for this research.

4. METHOD

This section describes the dataset used for training and evaluation, and introduces the three object detection networks selected for this study.

4.1 Dataset

A dataset of 21322 LWIR images is used to assess the neural networks. Images in the dataset are from different ferries sailing in the southern Funen archipelago with a camera facing in the direction of travel. The sensor used for obtaining the images is a Dalsa Calibir 640 GigE, equipped with a 90° lens, and has a resolution of 640 × 480 pixels.

The images were annotated with bounding boxes using custom software. The annotations were split in to two main classes, *boat* and *buoy*, but the *boat* class was split into the sub-classes *ferry*, *merchant vessel*, *vessel under oars*, *sailboat* and *motorboat*. Attributes to *sailboat* are *sails up* and *sails down*, and general attributes are *fishing*, *not manoeuverable*, and others for which particular traffic rules at sea are described in COLREGs. This paper addresses object classification in the two main classes, as our primary goal is the assessment of the performance of different neural networks in a general maritime setting. Table 2 shows the cardinality of objects in the dataset.

There is approximately one annotation per image, however, it should be taken in to account that the annotations

Table 2. Occurrences of classes in the dataset

Total images	Boats	Buoys
21322	26343	5644

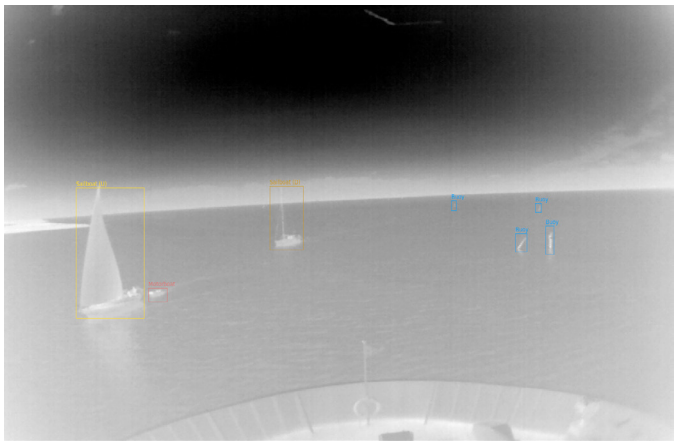

Fig. 1. Annotation Example: Images with annotations of several types. The boat classes shown here are consolidated before training of the network.

are not evenly distributed over the dataset. That is, there are many images containing no annotation, along with images containing several annotations. An example of an image with many annotations is shown in Figure 1.

4.2 Network Architectures

We assess three state of the art object detection networks; RetinaNet, YOLOv3 (You Only Look Once, Version 3) and Faster R-CNN. In the following we briefly introduce the architecture of each of the networks and why they are interesting to investigate for our specific application.

The three networks have been selected, partly because they show good results on the MS COCO and PASCAL VOC datasets, see Zhang et al. (2018), but also because they cover a range of different network types. Faster R-CNN is a classical two-stage detector which generally yields the best results, but is quite slow. RetinaNet is a one-stage detector that uses a novel loss function, that has shown to increase performance to levels comparable to the two-stage detectors. YOLOv3 is a very fast one-stage detector, but it does sacrifice some performance to achieve these fast inference times.

Common for the networks is that a loss function is need during training in order to address the models performance in a given training point. A loss function commonly used for classification tasks is the cross entropy (1),

$$CE(p, y) = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \ , \tag{1}$$

where M is the number of classes, y is a binary indicator telling if class c is the correct classification for observation o, and $p_{o,c}$ is the predicted probability that observation o belongs to class c.

In a binary case, i.e when two classes are used, the cross entropy is defined as (2),

$$CE(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise} \end{cases} \tag{2}$$

In learning networks, a *loss function* is used to express the difference between actual and predicted output, say $\epsilon = y - \hat{y}$. One example of a loss function is the mean square error $J = \sum \epsilon_i^2$ that, however, has a very high penalty for outliers. For regression tasks, a smooth L_1 loss can be used,

$$smooth_{L_1}(\epsilon) = \begin{cases} 0.5\epsilon^2 & \text{if} |\epsilon| < 1 \\ |\epsilon| - 0.5 & \text{otherwise,} \end{cases} . \tag{3}$$

The smooth L_1 loss function in (3) is used in two of the CNN networks we employ.

