Investigating the Transferability of TOG Adversarial Attacks in YOLO models in the Maritime Domain

Manasut, Phornphawit; Ibtasham, Md Saleh; Yaradanakul, Zeynep; Azimi, Sepinoud; Lafond, Sebastien; Iancu, Bogdan

DOI
10.1109/ACCESS.2025.3589512

Publication date
2025

Document Version
Final published version

Published in
IEEE Access

Citation (APA)
Manasut, P., Ibtasham, M. S., Yaradanakul, Z., Azimi, S., Lafond, S., & Iancu, B. (2025). Investigating the Transferability of TOG Adversarial Attacks in YOLO models in the Maritime Domain. *IEEE Access*, *13*, 126444-126464. https://doi.org/10.1109/ACCESS.2025.3589512

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 13 June 2025, accepted 7 July 2025, date of publication 15 July 2025, date of current version 23 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3589512

RESEARCH ARTICLE

Investigating the Transferability of TOG Adversarial Attacks in YOLO Models in the Maritime Domain

PHORNPHAWIT MANASUT[1], MD SALEH IBTASHAM[1], ZEYNEP YARADANAKUL[1], SEPINOUD AZIMI[2], SÉBASTIEN LAFOND[1], AND BOGDAN IANCU[1]
[1]Department of Information Technology, Åbo Akademi University, 20500 Turku, Finland
[2]Department of Technology, Policy and Management, Delft University of Technology, 2628 CD Delft, The Netherlands

Corresponding author: Bogdan Iancu (Bogdan.Iancu@abo.fi)

The work has been partially supported by the Erasmus Mundus Joint Master Degree (EMJMD) master's programme in Engineering of Data-Intensive Intelligent Software Systems (EDISS - European Union's Education, Audiovisual and Culture Executive Agency grant number 619819).

ABSTRACT In recent years, CNN-based object detectors have been widely adopted in autonomous systems. Although their capabilities are employed across various industries, these detectors are inherently susceptible to adversarial attacks. Despite extensive studies on their effects on image classification, adversarial attacks remain largely unexplored in object detection. In particular, we note the reduced number of studies employing benchmarks for these types of attacks. Object detectors can be easily deceived by adding carefully devised perturbations to their inputs, rendering them unreliable. This study investigates the transferability of one such adversarial attack type, the Targeted Objectness Gradient (TOG), on different variations of the YOLO architecture to formally assess its vulnerability under different scenarios in the maritime domain. To investigate the significance of TOG adversarial attacks across variations of YOLO architectures and combinations of maritime datasets (all publicly available), we conducted a statistical analysis of black-box and white-box attacks. Our research questions were formulated to address a range of concerns that encompass various complexities to be considered in the detection of maritime objects. Our presented results underline the transferable nature of TOG adversarial attacks and the compelling need to benchmark such attacks in the maritime object detection domain.

INDEX TERMS Maritime object detection, adversarial attacks, targeted objectness gradient (TOG) attack, adversarial attack transferability.

I. INTRODUCTION

Object detection and classification alike have gained prominence over the past decade due to tremendous advances in deep learning. The advent of deep convolutional neural networks (DCNNs) was brought about by the launch of AlexNet at the ImageNet Large-Scale Visual Recognition Challenge in 2012, which caused a paradigm shift in computer vision [24]. Among the most prominent object detection models, the You Only Look Once (YOLO) family has become popular because of its impressive speed-accuracy trade-off within the tasks of object detection and tracking [21].

However, CNN-based detectors are susceptible to adversarial attacks [56]. Adversarial attacks refer to the deliberate manipulation of input images by adding imperceptible perturbations, called adversarial samples, which can mislead models and cause them to make incorrect detections. These attacks pose a significant threat to the robustness and reliability of object detection systems, particularly in critical domains such as autonomous navigation. Upon closer inspection, an even more notable observation was made: adversarial samples devised to attack a specific architecture proved to be successful in deceiving other, even dissimilar ones [9], [59].

The associate editor coordinating the review of this manuscript and approving it for publication was Yeon-Ho Chung.

Although adversarial attacks have been extensively studied in the context of image classification, research on their impact on object detection models, particularly in the maritime domain, remains limited. Maritime environments present unique challenges for object detection owing to factors such as complex backgrounds, occlusions, varying lighting conditions, and the presence of water reflections. Understanding the transferability of adversarial attacks in the maritime domain is crucial for assessing the security and reliability of object detection systems used in applications such as maritime surveillance, navigation assistance, and autonomous navigation. Transfer attacks are adversarial attacks in which the source and target models differ in terms of the model architecture or training dataset.

In this study, we investigated the transferability of adversarial attacks on YOLO models in the maritime domain. We evaluated the effectiveness of adversarial attacks on YOLO models trained on three publicly available maritime datasets and all combinations thereof. By evaluating the vulnerability of the YOLO models to adversarial attacks, we can gain insights into the robustness of these models in maritime scenarios and identify potential security risks. To achieve our objectives, we employed Targeted Objectness Gradient (TOG) attacks and utilized a black-box attack scenario in which the attacker has no access to the model's internal parameters and can only manipulate the adversarial sample. Hence, we investigated potential attack combinations across multiple YOLO architectures. To determine whether the YOLO architecture demonstrates superior resilience compared to other architectures when faced with these attacks, we employed various statistical tests. Our findings not only reveal the robustness of various YOLO models but also offer valuable insights into potential strategies for reinforcing their security.

Contributions. We assessed all possible attack combinations using multiple YOLO architectures and datasets. We performed statistical analyses to investigate a variety of research questions pertaining to determining whether any YOLO architecture is generally better at generating adversarial perturbations or whether any dataset presents a clear statistical advantage in training the surrogate model.

Organization. The remainder of this paper is organized as follows. In Section II, we provide essential background information on object detection in general, various iterations of the YOLO architecture, adversarial attacks, and a detailed depiction of the Targeted Objectness Gradient (TOG) attack. Section III offers a detailed account of our experimental methodology and describes the experimental design. Section IV reveals the results of our investigation, providing a comprehensive analysis of the findings. Section V presents a discussion regarding the practical signification of TOG attacks and the feasibility of deploying them in real maritime environments, possible defense strategies and the implications for autonomous maritime navigation. The conclusion is presented in Section VI. Section VII describes the threats to the validity and reliability of the study.

II. BACKGROUND

A. OBJECT DETECTION

Object detection, image classification, and image segmentation are distinct computer vision tasks. Object detection, specifically, involves identifying and localizing one or multiple object types within an image, providing information about each object's class and its bounding box coordinates. Image classification, on the other hand, focuses on assigning a single label or class to an entire image, without specifying the object's location or count. Image segmentation partitions an image into regions at the pixel level, subdividing an image into its constituent parts, and extracting the parts of interest [60].

More specifically, object detection is a supervised learning problem that involves identifying all instances of predetermined object classes in an image and providing coarse localization using axis-aligned boxes [62]. Object detectors play a pivotal role in computer vision and can be broadly classified into two main groups: two-stage and one-stage detectors. Two-stage detectors aim to locate objects in the image during the first stage and then classify and localize them during the second stage [58]. Two-stage detectors such as Faster R-CNN [14] utilize a region proposal network (RPN) to suggest candidate bounding boxes, followed by a second stage for classification and regression of the aforementioned bounding boxes. These models prioritize accuracy, but are computationally intensive.

In contrast, one-stage detectors streamline the process by directly classifying each region of interest as background or an object of interest, making them more computationally efficient while maintaining a fair accuracy. Keypoint-based detectors, such as CornerNet [26], predict keypoints to generate bounding boxes, whereas center-based detectors, such as the YOLO series and Centernet [61] determine object locations based on centers, offering alternatives with distinct trade-offs in accuracy and efficiency [18]. For our experiments, we focused on the performance of single-stage detectors, specifically those from the YOLO family [52].

Detecting vessels in the maritime environment is of paramount importance for the development and operation of autonomous ferries and other autonomous waterborne vehicles. As autonomous cars rely on sensors such as radar to perceive their surroundings, autonomous ferries must have the ability to sense and understand the maritime environment for safe navigation and efficient operation. Vessel detection is critical for ensuring collision avoidance, route planning, and coordination with other ships and maritime traffic. The use of deep learning-based detection algorithms allows maritime vessels to effectively use cameras and other sensors to collect vital information about their surroundings, thus contributing to the evolution and success of autonomous maritime transport [15].

Sensor fusion is essential for advancing the development of reliable autonomous navigation systems, particularly in challenging environments, such as the maritime domain. Integrating data from multiple sensors, such as radar, LiDAR, RGB cameras, and IR cameras, enhances the ability to

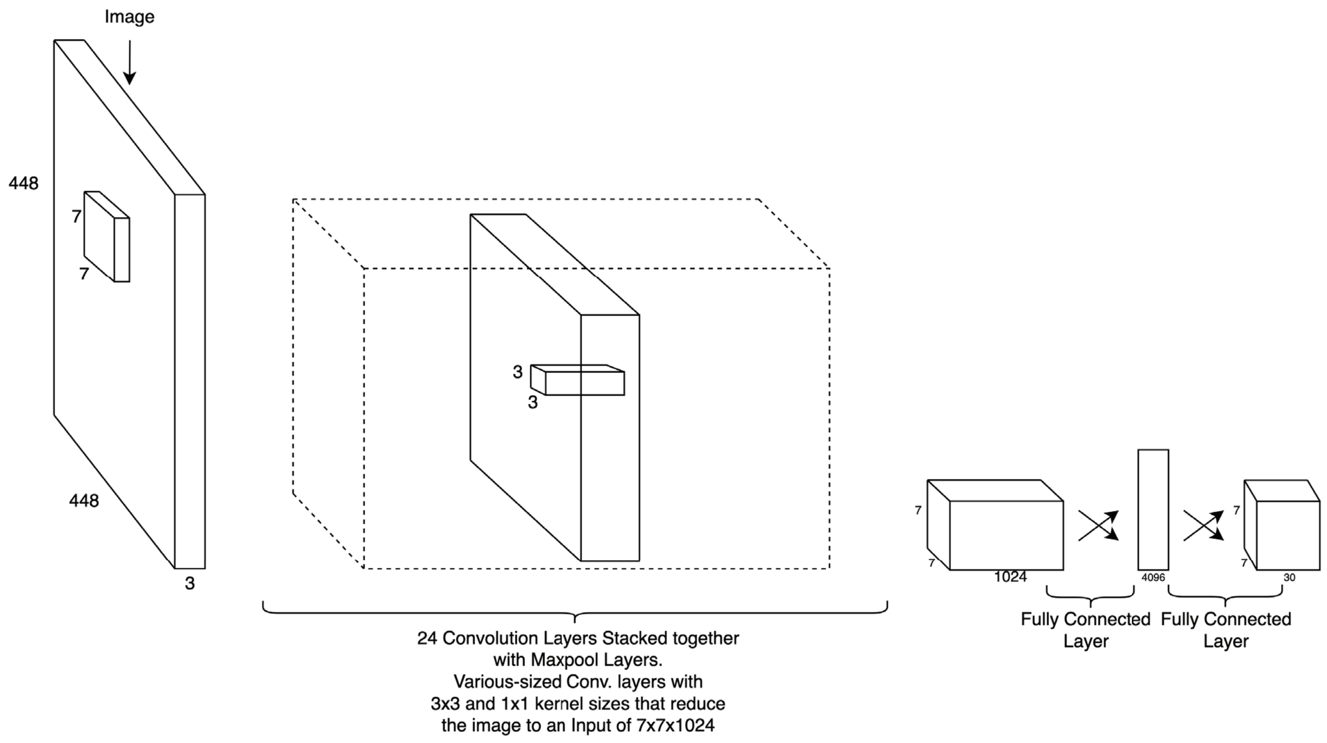


Image
448
7
7
448
3
3
3
24 Convolution Layers Stacked together with Maxpool Layers. Various-sized Conv. layers with 3x3 and 1x1 kernel sizes that reduce the image to an Input of 7x7x1024
7
7
1024
4096
7
7
30
Fully Connected Layer
Fully Connected Layer

FIGURE 1. The original YOLO detection framework [40] consists of 24 convolutional layers followed by 2 fully connected layers. The (3 × 3) and (1 × 1) convolution layers offer feature reduction from previous layers. The network takes a (448 × 448) input image and then produces a (7 × 7 × 30) tensor of predictions.

achieve robust object detection, even under dynamic and diverse conditions. This approach overcomes the limitations of individual sensors, with radar providing distance and velocity information and cameras offering high-resolution data. The fusion of these sensor outputs, aided by techniques such as Probabilistic Data Association (PDA) and Convolutional Neural Networks (CNNs), results in more accurate and comprehensive object detection, which is essential to ensure safe navigation and intelligent decision-making for autonomous vessels and vehicles in complex environments [16].

B. THE YOLO OBJECT DETECTOR

Real-time object detection plays a vital role in various fields, including autonomous navigation, robotics, video surveillance, and augmented reality [52]. The You Only Look Once (YOLO) framework [40] has gained prominence owing to its impressive balance of speed and accuracy in the identification of objects in images. The YOLO family has evolved through multiple iterations, progressively improving performance and addressing various limitations. Throughout this evolution, the trade-off between speed and accuracy has been a recurring theme, with each version striving to optimize these competing objectives in different ways.

