Vessel Detection in Satellite Images using Deep Learning

Darshana Sankhe
Department of Electronics and Telecommunication Engineering., D. J. Sanghvi College of Engineering, Mumbai, Maharashtra, India
darshana.sankhe@djsce.ac.in

Snehal Bhosale
Department of Electronics and Telecommunication Engineering, Symbiosis Institute of Technology, Symbiosis International (Deemed University), Lavale, Pune, Maharashtra, India
snehal.bhosale@sitpune.edu.in (corresponding author)

Received: 16 August 2024 | Revised: 17 September 2024 | Accepted:4 October 2024

 DOI: https://doi.org/10.48084/etasr.8755

ABSTRACT

Maritime surveillance and monitoring have emerged as crucial components, serving various purposes such as security, environmental protection, and economic activities. This paper focuses on utilizing Synthetic Aperture Radar (SAR) satellite imagery to detect and track vessels in maritime regions. SAR technology provides notable advantages in imaging capabilities, enabling effective vessel detection under diverse weather conditions and during both day and night. Deep learning (DL) models are trained employing annotated SAR images, including multiple vessel patterns, sizes, and orientations. The enhancement of model generalization and robustness is accomplished by applying transfer learning techniques and data augmentation strategies, ensuring reliable detection performance across different environmental conditions and vessel types. By leveraging SAR imagery, this paper aims to contribute to enhanced maritime situational awareness, enabling timely identification of small vessels, including those involved in illegal fishing, smuggling, or other illicit activities. The results of this research hold promise for bolstering maritime security, aiding search and rescue operations, and facilitating effective regulation of maritime traffic.

Keywords-YOLO; deep learning; satellite images; vessel detection

I. INTRODUCTION

Maritime regions present a vast and dynamic landscape, teeming with diverse activities ranging from legitimate commerce to illicit endeavors, posing challenges for effective surveillance and monitoring. In recent years, the fusion of cutting-edge technologies has emerged as a pivotal approach to enhance maritime security and surveillance. Among these technologies, SAR satellite imagery, with its unparalleled capacity for all weather conditions and day-and-night imaging, stands as a transformative tool for comprehensive maritime monitoring. With the advancements in SAR technology, the integration of DL techniques has revolutionized the realm of image analysis and pattern recognition. The combination of SAR imagery and DL algorithms demonstrates a compelling approach for the precise and effective identification of ships within the complex and congested maritime domain [1].

The present research explores the synergy between SAR satellite imagery and DL methodologies to address the pressing need for developing vessel detection and monitoring. Small vessels, often engaged in multiple activities including but not limited to fishing, smuggling, terrorist activities, and illegal trafficking, pose a challenge to conventional surveillance systems due to their size, mobility, and ability to blend with their maritime surroundings. The use of extensive data provided by SAR imagery and the capabilities of DL models along with training the corresponding models are deployed to detect the vessels and landmasses successfully. This will revolutionize maritime surveillance capabilities, empowering authorities with advanced tools to detect, monitor, and respond to activities involving small vessels. Such advancements will fortify maritime security, mitigating illicit maritime activities, and fostering safer and more efficiently managed maritime domains. This research aims to refine vessel detection methods while also envisioning a fundamental change in maritime surveillance, ushering in a new era of comprehensive and proactive monitoring facilitated by the integration of SAR satellite imagery and DL methods [2-5]. The proposed system is inspired by the basic thought of You Only Look Once (YOLO) [6–9].

YOLO is a series of DL models designed for fast object detection. The objective of the YOLO series is to rapidly detect the locations and classifications of objects by analyzing the entire image simultaneously. By consolidating the detection process into one single neural network, YOLO enables direct optimization of the detection performance from start to finish. In this paper, three models from this series are utilized: YOLOv3 [10-12], YOLOv4 [13-15], and YOLOv4-tiny [16].

A. YOLOv 3

YOLOv3 is a state of art DL model designed for real-time object detection. It is renowned for its efficiency and accuracy, making it a popular choice in various fields like surveillance, image analysis and autonomous driving. At its core, YOLOv3 processes an input image through a darknet backbone and extracts meaning features from an image, which are then used by the detection head to predict bounding boxes and class probabilities for the objects being present [17]. The YOLOv3 architecture is optimized for real-time object detection with a balance of speed and precision. At its core is the Darknet-53, a 53-layer Convolutional Neural Network (CNN) that incorporates residual connections to facilitate gradient flow and reduce the risk of vanishing gradients. The model operates at three distinct scales-high, medium, and low resolution-allowing it to detect objects of varying sizes with the help of predefined anchor boxes. YOLOv3 also utilizes an FPN-like structure that merges feature maps from different layers, improving the detection of both small and large objects. Each grid cell predicts multiple bounding boxes with associated confidence scores and class probabilities, with non-max suppression applied to eliminate overlapping boxes. A combination of loss functions-bounding box regression, confidence scores, and classification-is used to fine-tune the model's accuracy across scales.