RetinaNet RetinaNet is a convolutional network consisting of a *backbone* network and two subnetworks (Lin et al., 2018). The backbone network constructs a convolutional feature space from the input image. The backbone net can be any preferred convolutional network which means the RetinaNet architecture can accommodate a backbone CNN in an off-the-shelf fashion. The two subnetworks perform convolutional object classification and bounding box regression, respectively. The output of the backbone is arranged as a Feature Pyramid Network (FPN) in order to improve multi-scale predictions, see (Lin et al., 2017). RetinaNet makes use of anchor boxes to estimate the bounding box of an object, using a set of predetermined bounding box candidates of various sizes and aspect ratios. During training, an anchor is assigned to a ground truth box if intersection-over-union (IoU) between labelled and observed is above a threshold, here set to $h_{is-object} = 0.5$. If an anchor has an IoU less than $h_{is-background} = 0.4$ it is assigned to the background class. Anchors with an IoU between the two thresholds are ignored during training. If an anchor has been assigned to a ground truth, it will be fed to the regression subnetwork where the smooth L_1 loss is used to estimate the center coordinates of the bounding box together with its height and width.
Each level of the FPN will have 9 anchors assigned to each cell of the feature space. The anchors will have a size corresponding to the scale of the given level. RetinaNet uses five pyramid levels, with each level having its own pair of classification and regression subnetworks. The five pairs are concatenated such that the outputs corresponds to objects at all levels.

A salient new feature of the RetinaNet was the loss function, *Focal Loss*, used for classification, which allows better classification of background and distant objects in areas with dense objects or large differences between near and far away objects.

Rewriting the binary cross entropy (2),

$$\begin{aligned} CE(p, y) = CE(p_t) = -log(p_t), \\ p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

after which (Lin et al., 2018, p. 3) defined a *Focal Loss*,

$$FL(p_t) = -\alpha_t (1-p_t)^{\gamma} log(p_t), \tag{5}$$

where the factor $(1-p_t)^\gamma$ focuses the loss such that easy training examples are down-weighted, thus addressing the imbalance caused by the background class being over-represented, and α_t is a balancing factor.

For this article, RetinaNet was implemented using the Tensorflow Object Detection API (Huang et al., 2017). They find anchor sizes and aspect ratios using K-means clustering to accommodate the abundance of small objects in a dataset. The backbone network is ResNet-50 by (He et al., 2016). Training was initialized using weights trained on the COCO dataset (Lin et al., 2014).

YOLOv3 YOLOv3 is the latest addition to the YOLO-type networks, see (Redmon and Farhadi, 2018). YOLOv3 is, like RetinaNet, a one-stage detector, meaning that the input image is put through a network once yielding a set of predicted bounding boxes and classes. YOLOv3 consists of a backbone network for feature extraction and a number of sub-networks for bounding box regression and classification. Unlike RetinaNet, which uses an off-the-shelf feature extraction network, YOLOv3 uses a custom network called Darknet-53. Bounding box regression is done in a FPN-inspired way on three different scales. For each scale, the feature map is divided into a grid, on which three anchor boxes are assigned to each grid cell. The size and aspect ratio of the anchor boxes are predetermined using K-means on the dataset. Each anchor box with a ground truth IoU of above 0.5 is assigned to that ground truth box. Anchors with an overlap less than 0.5 is assigned to the background class. The appropriate anchor boxes are then sent to the regression network and classification network. YOLOv3 tries to take the class imbalance between the background and the ground truth into account by scaling down the loss by a set factor in case the current anchor box does not overlap a ground truth. For training, an open source implementation has been used (qqwweee, 2018). Training weights are initialized using weights trained on the COCO dataset (Lin et al., 2014). Due to the way the feature maps is divided into a grid, YOLOv3 only accepts square images. The inputs are therefore reshaped to 640×640 pixels during training.

Faster R-CNN One of the latest members of the family of region convolutional neural networks (R-CNN) is Faster R-CNN by (Ren et al., 2017). It builds upon the methods developed for the networks R-CNN by (Girshick et al., 2014) and Fast R-CNN by (Girshick, 2015). They all predict the contents of regions in images, by outputting bounding boxes and a prediction score for each candidate region fed to the networks.