The original YOLO model [40] prioritized fast object detection using a single convolutional neural network (CNN) to simultaneously predict object locations and classes. The general structure of the single-stage CNN is illustrated in Figure 1. However, this focus on speed came at the cost of accuracy, especially when detecting small or overlapping objects. Subsequent YOLO versions introduced refinements, such as anchor boxes and pass-through layers (e.g., in YOLOv2: YOLO9000) [41], as a measure to enhance object localization and thus improve overall accuracy. YOLOv3 [42] further improved performance by using a multiscale feature extraction architecture for effective detection across different scales. As the YOLO framework progressed, models such as YOLOv4 [5] and YOLOv5 [22] introduced other innovations, such as new network backbones, improved data augmentation techniques, and optimized training strategies, leading to significant accuracy gains while maintaining real-time performance. Commencing with Scaled-YOLOv4 [55], official YOLO models have fine-tuned the speed-accuracy trade-off by offering different model scales tailored to specific applications and hardware requirements. These versions often provide lightweight models optimized for edge devices, sacrificing accuracy for reduced computational complexity and faster processing.

YOLOv1 performs the object detection task entirely as a regression task. First, it splits the input image into an $S \times S$ grid and predicts B bounding boxes per grid. A grid cell is responsible for detecting an object if the center of the object falls within its boundaries, predicting B bounding boxes along with confidence scores that indicate both the likelihood of an object being present and the accuracy of the predicted box. For each bounding box, it outputs a quintuple $(P_c, b_x, b_y, b_h, b_w)$ which, respectively, are the confidence score of the bounding box, the center b_x and center b_y values of the bounding box relative to the grid cell, and finally the height and width of the bounding box b_h and b_w, respectively. The confidence score P_c is defined as the intersection over union (IoU) between the predicted bounding box and the ground-truth, considering that the object is in the cell, otherwise 0. The representation of the bounding box is reformulated as $(t_x, t_y, t_w, t_h, t_o)$, where predictions are made relative to the center of the grid cell (c_x, c_y) and the previous dimensions of the bounding box p_w and p_h. This approach, inspired by YOLOv1, constrains the predicted coordinates to a normalized range using the logistic activation function. By incorporating these transformations, the model ensures that the predicted bounding boxes align with the YOLOv1 format $(P_c, b_x, b_y, b_h, b_w)$. The specific relationships between these parameters are defined as follows. In addition, it also includes the confidence score of the C classes. Overall, it produces a tensor of $S \times S \times (B \times 5 + C)$. Intuitively, it can be understood that YOLO performs a prediction per grid cell to get the best bounding box with the most likely class of that grid [40].

From YOLOv2 onward, this prediction output changes to $S \times S \times (B \times 5 \times C)$ with the introduction of anchor boxes. Instead of predicting B bounding boxes per grid from scratch, the model's predictions are based on anchors. This modification addresses a key limitation of YOLOv1, which struggles to detect bounding boxes with unseen dimensions during training. Moreover, instead of relying on confidence scores assigned per cell for each class, the YOLOv2 architecture predicts a confidence score for each class directly for every bounding box. Additionally, instead of predicting class confidence scores at the grid cell level, the model now predicts class probabilities for each bounding box independently, scaling them by the predicted objectness score to produce final class scores. The representation of the bounding box is reformulated as $(t_x, t_y, t_w, t_h, t_o)$, where predictions are made relative to the center of the grid cell (c_x, c_y) and the dimensions of the prior bounding box p_w and p_h. This approach, inspired by YOLOv1, constrains the predicted coordinates within a normalized range using a logistic activation function. By incorporating these transformations, the model ensures that the predicted bounding boxes align with the YOLOv1 format $(P_c, b_x, b_y, b_h, b_w)$. The specific relationships between these parameters are defined as follows:

$$\begin{aligned} P_c &= t_o \\ b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \end{aligned}$$

where σ represents the sigmoid function.

As there can be many duplicate bounding boxes, a non-maximum suppression algorithm is applied to the candidate bounding boxes produced by the YOLO models [52]. Given a set of predicted bounding boxes $PredB$ and their confidence scores S, IoU threshold τ and confidence threshold T, the algorithm operates as follows:

1) Filter out bounding boxes with confidence score lower than T from $PredB$.
2) Remove the current highest confidence bounding box b from the set.
3) Compare b to the rest of the bounding boxes in the set and remove the boxes whose IoU with b is above τ.
4) Add b to the filtered set.
5) Repeat steps 2-4 until $PredB$ is $\emptyset$ (empty).

In YOLOv3, the confidence score t_0 is changed to an objectness score and is predicted using logistic regression, where the score is 1 for an anchor box that best overlaps with the ground-truth and 0 for the rest. Hence, only one anchor is assigned to each ground-truth object. In the absence of ground-truth, this is treated as a classification loss. In addition, the class prediction scheme was changed to logistic classifiers trained using binary cross-entropy, converting the problem into multiclassification to allow multilabel assignment to a box.

Subsequently, Ultralytics developed its own versions of YOLOv5 and YOLOv8, which contain further improvements from YOLOv3 and YOLOv4. The Ultralytics implementation introduced an algorithm called AutoAnchor, which adjusts the anchors based on the dataset itself and the training settings (image size, learning rate, etc.). To generate new anchors, AutoAnchor uses K-means to generate initial conditions for a Genetic Evolution algorithm to evolve anchor boxes to be a better fit for the specific dataset and training settings. Hence, better anchors typically indicate a better model performance. Although YOLOv5 maintained the use of anchors, YOLOv8 removed the use of these anchors and made a few adjustments to the convolutional backbone of its architecture. Finally, Ultralytics continually developed these two models to allow them to perform vision tasks, such as image classification, pose estimation, and image segmentation, in addition to object detection.

The loss function of the YOLO models can be divided into three main components: localization, confidence, and classification losses. These three components sum up to five terms, as follows:

The following two terms represent the localization loss (LL):

$$\begin{aligned} LL = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{PredB} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x})^2 + (y_i - \hat{y}_i)^2] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{PredB} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2], \end{aligned} \quad (1)$$

where S^2 refers to the set of all grid cells and $PredB$ refers to the set of predicted bounding boxes, and $\mathbb{1}_{ij}^{obj}$ indicates the presence of an object in the i^{th} grid cell and the j^{th} bounding box. For the first term (location), the sum-squared error is applied to calculate the loss, whereas for the second term (width and height), the square root is applied to the width and height before the sum-squared error is calculated to reduce the range of values.

The next two terms represent confidence loss (COL), and are associated with the confidence score of each bounding box. Specifically, this loss is computed from a single bounding box within a grid cell, even if the object is not present in that cell according to the ground-truth, as follows:

$$COL = \sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{1}_{ij}^{obj}[(C_i - \hat{C}_i)^2] + \lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj}[(C_i - \hat{C}_i)^2] \tag{2}$$

where C_i is the confidence score of the ground-truth, $\hat{C}_i$ denotes the confidence score predicted by the model, $\mathbb{1}_{ij}^{noobj}$ is 1 if there is an object in the i^{th} cell and 0 otherwise. Since there are many more predicted no-object boxes, λ_{noobj} is applied to reduce the loss contributed by no-object boxes, and the default value is 0.5. For both terms, the sum-squared error is applied to calculate the loss.

Finally, the last term represents the classification loss (CLL):

$$CLL = \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} [(p_i(c) - \hat{p}_i(c))^2] \tag{3}$$

where for each i predicted bounding box with an object, the squared error of the conditional probability of the ground-truth and predicted box is calculated and summed up for each class c.

In our experiments, two variations of each YOLO version were used. For YOLOv3, Darknet-53, which has 53 convolutional layers, was used as the backbone to extract features [2], [42]. Its tiny counterpart reduces the depth of these convolutional layers by replacing them with multiple max-pooling layers and reduces the number of upsamplings in the detection head from three to two [2].

For both YOLOv5 and YOLOv8, the differences between the 'x' and 'n' variations are the multipliers for the width and height and the maximum number of channels for all layers. This persists with other variations such as 's', 'm', and 'l'. Unlike YOLOv3, the general arrangement of the modules remains the same regardless of the scale. For example, the CBS modules in the YOLOv5 stack Conv, BatchNorm, and SiLU layers together [31]. The size of this type of module differs for each variation in terms of depth, width, and maximum number of channels.

C. ADVERSARIAL ATTACKS

Adversarial attacks severely compromise the efficiency and reliability of object detectors. This inherent vulnerability hinders the widespread adoption of object detection applications in crucial domains, including autonomous vehicles, where reliability is paramount [4]. The investigation of adversarial attacks in the maritime domain has been relatively limited. Although significant advances have been made in the field of adversarial robustness, most research has focused predominantly on image classification, due to its relatively simpler theoretical and practical nature. In contrast, adversarial attacks on object detection remain comparatively understudied because of the intricate architectures of the detectors themselves.

However, there are different ways to categorize adversarial attacks in terms of object detection, but the most common method is based on whether the attacker has access to the model and whether specific targets are available for the attack. Attacks can be divided into two types based on whether the attacker has access to the model: white-box and black-box attacks [3], [7]. White-box attacks [38] occur when the attacker has access to a model's parameters. In contrast, black-box attacks [33] occur when the attacker only has access to the network inputs and outputs of the attacked detector, without any knowledge of its internal parameters.

To attack object detectors, the general approach is to introduce perturbations embedded in a clean image to generate adversarial samples. At a high level, attacks can be classified according to their targets into targeted and untargeted attacks. Targeted attacks refer to adversarial attempts to mislead the model toward a specific target label, whereas untargeted attacks only aim to cause the model to make incorrect predictions without any specific label [45].

D. TRANSFERABILITY OF ADVERSARIAL ATTACKS

Although adversarial examples elicited some apprehension regarding the robustness and reliability of ML systems, an even more interesting phenomenon arised: the transferability of adversarial examples. Transferability describes the capability of an adversarial example, created for one model, to effectively deceive a new model, with a different architecture. This poses a significant threat to the security of deep learning-based systems. Adversarial examples can effectively deceive models beyond those for which they were initially designed. This characteristic enables attackers to exploit vulnerabilities in deep neural networks without requiring specific knowledge of the target system [57].

Adversarial attacks can take advantage of the assumption that, despite variations in training data, models designed to detect the same objects share commonalities that can be exploited. This is particularly useful when the attacker lacks direct access to the target model [49].

For instance, a study explored cross-dataset transference by training Darknet models on different training sets and using the TOG framework for adversarial set generation [49]. This study measured the attack performance of both the source and target models, highlighting the impact of dataset intersection. The models were trained on varied intersection sizes, showing a correlation between attack transference and dataset intersection in the experimental results.

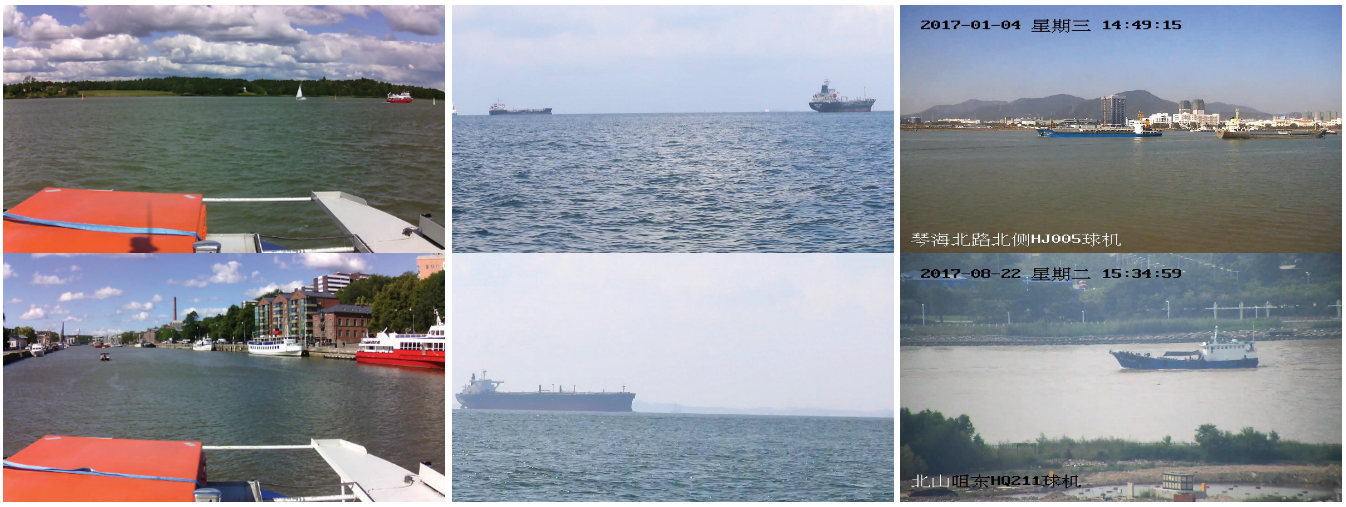


FIGURE 2. Example images from the 3 base datasets: ABOships-PLUS [18] (left column), SMD-Plus [23] (middle column), and SeaShips [46] (right column).