B. YOLOv4

It is a real-time object detection model developed to address the limitations of previous YOLO versions like YOLOv3 and other object detection models. Unlike other CNN based object detectors, YOLOv4 is not only applicable for recommendation systems, but also for stand-alone process management and human input reduction. Its operation on conventional GPUs allows for mass usage at an affordable price, and it is designed to work in real-time on a conventional GPU while requiring only one such GPU for training. The YOLOv4 architecture improves upon YOLOv3 by enhancing both accuracy and speed for object detection tasks. It utilizes a CSPDarknet53 backbone, which incorporates Cross Stage Partial (CSP) connections to reduce computational costs without sacrificing accuracy. YOLOv4 improves feature extraction with a Path Aggregation Network (PANet), which allows better information flow across layers, aiding in the detection of both small and large objects. It also introduces various optimization techniques, such as Mish activation, Mosaic data augmentation, and DropBlock regularization to enhance training efficiency. The model continues to use multi- scale predictions to detect objects of different sizes, along with anchor boxes and non-max suppression to refine the detection process. These advancements make YOLOv4 well-suited for real-time object detection with higher performance [18].

C. YOLOv4 - Tiny

It is a light variant version of the YOLOv4 object detection model designed for deployment on resource-constrained devices, such as embedded systems and cellular phones. It not only maintains the core principles of YOLOv4, but also prioritizes efficiency through modifications to the architecture [19]. The YOLOv4-tiny model is a streamlined version of YOLOv4, optimized for faster object detection while reducing computational demands, making it well-suited for use in resource-limited environments. It simplifies the architecture by using fewer convolutional layers and CSP connections to maintain efficient feature extraction. Unlike the full version, YOLOv4-tiny focuses on two prediction scales instead of three, prioritizing speed with some loss in accuracy, especially for smaller objects. The model continues to utilize anchor boxes and non-max suppression to refine detections and incorporates techniques like Leaky ReLU activation and batch normalization to ensure stable training. YOLOv4-tiny is particularly effective in real-time applications, such as embedded systems and mobile devices, where processing speed is crucial.

II. SYSTEM IMPLEMENTATION

A custom dataset was created using images from Sentinel-1. Sentinel-1 is a radar imaging satellite operated by the European Space Agency and its design allows images to be captured in all weather conditions. Due to its ability to penetrate clouds, it provides high resolution images and is widely used for maritime applications. This dataset has over 2500 SAR images, with each of them having dimensions of 1200 by 800 pixels, covering approximately an area of about 5 km by 5 km. Furthermore, all the images are manually annotated to detect small vessels as well as land, which is further utilized for distance calculation [20]. Some samples can be seen in Figure 1.

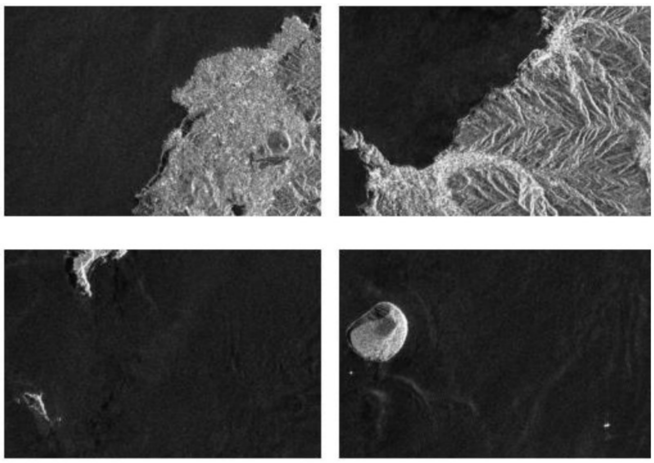

Fig. 1. Sample images in image dataset.

Once all the images are annotated, the dataset is structured to include both the original SAR images and corresponding annotations. Data processing techniques are also applied to the dataset, which includes augmentation as well to identify and remove images that are not clear or contain excessive noise [21-23]. As evidenced in Figure 2, by annotating, the algorithm identifies and labels the objects within an image, and in this case, the annotation task involves the detection of small vessels as well as the annotation of land areas, which will be used to calculate the distance between the vessels and the land. Moreover, this task is performed deploying software called labeling, which marks where a region of interest is located. Each image is carefully annotated to differentiate land and vessels.