The main difference between the different flavors of R-CNN are the methods by which interesting regions in the images are extracted. In R-CNN by (Girshick et al., 2014), regions of interest (ROI) are extracted using a selective search algorithm as proposed by (Uijlings et al., 2013). Each of these candidate regions are then fed to a CNN, which classifies the contents of these regions. Furthermore, a regression of the region is predicted to adjust the bounding box of the object contained in the candidate region. The selective search algorithm used in the R-CNN network makes inference times long, and as such this network is not suited for real-time applications.

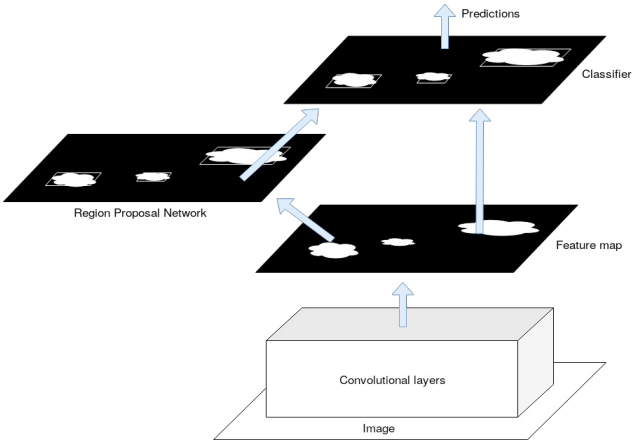

Fig. 2. Overview of the structure of the faster R-CNN, redrawn from (Ren et al., 2017).

Fast R-CNN by (Girshick, 2015) improves upon the method of generating ROI by reversing the order in which information is fed to the CNN. By feeding the input image directly to the CNN and extracting ROI from the output feature map, the number of inferences through the CNN can be reduced from 2000 in the original paper to one per image. The resulting ROI are warped into a fixed shape in order to pass them through the fully connected classification and regression layers. Although the selective search algorithm is still used to determine the best ROI, inference times are reduced by an order of magnitude in Fast R-CNN.

The Faster R-CNN, (Ren et al., 2017), aims to further improve inference time for the R-CNN networks, by determining ROI using a separate neural network, called the Region Proposal Network (RPN), instead of the selective search algorithm. This RPN is trained alongside the CNN used for generating the feature map, and the ROI generated from it are reshaped using ROI pooling to handle the fixed input requirement for the fully connected layers. This further decreases inference times by an order of magnitude, making the network usable in near real time applications. An overview of this network is shown in Figure 2.

In the RPN, a binary label can be assigned to each anchor at each position in the image corresponding to a position in the feature map. The value of the label is based on the intersection over union (IoU) with any of the ground truth boxes present in the image. A positive label is assigned to any anchor positioned such that its IoU with the ground truth box is larger than 0.7, i.e $IoU > 0.7$, and a negative label is assigned to anchors with $0.0 \leq IoU < 0.3$ with any ground truth box.

Anchors with an ambiguous IoU in the range $[0.3; 0.7]$ are not assigned any label, and therefore will not contribute to the loss of the RPN. The objective function sought to be minimized in the RPN of the Faster R-CNN is defined as (Ren et al., 2017),

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{6}$$

where i is the index of an anchor in a minibatch, p_i is the predicted label of an anchor, p_i^* is 1 if the anchor is a positive sample and 0 if it is a negative sample. t_i and t_i^* is a vector containing the coordinates of the predicted box and ground truth box for positive anchors, respectively.

The two terms in (6) are normalized by N_{cls}, the minibatch size and N_{reg}, which is the number of anchor locations. With available GPU memory, $N_{cls} = 1$ in this study. λ is a balancing parameter used to weigh the contributions of the two terms in the total loss score (6). λ was set such the the two terms contribute equally to the loss in (Ren et al., 2017).

The loss function used for the classifier term in (6), L_{cls} is the log loss, while the regression term, L_{reg}, uses the robust loss function defined in (Girshick, 2015):

$$L_{reg} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i - t_i^*), \tag{7}$$

where $smooth_{L_1}$, is the smooth L_1 loss function defined in (3).