E. TARGETED OBJECTNESS GRADIENT ATTACKS (TOG)

Targeted Objectness Gradient (TOG) attacks are a family of adversarial attacks developed to directly target object detectors [4]. The TOG attack iteratively exploits the gradients of the model to generate perturbations in an image [8]. Each attack in this family uses different components of an object detector's loss function to achieve its goal. All TOG attacks conform to the following general form:

$$x'_{t+1} = \prod_{x,\epsilon}[x'_t - \alpha\Gamma(\frac{\partial L^*(x'_t; O^*, W)}{\partial x'_t})] \tag{4}$$

where x'_t indicates an adversarial sample that has been perturbed at iteration t, $\prod_{x,\epsilon}$ function centers x'_t within ϵ with x (i.e. unperturbed image) as the center, α is the step size to move per pixel, Γ is the sign function of the gradient, L^* is the object detector's loss function, O^* is the ground truth associated to x. Both L^* and O^* change depending on the type of attack being carried out. Finally, W denotes the model weights.

Based on their targeted specificity, TOG attacks can be classified into the following categories:

1) TOG-Vanishing: the target detection is configured to ensure that the victim model detects no objects in the adversarial example, s.t. $L^* = L$ and $O^* = \emptyset$.
2) TOG-Fabrication: the target detection is designed to create a significant number of false objects, s.t. $L^* = -L$ and ground-truth O^* is the prediction of the model.
3) TOG-Mislabelling (most-likely): each object is assigned an incorrect label and the attack chooses the class label with the second-highest probability, s.t. $L^* = L$ and the ground-truth O^* is the prediction of the model.
4) TOG-Mislabelling (least-likely): each object is assigned an incorrect label and the attack chooses the class label with the lowest probability, s.t. $L^* = L$ and ground-truth O^* is the prediction of the model.
5) TOG-universal: this attack involves creating a single perturbation that is designed to be effective across various images, consistently leading to a object-vanishing effect.

Iterative optimization is necessary for all TOG attack types to generate effective perturbations during online detection, despite the fact that they are all very effective in deceiving an object detector. Using iterative optimization during training, the TOG universal attack can provide a universal perturbation that can be applied to any input to the detector.

III. RESEARCH SETUP

A. DATASETS

We selected various datasets containing maritime objects to retrain the YOLO models. To introduce as much variety as possible, we used three datasets that differed in the shapes, visible proportions, atmospheric conditions, illumination, occlusion, and scale variations of the vessels. We chose the ABOships-PLUS [18], SeaShips [46], and SMD-Plus [23] datasets as our primary sources of maritime imagery. Examples of images from each of the three datasets are shown in Figure 2.

1) ABOSHIPS DATASET

Among the three datasets, the ABOships-PLUS dataset [18] contains smaller maritime objects overall in terms of the occupied pixel area of the bounding box. This dataset was acquired from a set of videos collected using an RGB camera placed on a sightseeing watercraft. The video camera had a 65 ° field of view and the data was stored in HD resolution at 15 frames per second (FPS). The dataset comprises 9880 images extracted from the 135 videos captured from the sightseeing watercraft. The annotations include 4 classes of maritime objects including various types of vessels and other miscellaneous floaters.

Diversity in the ABOships dataset is accounted for by including images encompassing a variety of environmental conditions, such as variations in background, atmospheric conditions, illumination, occlusion, and scale. The images acquired from the videos were then separated into different work packages in chronological order. After the initial manual labeling of the images between different annotators, the CSRT tracker [13] was used to increase the consistency of the labeling process. Due to occlusion and object drift, the automated tracker process resulted in labeling errors. To compensate for this, a relabeling algorithm was used, in which the resulting traces left by the tracker were assigned to the same annotators for refinement. After the relabeling process, inconsistencies and missed objects in the dataset were corrected. As the results of this dataset on state-of-the-art object detectors are promising, ABOships was used as one of the primary sources of maritime objects.

2) SEASHIPS DATASET

The Seaships dataset [46] comprises 31 455 images of six common ship types. The images are part of 1-minute video segments that have been recorded in HD resolution. Three types of cameras were used to ensure high clarity, different scales, and rich video data were obtained. As the shape of the ships is heterogeneous and the bounding boxes around specific ship classes include extensive background data, this dataset included 45 different backgrounds to avoid background information being identified as ship features that would compromise object detection models. In addition to background selection, different lighting environments, visible proportions, and high occlusions were accounted for in the dataset. To maintain a higher quality, the annotations for this dataset were performed manually, following the PASCAL VOC2007 format [12]. During the annotation process, images that did not contain ship objects and images that were very similar to one another were excluded to prevent redundancy.

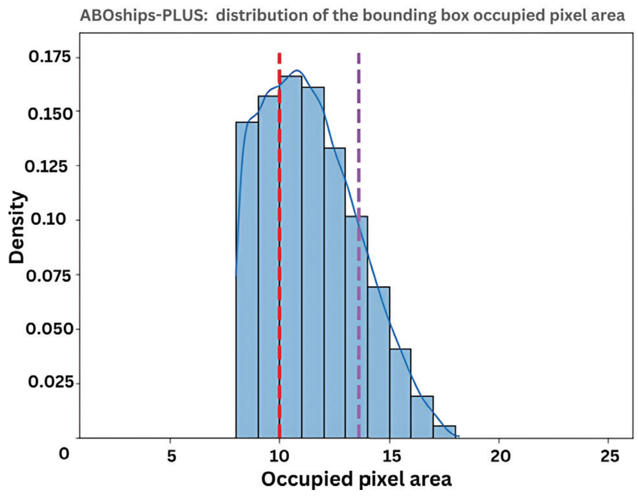


FIGURE 3. Histogram of the occupied pixel area of bounding boxes, at $\log_2$-scale for the AboShips-PLUS dataset (abbreviated as follows: ABOShips-PLUS = ABO).

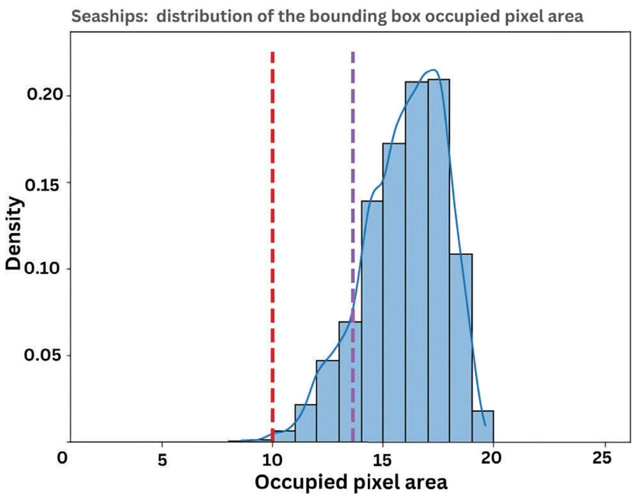


FIGURE 4. Histogram of the occupied pixel area of bounding boxes at $\log_2$-scale for the SeaShips dataset (abbreviated as follows: SeaShips = SS).

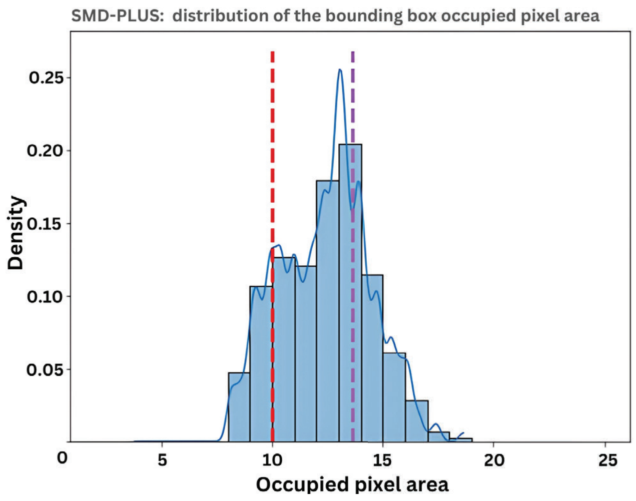


FIGURE 5. Histogram of the occupied pixel area of bounding boxes at $\log_2$-scale for the SMD-Plus dataset (abbreviated as follows: SMD-Plus = SMD).

3) SMD-PLUS DATASET

The third dataset included in our study was the Singapore Maritime Dataset (SMD) [39], specifically the SMD-Plus [23] version. This is a public dataset with precisely annotated videos. The SMD provides high-quality videos with 10 types of labeled objects. The previous version of SMD included some label errors and imprecise bounding boxes due to the fact that the annotations were created by non-expert volunteers. Therefore, the improved SMD-Plus incorporated some changes in the design of the dataset itself to minimize errors.

The improved version of SMD, SMD-Plus, includes class changes, mismatches, and deletions. As some of the classes were non-ship and the purpose of the dataset is to focus specifically on the vessels in the maritime environment, the non-ship classes were removed. Moreover, visually similar classes were merged to enhance clarity. The bounding boxes were corrected by tightening their edges. Missing

TABLE 1. Dataset and their bounding box (BB) sizes. Datasets are abbreviated as follows: ÅboShips = ABO [17], SeaShips = SS [46], SMD-Plus = SMD [23].

Source Dataset	Mean BB Size (log_2)	No. Small BB	No. Medium BB	No. Large BB	No. Total BB
ABO	11.41	10047	17467	5713	33227
SS	15.98	15	1034	8172	9221
SMD	12.39	26921	97297	48369	172587

and misclassified objects were corrected using precise annotations. As the previous version of SMD suffered from class imbalance problems, the SMD-Plus version incorporated data augmentation methods based on YOLO-V5 [22] such as mosaic and geometric transformations, employing the Online Copy & Paste and Mix-up techniques. The SMD-Plus dataset, after annotation correction and data augmentation, addressed many of the errors present in the original SMD. As the performance of SMD-Plus is also better than that of state-of-the-art object detection models, it was chosen as one of the baseline sources of maritime objects.

B. DATA PREPROCESSING

The images acquired from the aforementioned datasets (Aboships-PLUS, Seaships, and SMD-Plus) were the primary sources of maritime objects in our study. Although the images used in this study were curated from prior datasets, a degree of change in data loading, annotation structure, overall class modification, and combination of datasets were necessary to complete this study. This section elaborates on the changes applied to the source datasets to incorporate our study.

1) DATA LOADERS WITH ANNOTATION CONVERSION

This study, which primarily focused on the YOLO object detector architecture, required a specific dataset configuration to fully utilize the capabilities of the object detector. To enable the full use of the object detector architecture, the annotation of the dataset must comply with the Microsoft COCO format. Microsoft COCO is a cost-effective yet high-quality annotation strategy used for YOLO object detector models [40] for training and testing purposes.

Each dataset used in this study has a unique format for representing maritime data suitable for its own design. However, as this study requires the YOLO model for its primary testing suite, we used custom data loaders for each source of maritime objects because the object detector required the Microsoft COCO format [30]. The SeaShips dataset was created using the COCO format. Therefore, only loading the dataset was required. However, the ABOships and SMD-Plus datasets were designed using different annotation formats. Hence, both datasets required unique data loaders to convert their respective formats to the COCO format. The purpose of building the data loaders was not only to load the data but also to enable fine-tuned monitoring of the dataset statistics and changing of the dataset design as a whole.

2) DATASET DESIGN AND STATISTICS

By adopting changes in each dataset annotation, this study aimed to create a more generalized dataset and investigate adversarial attacks in the maritime domain. One of the most significant changes made to the datasets was class modification. This study employed a super-class inclusion strategy to avoid the impact of class imbalance present in the datasets. As mentioned above, the source datasets used in this study originated from different research experiments, and the class distribution was unique for each dataset. This lead to an improper class distribution among the datasets and also introduced new problems, such as inconsistent class names across datasets. Hence, the super-class called *maritime object* was introduced to encompass all the different maritime object types across the datasets. This change ensured that this study would not face class imbalance and enabled the use of different dataset combinations, addressing inconsistencies across the datasets.

The change in the design of the datasets enabled this study to further investigate the distribution of maritime objects in the datasets with respect to their size. To quantify such descriptive measures, the area of the bounding boxes surrounding maritime objects was considered as the unit of measurement. The area in the bounding boxes was calculated at the pixel level, where the width and height of the respective bounding boxes were calculated by the distance between pixels of the coordinates of the x-and y-axes, respectively. The $\log_2$-scale of the bounding box areas was compared with predefined thresholds to distinguish between small, medium, and large maritime objects. A detailed account of the total number of bounding boxes of different sizes from each source dataset is presented in Table 1.

The three datasets were carefully chosen to ensure that the study covered maritime objects of all sizes. With a closer inspection of the distribution of the bounding box areas, it is evident that all three datasets (ABOships-PLUS, SeaShips, and SMD-Plus) are skewed towards one size category (small, medium, or large). Moreover, to visualize such skewness, we illustrate the distribution of the annotated objects of each dataset based on the occupied pixel area at $\log_2$-scale in Figures 3, 4, and 5. This study considers that objects are classified into groups [17] of small ($log_2(area) < 10$), medium ($log_2(area) < 13.6$), and large ($log_2(area) > 13.6$) according to Microsoft COCO variants [30].

The vertical dashed lines in Figures 3, 4, and 5 represent the following values: the red line represents the threshold for small objects, and the purple line represents the threshold for big objects [17]. In each histogram, the entries to the left of the red line represent the small-object group, the entries between the red and purple lines represent the medium-sized group, and those to the right of the purple line represent the large-object group.