A rectangular box is utilized to denote the position of an object within an image. This box typically includes the coordinates of its center (x, y), width (w), and height (h). The coordinates of the box are often normalized to values between 0 and 1, representing the relative position within the image. For instance, x and y might range from 0 to 1, while w and h represent the width and height as fractions of the total image size. Each annotated object in YOLO format is associated with a category label, indicating the type of the object being present. In this study, the category label might be a vessel and a land indicated by a class index in the format given below. Thus, 0 represents the vessel while 1 represents the land. YOLO annotations are commonly stored in text files. Each annotated image corresponds to a separate text file containing the object annotations. The annotations within a YOLO file typically follow a specific format, specifying the object class and its bounding box coordinates. For example, the format might be: class-index x-center y-center width height. The YOLO format presents each vessel annotation with its normalized bounding box coordinates, relative to the image's width and height, and the category label for identification purposes.

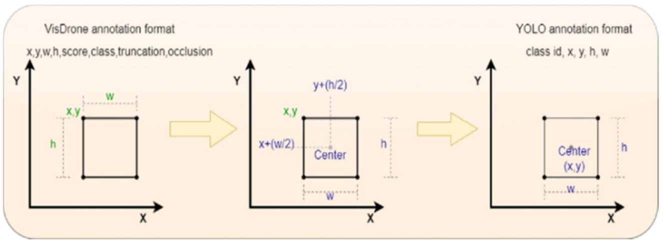

Fig. 2. Annotation architecture.

A. *Steps for Training YOLOv3*

A Google Colab notebook is opened and GPU acceleration enabled to expedite model training. Darknet is cloned and built from AlexeyAB's GitHub repository, configuring the Makefile to enable OPENCV and GPU for Darknet. Google Drive is mounted to the notebook to run detections. To create a custom YOLOv3 detector, the following components are required: labeled custom dataset, custom .cfg file, obj.data and obj.names files, and train.txt and test.txt files. Next, a dataset containing a large number of satellite images with annotated vessels is built and uploaded to the drive. Regarding the custom configuration file, the .cfg file is crucial in defining the object detection model's architecture and training parameters. The .cfg file is modified to suit the object detector needs. The max batch is set to 64 and subdivisioned to 16. The number of classes is changed to 2, as there are two classes to be detected: vessel and land. The obj.names file containing the class names, and the obj.data file is created specifying the backup path for model weights and the number of classes.

The pre-trained YOLOv3 weights, which have been trained on the COCO dataset with 80 classes, are leveraged to run YOLOv3 on these pre-trained classes and obtain detections. The custom object detector is trained and the Mean Average Precision, which in this case is 74.82%, is calculated.

B. *Training YOLOv4*

A Google Colab notebook is utilized to train the YOLOv4 model. GPU acceleration is enabled within the notebook to enhance the training speed. Darknet is cloned and built from AlexeyAB's GitHub repository, with the Makefile adjusted to enable OPENCV and GPU for Darknet, followed by the build process. Helper Functions are defined to display the images in the Colab Notebook after running the detections. A dataset comprising numerous satellite images is created, with vessels annotated and uploaded to the drive.

The .cfg file plays a crucial role in defining the object detection model's architecture and training parameters. The .cfg file is edited to suit the specific needs of the object detector. The Max batch is set to 64, and the subdivision is set to 16. The number of classes is changed to 2, as there are two classes to be detected: vessel and land. A new file, obj.names, is created, containing the names of the classes to be detected. Another file, obj.data, is created, which includes the backup path to save the model weights and number of classes.

Pre-trained YOLOv4 weights are downloaded. YOLOv4 has already been trained on the COCO dataset, which has 80 classes that it can predict. These pre-trained weights are utilized to run YOLOv4 and obtain detections. The Custom Object Detector is then trained, and the accuracy of the model is tested, achieving 82.96% in this case.

C. *Training YOLOv4-Tiny*

The darknet Git repository is cloned onto the Colab virtual machine. Within the 'yolo4-tiny' folder, a new directory named 'training' is created to store the trained weights, as specified in the 'obj.data' file that will be uploaded later. Next, the labeled custom dataset is uploaded as an 'obj.zip' file to the 'yolov4-tiny' folder on the drive. To create this 'obj.zip' file, the 'obj' folder is compressed, which should contain both the input image files in '.jpg' format and their corresponding YOLO format label files in '.txt' format. Helper functions are defined to enable image display in Colab Notebook after running the detections.

Regarding the custom configuration file, the '.cfg' file plays a crucial role in defining the object detection model's architecture and training parameters. As per the needs of the object detector, the '.cfg' file must be modified. The maximum batch size is set to 64, and the sub-divisions are set to 16. The number of classes has been changed to 2, as there are two classes to be detected: vessel and land. A new file named 'obj.names' is created, containing the names of the classes to be detected. Additionally, an 'obj.data' file has been created, which specifies the backup path to save the model's weights and the number of classes.