The region proposals generated by RPN are then classified using the Fast R-CNN network. The training of the two networks are performed by alternating training, i.e. the RPN is trained first, and the proposals from this is then used to train the Fast R-CNN network. The Fast R-CNN network is then used to initialize the RPN, and so on.

For training the Faster R-CNN used in this paper, we used the images as is, i.e. as 640×480 pixels in size and we employed transfer learning on the ResNet-50 feature extractor. This particular version of ResNet-50 uses *atrous convolutions* to increase the receptive field of the features extracted. For anchors, sizes of 8, 16, 32, 64 and 96 pixels were used along with aspect ratios of 0.5, 1 and 2. A *stride* of 8 was chosen for the feature extractor, due to the small features present in our dataset.

To limit the number of region proposals, non-maximum suppression is used, such that we only get the 300 most significant detections from the RPN. Furthermore, only 100 detections are allowed per class per image to reduce class imbalance. For this article, Faster R-CNN was implemented using Tensorflow Object Detection API (Huang et al., 2017).

5. RESULTS

Comparing the methods, each network was evaluated on a common validation set, which was generated by randomly choosing 10% of the images in our dataset. Evaluation was done by computing precision and recall (8), and by mean average precision (9),

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN}, \end{aligned} \tag{8}$$

where TP denotes the sum of true positives in the validation set, i.e the number of instances correctly detected, FP the total number of false positives and FN are missed detections. A detection was deemed correct if a predicted bounding box and the true bounding box had an IoU above a given threshold. The predicted class should furthermore match that of the ground truth.

While evaluating, the classes were split into two depending on the pixel area. This was done as it was desired to detect an object as soon as possible when at sea. The area threshold for when an object is considered as small was set to 81 pixels. The threshold was determined by inspecting a histogram showing model detection versus pixel area, an example of which can be seen in Figure 4.

For method evaluation, it can be relevant to look at the precision-recall curves. A precision-recall curve is made by calculating the precision and recall of a model for a range of confidence and IoU thresholds, yielding a plot which shows the performance of a given model. A good model will have curves close to a precision and recall of 1, meaning that all data points were detected, and correctly classified. Plotting the precision-recall curve for a different set of IoU thresholds shows the methods ability to position the bounding boxes correctly.

Figure 3 show the precision-recall curves of the three models. It can be seen that Faster R-CNN is able to obtain a significantly better recall than the other methods with comparable precision. RetinaNet seems to have the overall lowest recall, and precision. YOLO lies somewhat in the middle between the two. These results are further supported by the results put forward in Table 6. Faster R-CNN is furthermore better at positioning bounding boxes relative to the ground truth, since the curves lie closer to each other across the IoU thresholds.

From the precision-recall curves, the average precision (AP) and mean average precision (mAP) can be computed. $AP(q)$ for object class q is obtained by integrating precision $p(r|q)$ as function of recall r, and mAB by averaging over Q object classes,

$$AP(q) = \int_0^1 p(r|q)dr, \; mAP = \frac{1}{Q}\sum_{q=1}^{Q} AP(q). \tag{9}$$

The mA was calculated for three different IoU thresholds, i.e. a detection has to have an IoU larger than the threshold to be counted as a true positive. The mAP results are presented in Table 3.

Table 3. Mean average precision

	F-RCNN	R-Net	YOLOv3	R-Net-U
mAP@0.25 IoU	0.93	0.96	**0.97**	0.94
mAP@0.50 IoU	0.81	0.86	**0.90**	**0.90**
mAP@0.75 IoU	0.32	0.33	0.29	**0.40**

In terms of mAP, the performance of the networks are quite similar at lower IoU thresholds. However, the upscaled RetinaNet is better at placing the bounding boxes, as shown by the higher mAP for the 0.75 IoU threshold.

The performance of the models on the different classes can be further investigated via their confusion matrices. The layout of such a matrix is shown in Table 4.