Figures 3, 4, and 5 illustrate that the ABOships-PLUS dataset comprises mostly small to medium-sized objects with the majority representing the smaller end of the spectrum. As stated previously, the ABOships-PLUS dataset was specifically chosen to emulate smaller objects. In contrast, the Seaships dataset represents mostly large maritime objects with a magnitude greater than 13.6 pixel area at log_2-scale. The SMD-Plus dataset has a more uniform distribution of maritime objects because it has representative bounding boxes of all sizes. All 3 of these datasets were considered to emulate the real-world maritime environment because they offer variations in terms of distance, magnitude, and scale.

3) DATASET COMBINATIONS

To maintain the fidelity and robustness of this research, the datasets employed were subjected to various combinations to safeguard both the generalizability and reliability of the study. This strategic approach was undertaken to ensure that the research findings maintain their broader applicability and consistent, replicable nature essential to the scientific process. The combinations of the datasets are shown in Table 2 with unique IDs assigned to each combination, as they will be mentioned multiple times in the subsequent sections of this study.

TABLE 2. Dataset and their Combinations. Datasets are abbreviated as follow: ABOShips-PLUS = ABO, SeaShips = SS, SMD-Plus = SMD.

Dataset ID	Source Dataset(s)
D1	ABO
D2	SS
D3	SMD
D4	ABO, SS
D5	ABO, SMD
D6	SS, SMD
D7	ABO, SS, SMD

Due to the widespread use of YOLO architectures for object detection, several open-source libraries are available. We chose the Ultralytics implementation of YOLO [1] for its extensive support and well-structured code. YOLOv3, YOLOv5, and YOLOv8 were selected partly because of the availability of their COCO pre-trained models within the Ultralytics library. The properties of these models are listed in Table 3.

C. ATTACK SCENARIO SETUP

In this study, we considered the following attack scenarios for an autonomous ferry. The attacker is assumed to operate in a completely black-box environment, lacking access to the target model's weights and critical information, such as the data it has been trained on and their specific model architecture. The attack relies solely on the input data and the model output. However, we allowed some leeway in that we assumed that the defender would use a popular architecture; hence, we selected various architectures of the YOLO detector. In addition, we assumed that some publicly available maritime data on the internet are utilized to retrain the target model.

D. EVALUATION METHODS

Various quantitative metrics can be used to assess the performance of object detectors on image datasets. The mean average precision (mAP) is a widely used metric in object detection, as it provides a comprehensive evaluation of detector performance, considering both the accuracy and the precision of detection [34]. Each bounding box prediction BB_i is compared with the ground-truth bounding box BB_{gt} using IoU (Intersection over Union). IoU quantifies the overlap between two bounding boxes by calculating the ratio of the area of the intersection of the predicted bounding box BB_i and the ground-truth bounding box BB_{gt} to their union and conforms to the following equation [52]:

$$IoU = \frac{\cap(BB_i, BB_{gt})}{\cup(BB_i, BB_{gt})} \tag{5}$$

If IoU falls below the desired threshold, the prediction is marked as a false positive (FP). Finally, the predicted bounding box class is compared to the ground truth and considered a true positive (TP) if the confidence of the predicted bounding box class is higher than the desired confidence threshold. With this, it is possible to calculate both the precision and the recall of the predictions at different confidence thresholds.

To calculate average precision (AP), a precision-recall curve is constructed by sorting the predicted bounding boxes based on their confidence scores. The precision and recall values were calculated at different thresholds, starting from the highest confidence scores. AP was computed by calculating the area under the precision-recall curve. This area represents the average precision of the object detector across different recall levels. If multiple object classes are present in the dataset, the AP is calculated separately for each class, and the mAP is obtained by averaging the individual AP values across all classes. This provides an overall performance measure for the object detector.

When calculating the performance of the attack, the target is evaluated against the validation set of the data on which it was trained. In this case, the validation set images were perturbed using a surrogate model. The performance of the target model on the perturbed validation set was compared with its performance on the original validation set. The performance used for comparison was the reduction in mAP50, which was calculated as follows:

$$mAP50_{reduction} = \frac{mAP50_{original} - mAP50_{perturbed}}{mAP50_{original}} \tag{6}$$

This equation provides the percentage of the original performance that is reduced owing to the attack. $mAP_{reduction}$ can also be used to evaluate the effectiveness of the adversarial generator, referred to as the 'attack model, ' in generating adversarial attacks against the victim model, referred to as the 'defender model.'. The higher the value, the more effective the attack model is. It is also possible for some models to improve their performance despite the data being adversarially perturbed by some models, as we will see in the results section.

TABLE 3. YOLO architectures used in the experiments and their size ranges.

YOLO Version	Variant	Number of Layers	Parameters	Weight File Size (MB)
YOLOv3	Original	310	103754144	208
YOLOv3	Tiny	82	12132642	24.3
YOLOv5	Extra Large	493	97276448	195
YOLOv5	Small	262	9122579	18.5
YOLOv8	Extra Large	365	68229648	137
YOLOv8	Small	225	11135987	22.5

E. RESEARCH QUESTIONS FOR THE ATTACK SCENARIO

We investigated the transferability of the TOG fabrication attack in different YOLO architectures, as depicted in Table 3. These architectures were trained on the different datasets described in Table 2. First, we performed transfer learning by employing six different YOLO models on seven different datasets, accounting for 42 trained models. Subsequently, each of these models was used as an adversarial generator to generate adversarial samples for an attack against every other model, including itself. The adversarial samples are generated using the same data employed to validate the victim model. As a result, there are 1764 attacks in total, with all but 42 being transfer attacks, while the remaining 42 are effectively white-box attacks.

For example, to observe the performance of M1 as an adversarial generator against M2, we took dataset D2, on which M2 was trained and tested, and generated adversarial samples using M1 and entitled it $D2_{M1}$. M2 was then again evaluated with $D2_{M1}$ and subsequently $mAP50_{reduction}$ was calculated using the resulting $mAP50$ from Table 4. This experiment was repeated for every possible combination of model pairs. The effectiveness of the attack is defined in terms of $mAP50_{reduction}$.

To investigate the transferability of the attacks, we posed the following research questions:

1) RQ1: Are there significant differences in attack effectiveness among the YOLO architectures listed in Table 3 in general, and if so, which YOLO architecture(s) should be preferred as attack generator?
2) RQ2: Are there differences in the effectiveness of the attack among the YOLO architectures listed in Table 3 when there is no overlap in the datasets trained on and if so, which YOLO architecture(s) should be preferred as an attack generator?
3) RQ3: Are there differences in attack effectiveness among the YOLO architectures listed in Table 3 when there are overlaps in the datasets trained on? If so, which YOLO architecture(s) should be preferred as attack generator(s)?
4) RQ4: Is there a difference between surrogate models that are trained on more data (e.g. models trained only on D7) than the models that are trained on less data (e.g. models trained on D1) in the ability to generate adversarial samples that transfer well to victim models?
5) RQ5: In scenarios where attack instances do not involve data intersections between surrogate and victim models, we investigate whether a difference exists between surrogate models trained on varying amounts of data. Specifically, we compare models trained on more data (e.g., models trained exclusively on D4) to those trained on less data (e.g., models trained on D1) in terms of their ability to generate adversarial samples that transfer effectively to the victim models.
6) RQ6: This research question explores whether there exists a difference in adversarial transferability between surrogate models trained on different base datasets, specifically D1, D2, and D3.
7) RQ7: In the scenario where attack instances do not include any data intersection between the surrogate and victim models, we investigated whether there is a difference in the ability of adversarial transferability among surrogate models trained on different base datasets (D1, D2, and D3).

Addressing these questions will provide insights into the performance and transferability of the surrogate and meta-surrogate models in attacking the target model, as well as the influence of dataset intersections on their effectiveness.

IV. RESULTS

A. MODELS PERFORMANCES

Each version of the YOLO model in Table 3 was trained with each dataset in Table 2. Each training instance was run for 30 epochs. All training instances used the same configuration as that defined in the Ultralytics Library. The hyperparameters were as follows: a constant learning rate of 0.01 with an SGD optimizer and a momentum of 0.937, with a weight decay parameter of 0.0005. The images were resized to 640×640 square pixels, with the number of images per batch adjusted to 2. All experiments were performed on NVIDIA GeForce RTX 2080 Ti, with 11 GB GDDR6 and 4352 NVIDIA CUDA cores, CUDA Version: 11.3, and PyTorch version 1.1.0.

Table 4 reveals that models trained and validated on datasets incorporating the base dataset D3 exhibited the smallest $mAP50_{reduction}$ when subjected to adversarial attacks. Conversely, any combination involving D1 resulted in a greater $mAP50_{reduction}$. This discrepancy arises because D1 presents significant challenges for model training, primarily because of its relatively small bounding boxes and ships blending in with the background. A qualitative illustration of the perturbation effects on three of the models is provided in Figure 6.

TABLE 4. Model performances after transfer learning of YOLOv3, YOLOv3Tiny, YOLOv5, YOLOv5S, YOLOv8 and YOLOv8S object detectors on datasets combinations from Table 2. *mAP50* refers to the *mAP* value at *IoU* of 0.5.

Model ID	Source Architecture	Source Dataset	mAP50	mAP50 reduction (white box attack) (%)
M1	YOLOv3	D1	0.8154	0.720
M2	YOLOv3	D2	0.9926	0.514
M3	YOLOv3	D3	0.9900	0.157
M4	YOLOv3	D4	0.8738	0.655
M5	YOLOv3	D5	0.9722	0.242
M6	YOLOv3	D6	0.9899	0.181
M7	YOLOv3	D7	0.9753	0.280
M8	YOLOv3Tiny	D1	0.6828	0.715
M9	YOLOv3Tiny	D2	0.9789	0.390
M10	YOLOv3Tiny	D3	0.9202	0.166
M11	YOLOv3Tiny	D4	0.7288	0.641
M12	YOLOv3Tiny	D5	0.8771	0.223
M13	YOLOv3Tiny	D6	0.9176	0.213
M14	YOLOv3Tiny	D7	0.8797	0.288
M15	YOLOv5	D1	0.8190	0.680
M16	YOLOv5	D2	0.9930	0.331
M17	YOLOv5	D3	0.9895	0.110
M18	YOLOv5	D4	0.8715	0.555
M19	YOLOv5	D5	0.972	0.180
M20	YOLOv5	D6	0.9897	0.141
M21	YOLOv5	D7	0.9732	0.223
M22	YOLOv5S	D1	0.7919	0.780
M23	YOLOv5S	D2	0.9890	0.561
M24	YOLOv5S	D3	0.9887	0.150
M25	YOLOv5S	D4	0.8507	0.668
M26	YOLOv5S	D5	0.9693	0.235
M27	YOLOv5S	D6	0.9867	0.190
M28	YOLOv5S	D7	0.9671	0.275
M29	YOLOV8	D1	0.8158	0.698
M30	YOLOV8	D2	0.9940	0.379
M31	YOLOV8	D3	0.9882	0.127
M32	YOLOV8	D4	0.8753	0.570
M33	YOLOV8	D5	0.9731	0.206
M34	YOLOV8	D6	0.9880	0.189
M35	YOLOV8	D7	0.9739	0.256
M36	YOLOv8S	D1	0.8016	0.720
M37	YOLOv8S	D2	0.9906	0.458
M38	YOLOv8S	D3	0.9900	0.165
M39	YOLOv8S	D4	0.8589	0.614
M40	YOLOv8S	D5	0.9686	0.242
M41	YOLOv8S	D6	0.9879	0.222
M42	YOLOv8S	D7	0.9684	0.297

B. RESULTS AND DISCUSSION FOR THE ATTACK SCENARIO

This subsection is divided into eight parts, comprising an overview of the experiments and an account of each of the seven research questions. The first part shows the overall results of the experiments through a heatmap, which we discuss below, see Figure 7. Subsequently, the other sections present the results of each statistical analysis employed to answer each research question.

Each research question was investigated through statistical analysis and followed the same format. First, we introduce a table that details the statistical description of the $mAP50_{reduction}$ of the attack instances related to that question. Finally, if the alternative hypothesis of the relevant statistical test is accepted, then a table that describes the Wilcoxon Rank-Sum test will follow.

The relevant statistical tests mentioned above depend on the research questions. RQ1, RQ2, RQ3, RQ4, RQ6, and RQ7 use the Kruskal-Wallis statistical test because these research questions divide the attack instances result into multiple groupings. Therefore, the Kruskal-Wallis test was first applied and reported. If the alternative hypothesis is accepted, the Wilcoxon Rank-Sum and Bonferroni multi-test correction is then utilized and reported as a table. However, RQ5 represents a comparison between only two groups, and hence only the Wilcoxon Rank-Sum test is applied and reported. The significance threshold for all statistical tests was set at 0.05.

The Kruskal-Wallis and Wilcoxon Rank-Sum tests are used for the following reasons: each group of attack instances is independent, the instances within each group are representative of the overall scenario, and the attack results are ordinal in nature. The attack instances of the groups in each research question are independent of one another, i.e., the surrogate or adversarial generator models of each group do not appear in another. They may share common victim models but not surrogate models. Hence, we assumed independence between the groups. Although random sampling is not applicable to our case, we believe that the scenario we have set up here is diverse and expressive enough to represent

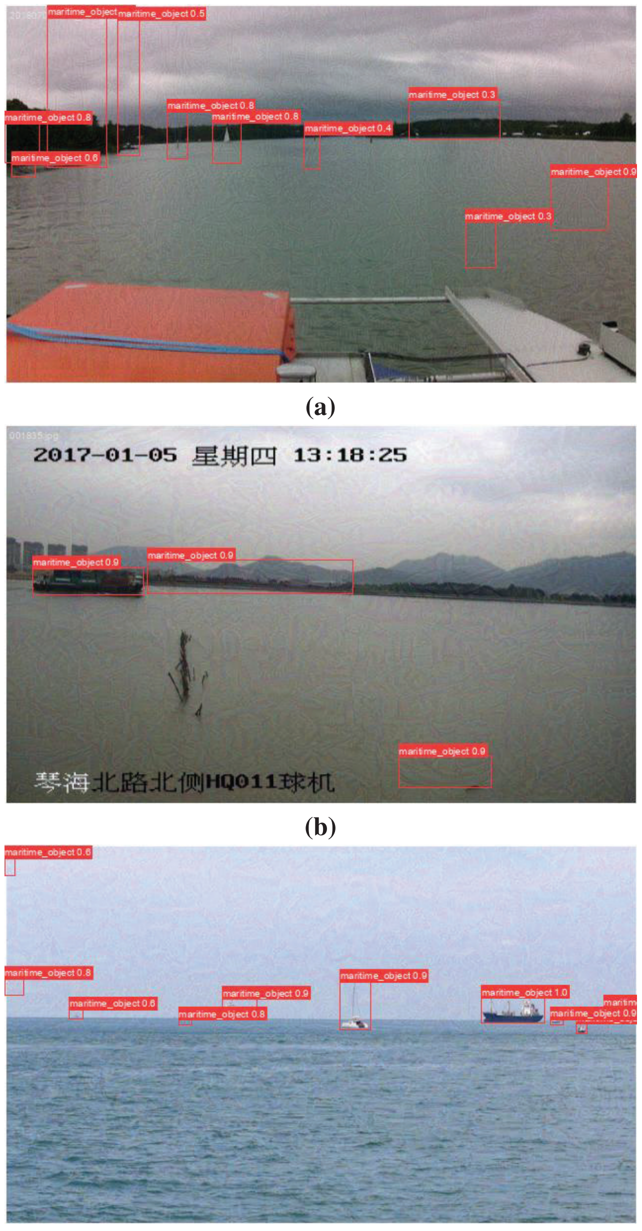


FIGURE 6. Qualitative detection results for white-box attack predictions: Models M15 (a), M16 (b), and M17 (c) on their respective validation datasets following perturbations by a TOG-fabrication attack on the same models.

potential attacks in real-world situations. Hence, we can assume random sampling criteria for this study. Because the data are ordinal in nature, all the criteria for using both statistical tests were met.

Result of all attack instances. Each model was used as an adversarial generator to evaluate the performance of all 42 models, including itself. This resulted in 1764 attack instances.

In the general equation of the TOG attack (Equation 4), ϵ, the upper limit of the perturbation amplitude, signifies the maximum change allowed at any point using the L_∞ distance metric. Intuitively, epsilon indicates the maximum amount by which a single pixel can be modified without restricting the total number of pixels that can be altered. For all our experiments, we used $\epsilon = 8$ as the standard value for all TOG attacks [9]. This means in practice that, given that the pixel ranges from 0 to 255, $\epsilon = 8$ indicates that no pixel is altered by more than $8/255 = 0.03137$ [6]. The step size was set to $\alpha = 2$ and the number of iterations was set to 10. All performed attacks were TOG-fabrication attacks, which, in our case, deceive the victim model to erroneously detect false ships (which are not, in fact, present in the image), see the qualitative results in Figure 6. The attack performance is measured by the reduction in $mAP50$, i.e. $mAP50_{reduction}$, expressed as the percentage decrease in $mAP50$ when the attack architecture is applied, see Equation 6.

In Figure 7, the diagonal represents the instances of white-box attacks. The y-axis indicates the surrogate model in the format of "Model Name - Dataset Combination" while the x-axis represents the victim model in the same format. The naming order corresponds to the model ID presented in Table 4. The brighter the cell on the heatmap, the more effective the adversarial generation by that surrogate model is.

Figure 7 illustrates the white-box attack instances along the diagonals of the heatmap, which appear to be the brightest in each row, a result which is expected by their very definition. In addition, visual inspection suggests a pattern in which different surrogate models exhibit varying effectiveness in generating adversarial samples for other model types. In the next section, we validate these observations using statistical testing.

1) RQ1: EFFECTIVENESS OF THE YOLO ARCHITECTURES IN $MAP50_{REDUCTION}$

To answer this research question, we investigated whether there were differences in the effectiveness of the attack among the YOLO architectures listed in Table 3, when there was no overlap in the datasets on which they were trained. Additionally, we sought to identify which YOLO architecture(s) would be most suitable as attack generator(s).

We utilized the entire set of 1764 attack instances for this purpose. We divided these instances into seven groups, each corresponding to one of the YOLO architectures listed in Table 3. A detailed statistical description of $mAP50_{reduction}$ percentage for each group is shown in Table 5.

The Kruskal-Wallis test resulted in a test statistic of 48.541 and a p-value of approximately 2.75×10^{-9}. Since this p-value was below the predetermined significance level, we rejected the null hypothesis, which assumed no differences among the groups. This suggests that there are statistically significant differences in $mAP50_{reduction}$ effectiveness between the groups when considering the entire dataset. Further analysis, as detailed in Table 6, involved pairwise Wilcoxon rank-sum tests, resulting in a total of 15 comparisons. In the subsequent table, it becomes evident that the null hypothesis is rejected in several model pairs, including YOLOv3

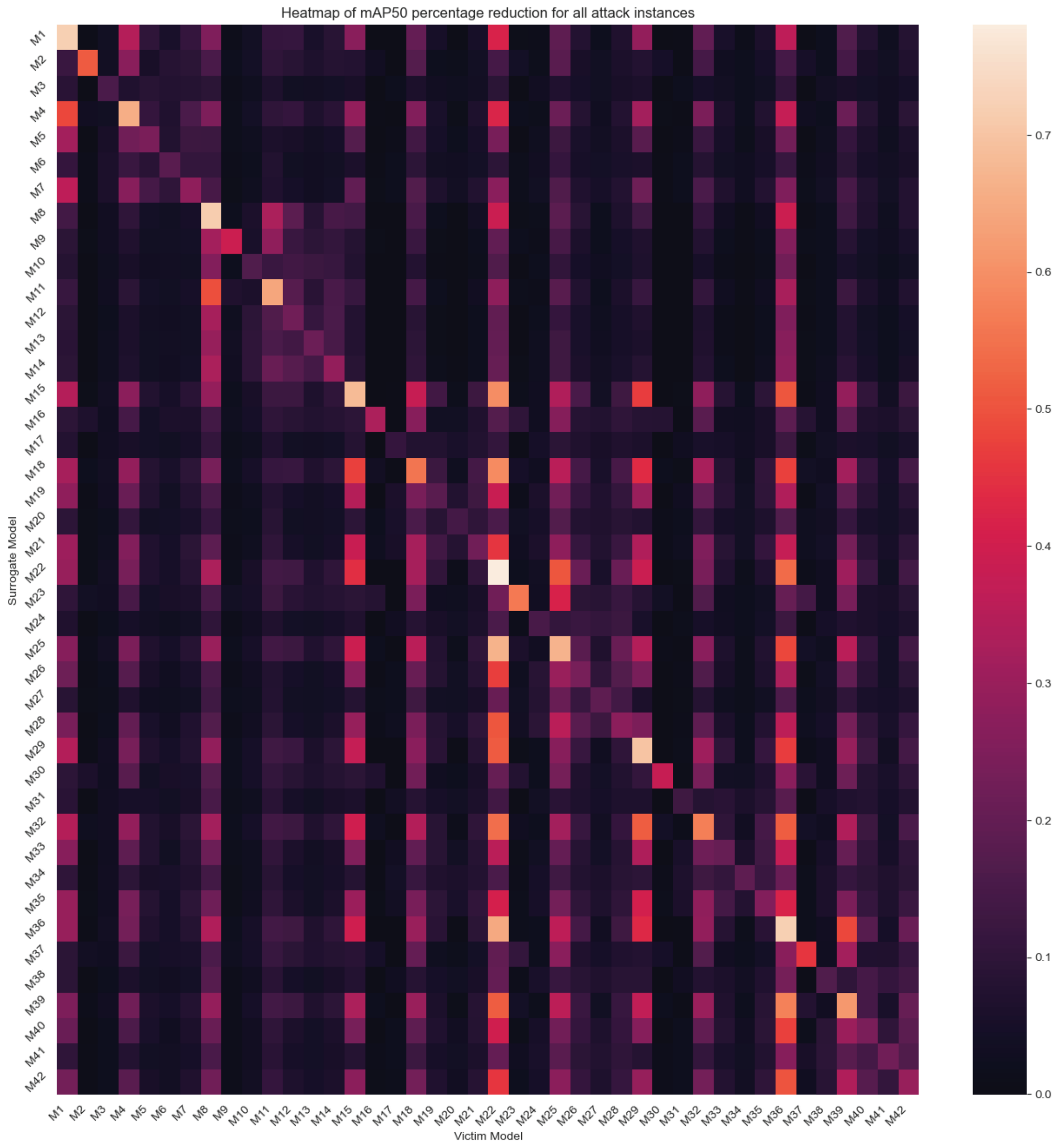


FIGURE 7. ***mAP*50$_{reduction}$ of each attack instance. The diagonal represents the white-box attack instances. The y-axis indicates the surrogate model, and the x-axis represents the victim model. The naming order fully corresponds to the model ID presented in Table 4.**

TABLE 5. **Statistical description of *mAP*50$_{reduction}$ based on various YOLO models as adversarial generators, with all attack instances included.**

	YOLOv3	YOLOv3tiny	YOLOv5	YOLOv5s	YOLOv8	YOLOv8s
count	294.0000	294.0000	294.0000	294.0000	294.0000	294.0000
mean	0.0864	0.0749	0.1087	0.1076	0.1094	0.1116
std	0.1009	0.0968	0.1175	0.1238	0.1153	0.1221
min	-0.0005	-0.0005	0.0010	0.0010	0.0016	0.0010
25%	0.0263	0.0175	0.0363	0.0306	0.0382	0.0323
50%	0.0528	0.0413	0.0690	0.0630	0.0706	0.0695
75%	0.1023	0.0953	0.1223	0.1330	0.1239	0.1399
max	0.7204	0.7159	0.6801	0.7803	0.6985	0.7206

and YOLOv5, YOLOv3 and YOLOv8, and YOLOv3tiny compared to all other models. Complementing these findings, the mean $mAP50_{reduction}$ for YOLOv3tiny, as shown in Table 5, was consistently lower than that of all other architectures.

In conclusion, based on our statistical analysis, YOLOv3tiny is generally less effective as a surrogate architecture for adversarial attacks when compared to other YOLO architectures considered in this study.

TABLE 6. Result of pair-wise Wilcoxon Rank-Sum test of different YOLO architecture as adversarial generator after multi-test Bonferronni corrections. In this case, all attack instances are included.

Architecture 1	Architecture 2	P-Value	Is significant
YOLOv3	YOLOv3tiny	0.1026	False
YOLOv3	YOLOv5	0.0402	True
YOLOv3	YOLOv5s	0.6860	False
YOLOv3	YOLOv8	0.0221	True
YOLOv3	YOLOv8s	0.0893	False
YOLOv3tiny	YOLOv5	9.589e-07	True
YOLOv3tiny	YOLOv5s	9.213e-05	True
YOLOv3tiny	YOLOv8	8.756e-07	True
YOLOv3tiny	YOLOv8s	5.476e-06	True
YOLOv5	YOLOv5s	1.0000	False
YOLOv5	YOLOv8	1.0000	False
YOLOv5	YOLOv8s	1.0000	False
YOLOv5s	YOLOv8	1.0000	False
YOLOv5s	YOLOv8s	1.0000	False
YOLOv8	YOLOv8s	1.0000	False

2) RQ2: EFFECTIVENESS OF YOLO ARCHITECTURES WITHOUT INTERSECTING TRAINING DATA

In this research question, we investigate whether there are differences in attack effectiveness among the YOLO architectures listed in Table 3 when there is no intersection in the datasets trained on. If so, we aim to identify which YOLO architecture(s) would be best suited as attack generator(s).

To address this question, we filter all attack instances with surrogate and victim models that share any overlapping training datasets, resulting in a collection of 1332 attack instances. We maintained the same grouping method utilized in RQ1, and Table 7 shows the statistical description of this dataset.

The Kruskal-Wallis test statistic result was 2.596, with a corresponding p-value of 0.761. Notably, this p-value exceeds the predetermined significance threshold. Consequently, we accept the null hypothesis, which posits that no significant differences in $mAP50_{reduction}$ effectiveness were observed between each group of architectures when considering only disjoint attack instances.

In essence, our statistical analysis suggests that, in cases where there is no overlap in training datasets, there is no substantial variation in attack effectiveness among the YOLO architectures under consideration.

3) RQ3: EFFECTIVENESS OF YOLO ARCHITECTURES ONLY WITH OVERLAPPING TRAINING DATA

In this research question, we investigate whether there are differences in attack effectiveness among the YOLO architectures listed in Table 3 when there are overlaps in datasets trained on, and if so, which YOLO architecture(s) would be preferred as attack generator(s).