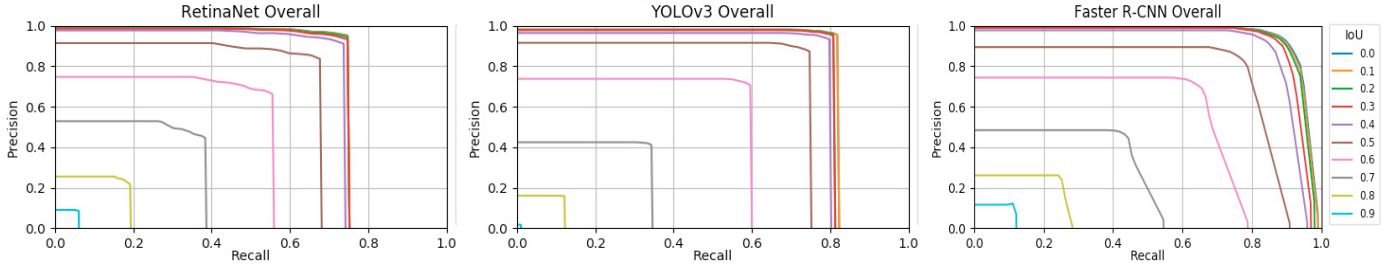

Fig. 3. Average precision-recall curves for RetinaNet, YOLOv3 and Faster R-CNN.

Table 4. Sample confusion matrix

Class		
	TP	FP
	FN	TN

where *TN* will be left out, as this metric is ill defined for the bounding box prediction problem.

The confusion matrices are computed using an IoU threshold of 0.3 and a classification confidence threshold of 0.6. A relatively low IoU threshold is chosen to favor a high recall, especially for small objects, as a small positioning error will result in a large change in IoU due to the relatively large ratio between the size of a pixel and the size of a small object.

Table 5 shows the resulting confusion matrices. Inspection shows that Faster R-CNN performs the best in all categories with both YOLOv3 and RetinaNet having difficulties with smaller objects, which is evident from Table 6, when looking at the precision and recall for the two classes. Average inference time for single images, using a Geforce GTX 1080 GPU, is summarized in Table 7.

Table 5. Confusion matrices for CNN models

	Faster R-CNN		RetinaNet		YOLOv3	
Buoy						
	204	15	176	3	198	0
	16	-	44	-	22	-
Small						
	329	19	181	2	291	21
	67	-	215	-	105	-
Boat						
	1717	70	1536	100	1496	5
	53	-	234	-	274	-
Small						
	873	40	435	17	801	26
	222	-	660	-	294	-

Table 6. Precision (P) and recall (R)

	F-RCNN	R-Net	YOLOv3	R-Net-U
Average P	0.96	0.95	**0.98**	**0.98**
Average R	**0.90**	0.67	0.80	0.86
Buoy P	0.93	0.98	**1.00**	0.99
Buoy R	0.93	0.80	0.90	**0.95**
Small buoy P	0.95	**0.99**	0.93	0.95
Small buoy R	**0.83**	0.46	0.73	0.76
Boat P	0.96	0.94	**1.00**	0.99
Boat R	**0.97**	0.87	0.85	0.92
Small boat P	0.96	0.96	**0.97**	0.96
Small boat R	**0.80**	0.40	0.73	0.78

Table 7. Average inference time on GTX 1080 GPU

	FR-CNN	R-Net	YOLOv3	R-Net-U
Inference time	459 ms	**40 ms**	74 ms	120 ms

Model performance is also illustrated by detection histograms of detected and not detected objects as function of bounding box area. Figures 4, 5 and 6 present such histograms for the three networks. It is seen that the majority of objects in the validation set have a bounding box area of less than 250 pixels and that most of the lower recall of RetinaNet comes from missed detections of objects with an area less than 250 pixels, after which RetinaNet obtains a better recall than YOLOv3. Faster R-CNN achieves high recall and precision for both small and large objects.