To address this question, we excluded all attack instances with surrogate and victim models that shared overlapping training datasets, resulting in a collection of 432 attack instances. Similar to RQ1 and RQ2, we employed the same grouping method, as presented in Table 8.

The Kruskal-Wallis statistical test yields a test statistic of 58.753 and a p-value of 2.199×10^{-11}. This p-value falls below the predetermined significance threshold, leading us to reject the null hypothesis, which posits that there are no observed differences in $mAP50_{reduction}$ among each group of architectures when only intersecting attack instances are considered. Instead, we accept the alternative hypothesis, indicating significant differences in $mAP50_{reduction}$ among these groups.

Following rejection of the null hypothesis, we proceeded with pairwise Wilcoxon Rank-Sum tests and applied Bonferronni corrections to identify pairs of architectures with observed differences in $mAP50_{reduction}$.

Table 9 reveals that YOLOv3 exhibits a statistically different attack performance compared to YOLOv3Tiny, YOLOv5, and YOLOv8, as evidenced by p-values below the significance threshold of 0.05. Consequently, we reject the null hypothesis for these pairs, concluding that YOLOv3 is a statistically better model as an adversarial generator than YOLOv3Tiny. In contrast, YOLOv3 performed worse than YOLOv5 and YOLOv8, although the p-values for these comparisons were not statistically robust, suggesting a weaker level of evidence.

Furthermore, YOLOv3Tiny exhibited a statistically different performance compared to all other models across the board. It consistently ranks as the model with the lowest performance as an adversarial generator for these maritime datasets. Finally, we conclude that YOLOv5 and YOLOv8 are generally the preferred models to be used as surrogate models.

4) RQ4: INFLUENCE OF THE AMOUNT OF TRAINING DATA ON ADVERSARIAL TRANSFERABILITY

In this research question, we investigated whether there is a difference between surrogate models trained on varying amounts of data with respect to their ability to generate adversarial samples that transfer effectively to victim models. In this analysis, we considered all attack instances. The grouping is based on surrogate models trained on different numbers of datasets: 1, 2, and 3. This resulted in three distinct groups: one dataset (models trained on D1, D2, and D3), two datasets (models trained on D4, D5, and D6), and three datasets (models trained on D7). Table 10 provides a statistical description of the $mAP50_{reduction}$ values for these groups, showing the distribution of these values.

The Kruskal-Wallis statistical test yielded a result of approximately 6.9614, with a corresponding p-value of approximately 0.0307. The p-value fell below the predefined significance threshold of 0.05. Therefore, we reject the null hypothesis, which posits that surrogate models trained on different numbers of datasets perform equally well in the transfer attacks. Instead, we accept the alternative hypothesis that there are observed differences in $mAP50_{reduction}$ effectiveness among these groups.

TABLE 7. Statistical description of $mAP50_{reduction}$ based on various YOLO models as adversarial generator. For this table, attack instances with overlapping dataset between attacking model and defending model are removed.

	YOLOv3	YOLOv3tiny	YOLOv5	YOLOv5s	YOLOv8	YOLOv8s
count	72.0000	72.0000	72.0000	72.0000	72.0000	72.0000
mean	0.0426	0.0598	0.0503	0.0526	0.0551	0.0592
std	0.0410	0.0779	0.0459	0.0522	0.0583	0.0655
min	-0.0005	-0.0005	0.0010	0.0010	0.0016	0.0010
25%	0.0069	0.0052	0.0093	0.0120	0.0111	0.0119
50%	0.0326	0.0274	0.0348	0.0342	0.0318	0.0318
75%	0.0630	0.0776	0.0772	0.0733	0.0832	0.0846
max	0.1550	0.3146	0.1866	0.2246	0.2715	0.2702

TABLE 8. Statistical description of $mAP50_{reduction}$ based on various YOLO models as adversarial generators. For this table, only attacks where surrogate model and training model share overlapping in terms of training dataset are considered.

Adversarial Generator Model → Statistical Description ↓	YOLOv3	YOLOv3Tiny	YOLOv5	YOLOv5S	YOLOv8	YOLOv8S
count	222	222	222	222	222	222
mean	0.1006	0.0798	0.1276	0.1254	0.1270	0.1286
std	0.1101	0.1019	0.1271	0.1347	0.1236	0.1311
min	0.0060	-0.0005	0.0117	0.0036	0.0066	0.0056
25%	0.0331	0.0194	0.0444	0.0386	0.0487	0.0437
50%	0.0593	0.0427	0.0795	0.0732	0.0804	0.0790
75%	0.1197	0.1063	0.1450	0.1666	0.1406	0.1741
max	0.7204	0.7159	0.6801	0.7803	0.6985	0.7206

TABLE 9. Result of the pair-wise Wilcoxon Rank-Sum test of different YOLO architecture as adversarial generators after multi-test Bonferronni corrections.

Architecture 1	Architecture 2	P-Value	Is Significant
YOLOv3	YOLOv3tiny	0.00896	True
YOLOv3	YOLOv5	0.0278	True
YOLOv3	YOLOv5s	0.858	False
YOLOv3	YOLOv8	0.0127	True
YOLOv3	YOLOv8s	0.0973	False
YOLOv3tiny	YOLOv5	3.203e-08	True
YOLOv3tiny	YOLOv5s	1.0711e-05	True
YOLOv3tiny	YOLOv8	1.879e-08	True
YOLOv3tiny	YOLOv8s	2.429e-07	True
YOLOv5	YOLOv5s	1	False
YOLOv5	YOLOv8	1	False
YOLOv5	YOLOv8s	1	False
YOLOv5s	YOLOv8	1	False
YOLOv5s	YOLOv8s	1	False
YOLOv8	YOLOv8s	1	False

Subsequently, we performed pairwise Wilcoxon tests and applied Bonferroni corrections, as shown in Table 11. Among the various group pair comparisons, only the comparison between surrogate models trained on 1 dataset alone and those trained on 3 datasets rejected the null hypothesis, with a p-value of 0.0305. This implies that on average training on a larger number of datasets results in more effective surrogate models for transfer attacks. However, it is important to note that all models in these two groups have overlapping training datasets, as the combination of 3 datasets (or D7) encompasses all datasets listed in Table 1.

The conclusion drawn from this analysis suggests that increasing the diversity of training data to increase the chance of intersection in training datasets can lead to more effective surrogate models for adversarial transfer attacks.

TABLE 10. Statistical description of $mAP50_{reduction}$ based on various surrogate models as adversarial generators. This table shows surrogate models trained on 1 dataset (D1, D2, D3), 2 datasets (D4, D5, D6), and 3 datasets (D7) together.

	1 Dataset	2 Datasets	3 Datasets
Count	756.0000	756.0000	252.0000
Mean	0.0941	0.1014	0.1120
Std	0.1142	0.1144	0.1106
Min	-0.0005	-0.0005	0.0006
25%	0.0289	0.0281	0.0305
50%	0.0570	0.0617	0.0701
75%	0.1067	0.1206	0.1685
Max	0.7803	0.6685	0.5088

5) RQ5: INFLUENCE OF THE AMOUNT OF TRAINING DATA ON ADVERSARIAL TRANSFERABILITY WITHOUT DATASET INTERSECTION

In scenarios where attack instances do not involve data intersections between surrogate and victim models, we investigate whether a difference exists between surrogate models trained on varying amounts of data. Specifically, we compare models trained on more data (e.g., models trained exclusively on D4) with those trained on less data (e.g., models trained on D1) in terms of their ability to generate adversarial samples that transfer effectively to victim models.

For this analysis, we exclusively considered attack instances that did not involve any dataset intersections between the surrogate and victim models. This results in the inclusion of only two groups of surrogate models: one trained on a single dataset and the other trained on two datasets. Because there are only two groups of attack instances, we employed the Wilcoxon test without the need for Kruskal-Wallis or Bonferroni correction. This analysis was conducted on 432 attack instances, and the corresponding statistical description is presented in Table 12.

TABLE 11. Result of pairwise Wilcoxon Rank-Sum test after Bonferronni correction of surrogate model groups based on number of datasets the surrogate models are trained on.

Number of datasets the model is trained on	Number of datasets the model is trained on	P-Value	Is Significant
1 Dataset	2 Datasets	0.463	False
1 Dataset	3 Datasets	0.0305	True
2 Datasets	3 Datasets	0.333	False

The results from the Wilcoxon pair-test indicate that the p-value exceeds the predefined significance threshold of 0.05, specifically at 0.221. Therefore, we do not find sufficient evidence to reject the null hypothesis. The null hypothesis suggests that there is no observable difference in $mAP50_{reduction}$ between the two groups, whereas the alternative hypothesis implies that such a difference exists. Hence, we conclude that in scenarios without dataset intersections between the surrogate and victim models, there is no significant difference in the ability of the surrogate models trained on more data compared to those trained on less data when generating adversarial samples that transfer well to the victim models.

TABLE 12. Statistical description of $mAP50_{reduction}$ based on various surrogate models as adversarial generators. The attack instances are grouped based on the number of datasets on which the surrogate models are trained (1 dataset = {D1,D2,D3}, 2 datasets = {D4, D5, D6}). Additionally, all attack instances which have surrogate and victim models sharing intersections in training data are also removed.

	1 Dataset	2 Datasets
Count	324.0000	108.0000
Mean	0.0533	0.0533
Std	0.0555	0.0654
Min	-0.0005	-0.0005
25%	0.0116	0.0059
50%	0.0368	0.0254
75%	0.0770	0.0809
Max	0.3146	0.2926

6) RQ6: IMPACT OF BASE DATASETS ON ADVERSARIAL TRANSFERABILITY WITH ALL DATA

This research question explores whether a difference exists in the ability of adversarial transferability between surrogate models trained on different base datasets, specifically D1, D2, and D3. To investigate this, we organized the attack instances into groups based on the dataset used to train the surrogate models. Attack instances involving surrogate models trained on other datasets (D4, D5, D6, and D7) were excluded. This resulted in a total of 756 attack instances being included in the analysis.

The Kruskal-Wallis test gives a p-value of approximately 7.34×10^{-06}; thus, we reject the null hypothesis. The null hypothesis for this research question suggests that there is no difference in $mAP50_{reduction}$ among the surrogate models trained on different base datasets, whereas the alternative hypothesis suggests that such differences exist. Subsequent Wilcoxon tests (Table 14) further confirmed the rejection of the null hypothesis for the following pairs: D1-D3 and D2-D3. Consequently, we conclude that surrogate models trained on D1 or D3 tend to exhibit better adversarial generation capabilities than those trained on D2. These findings are supported by the descriptive statistics in Table 13.

TABLE 13. Statistical description of $mAP50_{reduction}$ based on various surrogate models as adversarial generator. Attack instances are grouped based on based datasets the surrogate models are trained on (D1 = ABO, D2 = SS, D3 = SMD).

	D1	D2	D3
Count	252	252	252
Mean	0.1345	0.0570	0.0907
Std	0.1621	0.0444	0.0894
Min	-0.0005	0.0000	-0.0005
25%	0.0212	0.0296	0.0336
50%	0.0747	0.0488	0.0665
75%	0.1763	0.0758	0.1040
Max	0.7803	0.2575	0.5612

TABLE 14. Result of pair-wise Wilcoxon rank-sum test after Bonferronni correction. Each pair is a group of attack instances whose surrogate models are trained on the same base dataset (D1, D2, or D3).

Dataset 1	Dataset 2	P-Value	Is Significant
D1	D2	8.24e-05	True
D1	D3	1	False
D2	D3	9.32e-05	True

7) RQ7: IMPACT OF DATA INTERSECTION ON ADVERSARIAL TRANSFERABILITY WITHOUT DATA INTERSECTION

In the scenario where attack instances do not include any data intersection between surrogate and victim models, we investigated whether there exists a difference in the ability of adversarial transferability among surrogate models trained on different base datasets (D1, D2, D3).

Data were organized following the same grouping as described in Section IV-B6, with the exclusion of all attack instances that contained any overlap in the training data between the surrogate and victim models, more details in Table 16.

To clarify, this statistical test only included results from experiments with the following models as adversarial generators (M1, M2, M3, M8, M9, M10, M15, M16, M17, M22, M23, M24, M29, M30, M31, M36, M37, M38, M43, M44, M45). The results are filtered further to not include any result on which the surrogate model and victim model share training data. For example, if M1 is the surrogate model, then the samples of M1 generating adversarial samples against itself (M1) and the following models (M4, M5, M7, M8, M11, M12, etc.) are removed because they are trained on D1, D4, D5, and D7, which share images from the same dataset. Once the results were appropriately filtered, the data were grouped into three sets depending on which dataset (D1, D2, or D3) the surrogate model was trained on.

The Kruskal-Wallis test, used to evaluate whether there are significant differences among the groups, generated a p-value of approximately 3.0085×10^{-13}. This p-value indicates a rejection of the null hypothesis, which posits that there is no difference in $mAP50_{reduction}$ between surrogate models trained on different base datasets when there is no data intersection between the surrogate and victim models. Conversely, we accept the alternative hypothesis, suggesting that surrogate models trained on distinct base datasets result in varying $mAP50_{reduction}$ values under these conditions.