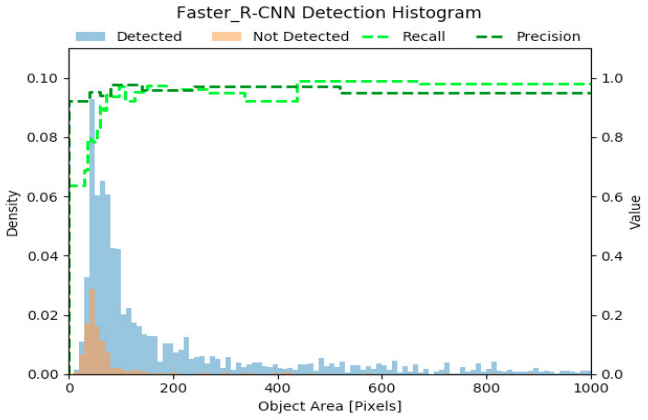

Fig. 4. Detection histogram for Faster R-CNN

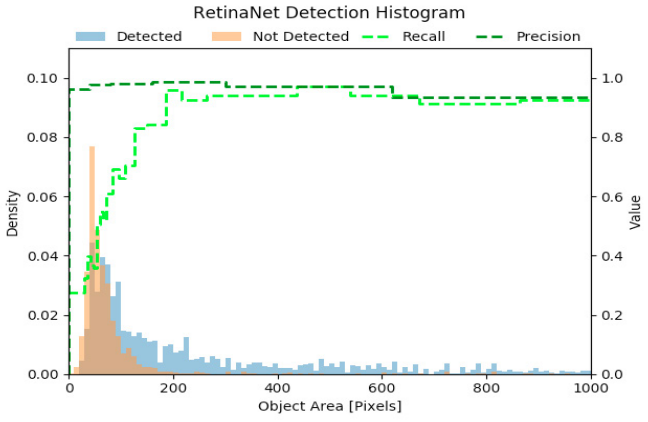

Fig. 5. Detection histogram for RetinaNet

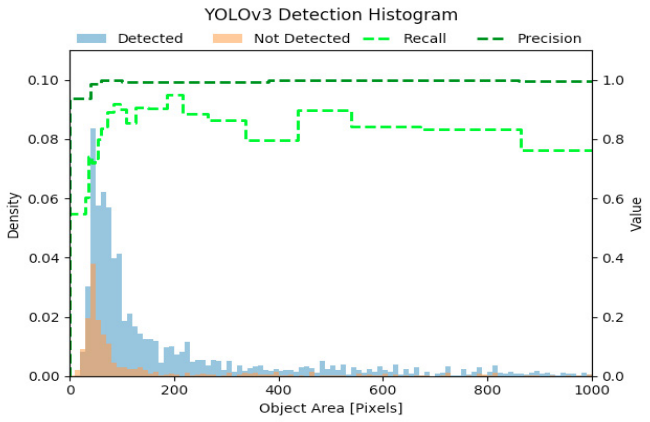

Fig. 6. Detection histogram for YOLOv3

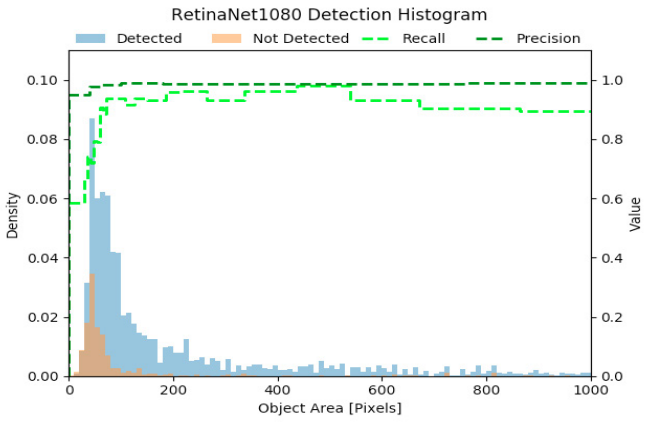

Fig. 7. Detection histogram for RetinaNet with upscaled images

6. DISCUSSION

The performance of the three networks considered in this paper was presented above. Of the three, Faster R-CNN achieved the best overall recall, while YOLOv3 obtained a slightly better precision than the two other networks.

In an autonomous navigation setting, the most important metric to consider is the recall, as it is better to detect a misclassified object, than to miss it altogether. With this in mind, Faster R-CNN is the best performing network. The good performance of Faster R-CNN comes at a price. The average inference time of this network is approximately 5 times that of YOLOv3, and approximately 10 times that of RetinaNet. This added computation time needs to be taken into account, as tracking of objects detected is necessary in an autonomous navigation situation. An inference time of half a second could hamper tracking of objects moving in close proximity to own vessel, or at high velocities relative to own vessel.

Investigation of the poor recall performance of RetinaNet was conducted. One of the issues identified was poor performance on small objects, which is due to the downsampling of the input images conducted in the ResNet-50 network. The first level in the FPN has a downsampling factor of 8 (Lin et al., 2017), which results in fewer features being extracted from small objects. It should therefore be possible to obtain better results by upsampling the input images to e.g 1440 × 1080 pixels and thus counteracting the downsampling of the ResNet-50 network.

A detection histogram for a RetinaNet model trained on 1440×1080 pixel images is presented in figure 7. The model is trained using the RetinaNet implementation found in (Gaiser et al., 2018). It is noted that the upscaled images have resulted in a model with a precision and recall comparable with that of Faster R-CNN, while keeping a relatively low average inference time of 120 ms.

The consequence of missing a detection of an object in a navigation setting could be adverse. The recall performance of the networks shows that they cannot be used for stand-alone detection in an navigation setting. Object tracking over several frames could significantly improve the performance.

7. NAVIGATION PERSPECTIVES

From a navigation perspective, an object requires attention when it is within a range of 5-10 times ship length and has a predicted closest point of approach of less than 3-5 times ship length. The distance of attention and distance of action depend on vessel size and speed.

For vehicles with AIS or a Radar signature, distances are given. For camera-based imaging, objects can be detected with a probability described by the recall values. With object size in image s^{obj}_{image}, object physical size s^{obj}_{phys}, focal length f_{lens}, sensor pixel size s_{sensor} and distance to an object, d^{object}_{phys} ,

$$d^{object}_{phys} = s^{obj}_{phys} \frac{f_{lens}}{s^{obj}_{image} s_{sensor}}. \tag{10}$$

With minimum pixel size for detection s^{min}_{image}, an object can be detected at a maximum distance, d^{object}_{detect} ,

$$d^{object}_{detect} = s^{obj}_{phys} \frac{f_{lens}}{s^{min}_{image} s_{sensor}}. \tag{11}$$

The LWIR camera has a pixel size of $s_{sensor} = 17\mu m$, 90° HFOV and focal length $f_{lens} = 7.5mm$.

According to Figure 7, the upscaled RetinaNet has recall 85% for pixel area 50. This area represents a diagonal of 10 pixels. A leisure boat of dimensions height $20m$, width $3.5m$ and length $10m$, has a diagonal of $20.3m$ if seen from bow or stern, and $22.6m$ in athwart ship view. A leisure boat with such characteristic dimension can hence be detected at a distance of 850 m. The recall of 85% will mean that detection will take place in most frames but there will be frames with no detection. Adequate frame rate and object tracking will be able to alleviate this.

A vessel on oar with characteristic dimension $s^{obj}_{phys} = 1.5m$ could first be detected when $65m$ away from own ship with the 90° lens, which would give too short notice to navigate safely. Longer focal length, detection using daylight camera, or use of detection methods other than CNN technology, e.g. edge detection methods, would cope with detection in time for such small objects.

8. CONCLUSION

Three convolutional neural network architectures has been assessed for object detection and classification on long wave infrared images acquired at sea. It was shown that detection and classification of objects can be done with high precision while detecting $\sim 90\%$ of all objects. The off the shelf CNN technology uses down-scaling, which limits detection of far away objects. It was found that a good compromise between detection speed and precision/recall could be obtained by upscaling the input images.

The present investigation does not argue for only using LWIR imaging, but was an effort to compare off the shelf CNN technologies. Colour information is needed as well to interpret navigation lights at night and to decode colour codes on buoys and signal flags.

ACKNOWLEDGEMENTS

This research was sponsored by the Danish Maritime Foundation, Orients Fund and LauritzenFonden through the ASAN project.

REFERENCES

Blanke, M., Hansen, S., Stets, J.D., T. Koester AND, J.E.B., Maurin, A.L., Nykvist, N., and Bang, J. (2019). Outlook for navigation – comparing human performance with a robotic solution. In *Proc. 1st International Conference on Maritime Autonomous Surface Ships (ICMASS 2018)*. SINTEF Academic Press, Trondheim Norway. ISBN 978-82-536-1628-5.