Subsequently, Wilcoxon tests (see Table 16) were conducted to determine specific pairwise differences. The null hypothesis was rejected for the pairs D1-D2 and D1-D3, indicating that the surrogate models trained on D1 exhibited different $mAP50_{reduction}$ outcomes compared to those trained on D2 and D3, respectively. This observation is consistent with the descriptive statistics in Table 15. These results suggest that models trained on D2 or D3 tend to be more effective adversarial generators when there is no intersection in the training datasets between the surrogate and victim models.

TABLE 15. Statistical description of $mAP50_{reduction}$ based on various surrogate models as adversarial generators. Attack instances are grouped based on the datasets the surrogate models are trained on (D1 = ABOships, D2 = SS, D3 = SMD). All attack instances with any intersection of training data between surrogate and victims models are removed.

	D1	D2	D3
count	108.0000	108.0000	108.0000
mean	0.0223	0.0639	0.0737
std	0.0191	0.0553	0.0662
min	-0.0005	0.0000	-0.0005
25%	0.0064	0.0085	0.0283
50%	0.0175	0.0639	0.0528
75%	0.0341	0.0901	0.0928
max	0.0824	0.2575	0.3146

TABLE 16. Result of pair-wise Wilcoxon Rank-Sum test after Bonferronni correction. Each pair is a group of attack instances whose surrogate models are trained on the same base dataset (D1, D2, or D3). All attack instances with any intersection of training data between surrogate and victims models are removed.

Dataset 1	Dataset 2	P-Value	Is Significant
D1	D2	1.02e-07	True
D1	D3	2.84e-13	True
D2	D3	1	False

V. DISCUSSION

A. SUMMARY AND PRACTICAL SIGNIFICANCE

1) FEASIBILITY AND IMPLEMENTATION OF TOG ATTACKS IN REAL DEPLOYMENT ENVIRONMENTS

The implementation of TOG attacks in real-world maritime autonomous systems, such as self-navigating ferries, poses several practical challenges. Although generating TOG attacks in real time allows greater adaptability, it requires significant computational power and low-latency processing capabilities. Moreover, introducing physical perturbations adds another level of complexity, which requires careful consideration to maintain effectiveness under various environmental conditions.

The possibility of an attacker accessing detailed system information and intervention capabilities further complicates the scenario, highlighting the need for robust defence mechanisms. Understanding these factors is crucial for assessing the real-world applicability and security implications of TOG attacks in maritime environments. In the following, we outline a few aspects that address the feasibility and implementation of TOG attacks in practical deployment scenarios.

Online Generation of TOG Attacks: TOG attacks can be generated online, utilizing real-time data from sensors and cameras. This approach allows an attacker to dynamically adjust perturbations based on the current environmental conditions [10]. However, implementing online attacks requires significant computational resources and low-latency processing capabilities to ensure timely and effective perturbations [8].

Physical Perturbation of Input Images: Physical perturbations can be implemented using adversarial patches or stickers placed on images of real-world objects. These patches are crafted to manipulate object detectors by embedding deceptive patterns that lead to misclassifications or missed detections. Physical attacks require precise placement and design of perturbations to ensure that they remain effective under varying conditions, such as lighting, angles, and distances [48].

Attacker Information and Intervention Capabilities: An attacker requires a comprehensive knowledge of the target system, such as its model architecture, training data, and sensor inputs, which is essential to generate effective adversarial examples. However, this requires a high level of access to and control over the target system [10].

2) POTENTIAL THREAT SCENARIOS AND CHALLENGES TO EXISTING PERCEPTION SYSTEMS

Adversarial attacks pose serious risks to maritime autonomous systems by compromising environmental perception, leading to navigation errors, collisions, and misidentification of objects in complex environments. Addressing these threats requires improving the resilience of perception systems through adversarial training, real-time attack detection and mitigation, and sensor fusion-based redundancy [51].

Threat scenarios: TOG attacks pose a critical threat to maritime perception systems, particularly those deployed in autonomous surface vessels and coastal surveillance. These attacks exploit the gradients used by object detectors, such as those in the YOLO [42] or Faster R-CNN [43] models to manipulate the objectness score and suppress or fabricate object detection. In the maritime context, a major threat is the disruption of navigation systems. Adversarial attacks can strategically degrade maritime computer vision and obfuscate navigational hazards, such as buoys and other maritime vessels. This can lead to route deviations, unsafe maneuvers, or even collisions, particularly in congested environments, such as the foreport area [28], [51].

Challenges to Existing Perception Systems: Adversarial training has shown potential to enhance model robustness by introducing perturbed inputs during the training process, thereby increasing the network's resistance to adversarial manipulations. Real-time detection and mitigation are critical issues. Recent studies have demonstrated the effectiveness of combining Gaussian mix variational autoencoders (GMVAE) and reinforcement learning (RL) to identify and counter adversarial patterns in dynamic environments [19]. Furthermore, implementing sensor redundancy and data fusion strategies enhances system resilience by enabling the cross-verification of sensory inputs, which is essential for maintaining operational integrity under adversarial conditions. Roheda et al. introduced a robust multimodal sensor fusion framework that uses generative networks to learn the latent space across different sensor modalities. This approach allows for the detection of damaged sensors and maintains system performance in the presence of noisy or faulty sensor data, contributing to the overall resilience of the system [44]. In the context of maritime object detection, in [35] Mohan et al. developed a cross-sensor vision system that integrates data from multiple sensors using CNNs. Their system demonstrated high accuracy in maritime vessel detection across various sensor types, indicating the potential of sensor fusion for enhancing detection capabilities in complex maritime environments.

B. TOWARD ROBUSTNESS: DEFENSE STRATEGIES AGAINST ADVERSARIAL ATTACKS

This study focused on the transferability and effectiveness of adversarial attacks, particularly TOG [8] attacks on YOLO [41] object detectors, in the maritime domain. In the context of autonomous maritime navigation, for example, it is imperative to also consider defense strategies. Although we leave the detailed presentation of such strategies to other studies, we provide a brief overview of the existing defense mechanisms as follows.

Adversarial Training: is one of the most prevalent and effective strategies against adversarial attacks and consists of augmenting the training dataset with adversarially perturbed examples to allow the model to learn more robust and generalizable features [25], [29]. Projected Gradient Descent (PGD) is a widely used optimization-based method to generate adversarial perturbations in quasi-continuous domains such as images. It plays a central role in adversarial training [32] and is commonly used to construct adaptive attacks for evaluating the robustness of defense mechanisms [53]. Regional Adversarial Training (RAT) improves this approach by sampling various perturbations within an adversarial region and applying distance-sensitive label smoothing to improve robust generalization [47].

Input Image Transformation Techniques: mitigate adversarial perturbations by altering the input image before it is processed by the model. They are typically lightweight and model-agnostic, making them excellent candidates for real-time applications, such as autonomous navigation. Among the most successful techniques, image compression algorithms have been employed to defend against adversarial examples in several studies [11], [20].

C. MARITIME AUTONOMOUS NAVIGATION

Maritime autonomous surface ships (MASS) have revolutionized the maritime industry, resulting in remarkable advances in operational performance, overall system integration, and increased efficiency [50]. A primary challenge for MASS is the integration of international regulations for preventing collisions at sea (COLREGs) into autonomous navigation and collision avoidance systems. The decision-making process for navigation in MASS encompasses the detection and tracking of other ships, assessment of collision risks, planning of safe routes, and maneuvering performance according to COLREG regulations. A robust collision risk inference system is essential for MASS to comply with COLREGs. Such a system evaluates the degree of danger in real time, determines the appropriate response distances, and requires an adaptive inference approach to consider vital variables in COLREGs [37]. Local route planning is another critical component, where algorithms must account for near-collision risks and ensure compliance with specific COLREGs rules, such as those governing lookout, risk assessment, and maneuvering [36].

Collision avoidance is a core issue in autonomous maritime navigation [36], [37], [54]. Modern maritime collision avoidance increasingly relies on sensor fusion, based on camera systems, radar, AIS, etc. Integrating object detection into MASS is crucial for improving situational awareness, ensuring collision avoidance, and enabling autonomous navigation. Object detection facilitates real-time awareness of surrounding vessels and maritime objects, which is crucial for preventing collisions and ensuring that navigation decisions adhere to the COLREG regulations [36], [37]. However, adversarial attacks can significantly undermine object detection models, causing the system to misinterpret or fail to recognize objects, leading to incorrect environmental understanding and potentially dangerous navigation decisions. Deep neural network-based object detection models, including YOLO, are susceptible to a range of adversarial attack techniques, exposing a critical vulnerability in the robustness of MASS [27].

VI. CONCLUSION

In this paper, we benchmark various YOLO architectures (see Table 3), which were trained on different combinations of datasets (see Table 2) and investigated their ability to generate adversarial samples to reduce victim models' $mAP50$. Furthermore, we statistically explored several research questions related to the effectiveness (or $mAP50_{reduction}$) of different YOLO architectures in adversarial attacks on object detection models. Our findings provide insights into the performance of these architectures and their suitability as adversarial generators.

RQ1: We observed significant differences in attack effectiveness between YOLO architectures when considering all attack instances. Among those showing statistical difference, YOLOv3Tiny had the lowest performance with $mAP50_{reduction}$ of 0.0864, while YOLOv5 and YOLOv8 showed promise as better adversarial generators at a mean $mAP50_{reduction}$ of 0.1087 and 0.1094, respectively.

RQ2: When considering only non-overlapping attack instances, we found no significant differences in the attack effectiveness among the YOLO architectures. This suggests that when there is no overlap in the datasets between the surrogate and victim models, the choice of YOLO architecture may not be a critical factor.

RQ3: In cases where only overlaps existed in the datasets, we observed significant differences in the attack effectiveness among the YOLO architectures. Statistically, YOLOv3 outperformed YOLOv3Tiny, with a mean $mAP50_{reduction}$ of 0.1006 versus 0.0798, respectively, but it was slightly outperformed by YOLOv5 and YOLOv8, which had mean $mAP50_{reduction}$ values of 0.1276 and 0.1270 respectively. However, the latter comparisons against YOLOv5 and YOLOv8 had relatively weak statistical evidence in terms of p-values of 0.0278 and 0.0127, respectively, compared to other statistically significant comparisons in the research question, where the p-values were below 0.01.

RQ4: Surrogate models trained on more data (e.g., D7) demonstrated a statistically significant advantage in generating effective adversarial samples at a mean $mAP50_{reduction}$ of 0.1120 compared to models trained on less data (e.g., D1) at a mean $mAP50_{reduction}$ of 0.0941. However, it should be noted that the group pairing that shows statistical significance has a group whose surrogate models are trained on a dataset with some intersection with the victim model.

RQ5: Interestingly, when attack instances excluded data intersections between surrogate and victim models, there was no observable difference in attack effectiveness between models trained on different amounts of data.

RQ6: Significant differences in adversarial transferability were observed between surrogate models trained on different base datasets (D1, D2, and D3). Models trained on D1 or D3 generally performed better, at a mean $mAP50_{reduction}$ of 0.1345 and 0.0907 respectively, than those trained on D3 at a mean $mAP50_{reduction}$ of 0.0570.

RQ7: Similar to RQ6, when the attack instances excluded data intersections, we observed significant differences in adversarial transferability between the models trained on different base datasets. Models trained on D2 or D3 performed better, at a mean $mAP50_{reduction}$ of 0.0639 and 0.0737 respectively, than those trained on D1 at a mean $mAP50_{reduction}$ of 0.0223.

In summary, our research provides insights into the selection of YOLO architectures for adversarial attacks in the maritime environment. The choice of architecture can significantly impact the attack effectiveness, especially when considering dataset overlaps and the amount of training data. However, the effectiveness of the architecture can vary depending on the specific scenario and dataset intersections. In addition, while our work considers various data intersection scenarios, we would like to note that this list is by no means exhaustive, and more public datasets can be included. Researchers and practitioners should consider these findings when designing adversarial attacks in the field of object detection.

Particulary, this study benchmarks the effectiveness of various YOLO architectures in generating adversarial samples to reduce victim models' mAP50. Our key findings include the following. YOLOv5 and YOLOv8 demonstrated the highest attack effectiveness, whereas YOLOv3Tiny exhibited the weakest performance. When there were no data and overlap between the surrogate and victim models, the choice of YOLO architecture was not a significant factor. However, in cases with overlapping datasets, YOLOv5 and YOLOv8 outperformed YOLOv3 and YOLOv3Tiny. Surrogate models trained on larger datasets (e.g., D7) generated stronger attacks; however, this advantage disappeared when dataset intersections were removed. The choice of base dataset also influenced adversarial transferability, with models trained on D1 or D3 performing better. These findings highlight the impact of dataset overlap and training data volume on adversarial attack effectiveness in object detection.

VII. THREATS TO VALIDITY AND RELIABILITY

The validity of this study is prone to weak generalizability of the results, since the experiments were conducted using a family of object detectors (YOLO) and an attack type (TOG), effectively limiting the reliability at scale. Transferability results may not generalize to other object detection models (e.g., Faster R-CNN, SSD, etc.) or other types of adversarial attacks. Furthermore, the simple attack scenario presented in Section III-C constrained the adversarial attacks and object detectors, which we considered relevant for the study based on publicly available datasets and models. The results presented in this study are highly dependent on the available data and detectors' architectures.

REFERENCES

[1] (2024). *Ultralytics YOLO*. Accessed: Jul. 1, 2024. [Online]. Available: https://github.com/ultralytics

[2] P. Adarsh, P. Rathi, and M. Kumar, "YOLO v3-tiny: Object detection and recognition using one stage improved model," in *Proc. 6th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Mar. 2020, pp. 687–694.

[3] N. Akhtar, A. Mian, N. Kardan, and M. Shah, "Advances in adversarial attacks and defenses in computer vision: A survey," *IEEE Access*, vol. 9, pp. 155161–155196, 2021.

[4] A. Amirkhani, M. P. Karimi, and A. Banitalebi-Dehkordi, "A survey on adversarial attacks and defenses for object detection and their applications in autonomous vehicles," *Vis. Comput.*, vol. 39, no. 11, pp. 5293–5307, Nov. 2023.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[7] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Trans. Intell. Technol.*, vol. 6, no. 1, pp. 25–45, Mar. 2021.

[8] K.-H. Chow, L. Liu, M. Emre Gursoy, S. Truex, W. Wei, and Y. Wu, "TOG: Targeted adversarial objectness gradient attacks on real-time object detection systems," 2020, *arXiv:2004.04320*.

[9] K.-H. Chow, L. Liu, M. E. Gursoy, S. Truex, W. Wei, and Y. Wu, "Understanding object detection through an adversarial lens," in *Proc. 25th Eur. Symp. Res. Comput. Secur. Comput. Security*, Guildford, U.K. Cham, Switzerland: Springer, Jan. 2020, pp. 460–481.

[10] K.-H. Chow, L. Liu, M. Loper, J. Bae, M. E. Gursoy, S. Truex, W. Wei, and Y. Wu, "Adversarial objectness gradient attacks in real-time object detection systems," in *Proc. 2nd IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl. (TPS-ISA)*, Oct. 2020, pp. 263–272.

[11] G. K. Dziugaite, Z. Ghahramani, and D. M. Roy, "A study of the effect of JPG compression on adversarial images," 2016, *arXiv:1608.00853*.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[13] Khurshedjon Farkhodov, Suk-Hwan Lee, and Ki-Ryong Kwon, "Object tracking using VOC tracker and R-CNN," in *Proc. Bioimaging*, 2020, pp. 209–212.

[14] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[15] S. V. Grini, "Object detection in maritime environments," Master's thesis, NTNU, 2019.

[16] M. H. Haghbayan, F. Farahnakian, J. Poikonen, M. Laurinen, P. Nevalainen, J. Plosila, and J. Heikkonen, "An efficient multi-sensor fusion approach for object detection in maritime environments," in *Proc. 21st Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2163–2170.

[17] B. Iancu, V. Soloviev, L. Zelioli, and J. Lilius, "ABOships—An inshore and offshore maritime vessel detection dataset with precise annotations," *Remote Sens.*, vol. 13, no. 5, p. 988, 2021.

[18] B. Iancu, J. Winsten, V. Soloviev, and J. Lilius, "A benchmark for maritime object detection with centernet on an improved dataset, ABOships-PLUS," *J. Mar. Sci. Eng.*, vol. 11, no. 9, p. 1638, Aug. 2023.

[19] G. Ingle, K. Patil, and S. Pawale, "A robust defense mechanism against adversarial attacks in maritime autonomous ship using GMVAE+RL," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 4, pp. 1–16, 2025.

[20] X. Jia, X. Wei, X. Cao, and H. Foroosh, "ComDefend: An efficient image compression model to defend adversarial examples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6077–6085.

[21] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of YOLO algorithm developments," *Proc. Comput. Sci.*, vol. 199, pp. 1066–1073, Jan. 2022.

[22] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, K. Michael, J. Fang, Z. Yifu, C. Wong, D. Montes, and Z. Wang, "Ultralytics/YOLOv5: V7. 0-YOLOv5 SOTA realtime instance segmentation," Tech. Rep., 2022.

[23] J.-H. Kim, N. Kim, Y. W. Park, and C. S. Won, "Object detection and classification based on YOLO-V5 with improved maritime dataset," *J. Mar. Sci. Eng.*, vol. 10, no. 3, p. 377, Mar. 2022.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 60, May 2017, pp. 84–90.

[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*.

[26] H. Law and J. Deng, "CornerNet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Jan. 2018, pp. 765–781.

[27] C. Lee and S. Lee, "Evaluating the vulnerability of YOLOv5 to adversarial attacks for enhanced cybersecurity in MASS," *J. Mar. Sci. Eng.*, vol. 11, no. 5, p. 947, Apr. 2023.

[28] C. Lee and S. Lee, "Vulnerability of clean-label poisoning attack for object detection in maritime autonomous surface ships," *J. Mar. Sci. Eng.*, vol. 11, no. 6, p. 1179, Jun. 2023.

[29] S. Lee, H. Lee, and S. Yoon, "Adversarial vertex mixup: Toward better adversarially robust generalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 269–278.

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. 13th Eur. Conf. Comput. Vis.*, Zurich, Switzerland. Cham, Switzerland: Springer, Jan. 2014, pp. 740–755.

[31] H. Liu, F. Sun, J. Gu, and L. Deng, "SF-YOLOv5: A lightweight small object detection algorithm based on improved feature fusion mode," *Sensors*, vol. 22, no. 15, p. 5817, Aug. 2022.

[32] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[33] K. Mahmood, R. Mahmood, E. Rathbun, and M. van Dijk, "Back in black: A comparative evaluation of recent state-of-the-art black-box attacks," *IEEE Access*, vol. 10, pp. 998–1019, 2022.

[34] V. Mazzia, A. Khaliq, F. Salvetti, and M. Chiaberge, "Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application," *IEEE Access*, vol. 8, pp. 9102–9114, 2020.

[35] V. Mohan and S. J. Simske, "Cross-sensor vision system for maritime object detection," *Frontiers Mar. Sci.*, vol. 10, Mar. 2023, Art. no. 1112955.

[36] H. Namgung, "Local route planning for collision avoidance of maritime autonomous surface ships in compliance with COLREGs rules," *Sustainability*, vol. 14, no. 1, p. 198, Dec. 2021.

[37] H. Namgung and J.-S. Kim, "Collision risk inference system for maritime autonomous surface ships using COLREGs rules compliant collision avoidance," *IEEE Access*, vol. 9, pp. 7823–7835, 2021.

[38] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 739–753.

[39] D. K. Prasad, D. Rajan, L. Rachmawati, E. Rajabally, and C. Quek, "Video processing from electro-optical sensors for object detection and tracking in a maritime environment: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 8, pp. 1993–2016, Aug. 2017.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[41] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[42] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[43] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, Dec. 2015, pp. 91–99.

[44] S. Roheda, H. Krim, and B. S. Riggan, "Robust multi-modal sensor fusion: An adversarial approach," *IEEE Sensors J.*, vol. 21, no. 2, pp. 1885–1896, Jan. 2021.

[45] A. Serban, E. Poll, and J. Visser, "Adversarial examples on object recognition: A comprehensive survey," *ACM Comput. Surveys*, vol. 53, no. 3, pp. 1–38, May 2021.

[46] Z. Shao, W. Wu, Z. Wang, W. Du, and C. Li, "SeaShips: A large-scale precisely annotated dataset for ship detection," *IEEE Trans. Multimedia*, vol. 20, no. 10, pp. 2593–2604, Oct. 2018.

[47] C. Song, Y. Fan, A. Zhou, B. Wu, Y. Li, Z. Li, and K. He, "Regional adversarial training for better robust generalization," *Int. J. Comput. Vis.*, vol. 132, no. 10, pp. 4510–4520, Oct. 2024.

[48] D. Song, K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, and T. Kohno, "Physical adversarial examples for object detectors," in *Proc. 12th USENIX Workshop Offensive Technol. (WOOT 18)*, Jan. 2018, pp. 1–18.

[49] A. M. Staff, J. Zhang, J. Li, J. Xie, E. A. Traiger, J. A. Glomsrud, and K. B. Karolius, "An empirical study on cross-data transferability of adversarial attacks on object detectors," in *Proc. AI-Cybersec@ SGAI*, 2021, pp. 38–52.

[50] N. Tabish and T. Chaur-Luh, "Maritime autonomous surface ships: A review of cybersecurity challenges, countermeasures, and future perspectives," *IEEE Access*, vol. 12, pp. 17114–17136, 2024.

[51] M. J. Walter, A. Barrett, D. J. Walker, and K. Tam, "Adversarial AI testcases for maritime autonomous systems," *AI, Comput. Sci. Robot. Technol.*, vol. 2, pp. 1–16, Apr. 2023.

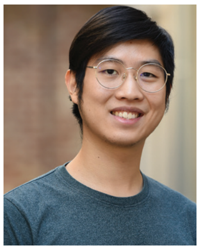
[52] J. Terven and D. Cordova-Esparza, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," 2023, *arXiv:2304.00501*.

[53] F. Tramèr, N. Carlini, W. Brendel, and A. Mądry, "On adaptive attacks to adversarial example defenses," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 1633–1645.

[54] A. Vagale, R. T. Bye, R. Oucheikh, O. L. Osen, and T. I. Fossen, "Path planning and collision avoidance for autonomous surface vehicles II: A comparative study of algorithms," *J. Mar. Sci. Technol.*, vol. 26, no. 4, pp. 1307–1323, Dec. 2021.

[55] C.-Y. Wang, A. Bochkovskiy, and H. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13024–13033.

[56] Y. Wang, Y.-A. Tan, W. Zhang, Y. Zhao, and X. Kuang, "An adversarial attack on DNN-based black-box object detectors," *J. Netw. Comput. Appl.*, vol. 161, Jul. 2020, Art. no. 102634.
[57] L. Wu, Z. Zhu, C. Tai, and E. Weinan, "Understanding and enhancing the transferability of adversarial examples," Tech. Rep., 2018.
[58] S. Sahil Abbas Zaidi, M. Samar Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," 2021, *arXiv:2104.11892*.
[59] Q. Zhang, Y. Zhao, Y. Wang, T. Baker, J. Zhang, and J. Hu, "Towards cross-task universal perturbation against black-box object detectors in autonomous driving," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107388.
[60] Y. J. Zhang, "A survey on evaluation methods for image segmentation," Tech. Rep., 1996.
[61] X. Zhou, D. Wang, and P. Krähenbuhl, "Objects as points," 2019, *arXiv:1904.07850*.
[62] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proc. IEEE*, vol. 111, no. 3, pp. 257–276, Mar. 2023.



PHORNPHAWIT MANASUT was born in Thailand. He received the bachelor's degree in computer science from Mahidol University, Thailand. After three years of working with the informatics industry, he returned to academia to pursue a master's degree, driven by his growing interest in artificial intelligence and image processing. His current research focuses on predicting phage-prokaryote interactions using geometric deep learning models.

MD SALEH IBTASHAM was born in Bangladesh.. He received the bachelor's degree in software engineering from the University of Dhaka, Bangladesh, and the master's degree in data-driven intelligent systems from Åbo Akademi University, Turku, motivated by his fascination with artificial intelligence. Currently, his research focuses on language modeling through retrieval-augmented generation frameworks.

ZEYNEP YARADANAKUL was born in Türkiye. She received the bachelor's degree in computer science, with minors in business analytics and decision and behavior from Sabancı University, Istanbul, Türkiye. She is currently pursuing the master's degree in data-intensive intelligent systems with Åbo Akademi University, Turku, Finland. She worked in IT project management, where she contributed to system development. Her current research focuses on exploring techniques and methodologies for document question answering.

SEPINOUD AZIMI received the Ph.D. degree from Åbo Akademi University, Finland, in 2015. She has held the title of Docent (equivalent to Associate Professor) in computer science with Åbo Akademi University, since 2018. She is an Assistant Professor with TPM (ESS-ICT). She is with the Department of Technology, Policy and Management, Delft University of Technology. She has over ten years of professional experience in teaching and research on biomedical topics and data analytics. Her main research interests include developing explainable AI (XAI) approaches in critical domains, such as the medical application domain and the use of AI in identifying blind spots related to inclusion and diversity.

SÉBASTIEN LAFOND was born in France. He received the M.Sc. (Tech.) degree from ESIGELEC and the master's degree in advanced studies from the University of Rouen, in 2000. In 2009, he defended his Ph.D. thesis from the Department of Information Technology, Åbo Akademi University. He is a Full Professor of software engineering with Åbo Akademi University. He was the principal investigator of the Business Finland Projects Decarbonizing Transport Corridors (DECATRIP) and Future Shipping Electrified (FUSE), working on software life cycle simulations and innovative digital twin solutions for maritime software systems. Within several national and international research projects, he has been working on the development of low-power computing platforms, low-latency media processing cloud-based solutions, and AI approaches in industrial software systems. His main research interests include energy-efficient many-core systems, parallel signal processing platforms, and AI-based approaches for automation and autonomous systems.

BOGDAN IANCU was born in Romania. He received the Diploma Engineer degree from the National University of Science and Technology POLITEHNICA Bucharest, Romania, in 2009, and the Ph.D. degree in computer science from the Department of Information Technology, Åbo Akademi University, Turku, Finland, in 2015. He received a docentship in computer science from Åbo Akademi University, in 2024. He has worked in both the academic and industrial sectors and is currently a University Lecturer with Åbo Akademi University. His research focuses on data ecosystems, data-centric AI, health data science, and maritime informatics.