Bloisi, D.D., Pennisi, A., and Iocchi, L. (2014). Background modeling in the maritime domain. *Machine vision and applications*, 25(5), 1257–1269.

Bousetouane, F. and Morris, B. (2016). Fast CNN surveillance pipeline for fine-grained vessel classification and detection in maritime scenarios. In *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th Ieee Int. Conf. On*, 242–248. IEEE.

Fefilatyev, S., Goldgof, D., Shreve, M., and Lembke, C. (2012). Detection and tracking of ships in open sea with rapidly moving buoy-mounted camera system. *Ocean Engineering*, 54, 1–12.

Gaiser, H., de Vries, M., Lacatusu, V., Carpani, V., Williamson, A., Liscio, E., and ... (2018). fizyr/keras-retinanet: 0.5.0. doi:10.5281/zenodo.1464720.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. Ieee Conf on Computer Vision and Pattern Recognition*. doi:10.1109/CVPR.2014.81.

Girshick, R. (2015). Fast r-cnn. In *Proc. IEEE Int. Conference on Computer Vision*. doi:10.1109/ICCV.2015.169.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 770–778. doi:10.1109/CVPR.2016.90.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. doi:10.1109/cvpr.2017.351.

Leclerc, M., Tharmarasa, R., Florea, M.C., Boury-Brisset, A.C., Kirubarajan, T., and Duclos-Hindié, N. (2018). Ship classification using deep learning techniques for maritime target tracking. In *2018 21st International Conference on Information Fusion (FUSION)*, 737–744. IEEE.

Leira, F.S., Johansen, T.A., and Fossen, T.I. (2015). Automatic detection, classification and tracking of objects in the ocean surface from UAVs using a thermal camera. In *Aerospace Conference, 2015 IEEE*, 1–10. IEEE.

Lin, T., Goyal, P., Girshick, R., He, K., and Dollar, P. (2018). Focal loss for dense object detection. *Ieee Transactions on Pattern Analysis and Machine Intelligence*. doi:10.1109/TPAMI.2018.2858826.

Lin, T.Y., Dollar, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE. doi:10.1109/cvpr.2017.106.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C.L. (2014). Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, 740–755. Springer International Publishing, Cham. doi:10.1007/978-3-319-10602-1-48.

Moreira, R.D.S., Ebecken, N.F.F., Alves, A.S., Livernet, F., and Campillo-Navetti, A. (2014). A survey on video detection and tracking of maritime vessels. *International Journal of Recent Research and Applied Studies*, 20(1).

Prasad, D.K., Rajan, D., Rachmawati, L., Rajabally, E., and Quek, C. (2017). Video processing from electro-optical sensors for object detection and tracking in a maritime environment: a survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8), 1993–2016.

qqwweee (2018). qqwweee/keras-yolo3. URL `https://github.com/qqwweee/keras-yolo3`.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. Technical Report arXiv:1804.02767 [cs.CV], Cornell University, Computer Science, Computer Vision and Pattern Recognition [cs.CV].

Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39. doi:10.1109/TPAMI.2016.2577031.

Rodin, C., De Lima, L., De Alcantara Andrade, F., Haddad, D., Johansen, T., and Storvold, R. (2018). Object classification in thermal images using convolutional neural networks for search and rescue missions with unmanned aerial systems. In *Proc. Int. Joint Conference on Neural Networks*. doi:10.1109/IJCNN.2018.8489465.

Rodin, C.D. and Johansen, T.A. (2018). Detectability of objects at the sea surface in visible light and thermal camera images. In *Proc. of Oceans 2018*.

Uijlings, J.R.R., Sande, K.E.A., De, V., Gevers, T., and Smeulders, A.W.M. (2013). Selective search for object recognition. *Int. Journal of Computer Vision*, 104, 154–171. doi:10.1007/s11263-013-0620-5.

Zhang, S., Wen, L., Bian, X., Lei, Z., and Li, S. (2018). Single-shot refinement neural network for object detection. doi:10.1109/CVPR.2018.00442.