The pre-trained YOLOv4-tiny weights are downloaded and the custom object detector is trained. The performance is evaluated by checking the mean average precision, which is achieved to be 69.31%.

D. Distance Calculation

The distance calculation entails determining the Euclidean distance between each identified vessel and the nearest coastline. This process begins with the application of a specialized Machine Learning (ML) model trained for the object detection task at hand. The model has been trained on a dataset comprising annotated examples of vessels within satellite imagery. The model analyzes the input satellite images and identifies regions likely to contain vessels. These regions are delineated by bounding boxes, which enclose the spatial extent of each detected vessel. The bounding boxes essentially provide concise information about the location and size of the detected vessels within the satellite image. Each bounding box is defined by its top-left and bottom-right corner coordinates. Similarly, the same ML model is employed to identify landmasses using the trained object detection model. This model has been applied to satellite images to detect features indicative of land areas. Upon analyzing the input images, the model identifies regions corresponding to landmasses and marks them with bounding boxes. These bounding boxes encapsulate the spatial extent of each detected landmass.

After the vessels and landmasses have been detected and their bounding boxes determined, the next step involves calculating the centroid of each bounding box. The centroid represents the geometric center of the bounding box, and it is calculated as the midpoint between the top-left and bottom-right corner coordinates along both axes.

With the locations of vessels and landmasses now determined, the straight-line, or Euclidean, distance between each vessel and the nearest landmass is calculated. Euclidean distance refers to the direct distance between two points in a Euclidean coordinate system. For each vessel, the distance to all the identified landmass center points is computed using the Euclidean Distance formula:

$$Distance = \sqrt{(x_{ship} - x_{land})^2 + (y_{ship} - y_{land})^2} \quad (1)$$

The coordinates of the vessel's center and the landmasses' center are determined. The distance calculation is repeated for each vessel-landmass pair to ascertain the proximity of each vessel to the nearest landmass. Ultimately, the computed distance is annotated on the corresponding vessel's bounding box in the satellite image. This annotation conveys information about the spatial relationship between vessels and landmasses. Each vessel's bounding box displays the distance from the vessel to the nearest landmass. By precisely calculating the distance between each vessel and landmasses, the system furnishes essential information about the spatial dynamics within the satellite imagery, facilitating informed decision-making for maritime surveillance, environmental monitoring, and other relevant applications.

III. RESULTS AND DISCUSSION

The performance of YOLO models is decided by their mean Average Precision (Map) scores. The mAP uses the ground-truth bounding box, compares it to the detected box, and returns a score. The higher the mAP score is, the more accurate the model is. The mAP score of YOLOv4 model is the highest, as displayed in Table I, this means that it has a higher probability of detecting vessels and landmasses. Hence, the weights of the trained YOLOv4 model are further used to calculate the distance between the vessel and the landmasses.

TABLE I. MODEL'S MAP SCORE

Model	mAP
YOLOv3	74.82%
YOLOv4	82.96%
YOLOv4 Tiny	69.31%

Figures 3-6 present various cases of output images demonstrating ships and the landmasses by using the reference dataset [20].

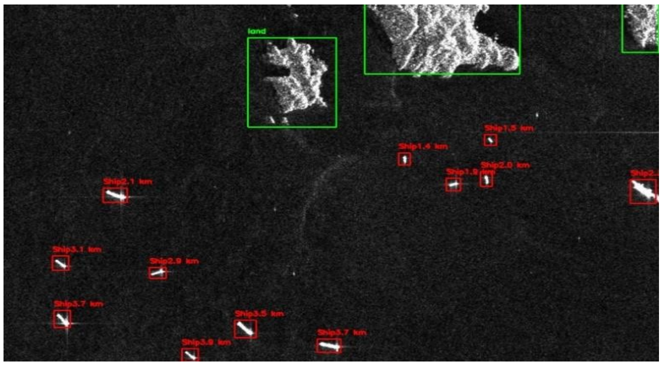

Fig. 3. Output image indicating vessel and landmass.

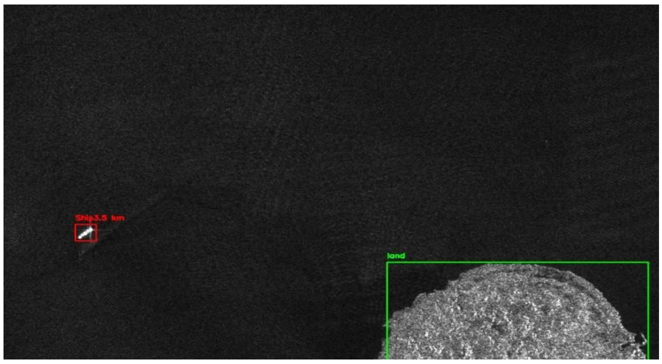

Fig. 4. Output image indicating vessel and landmass.

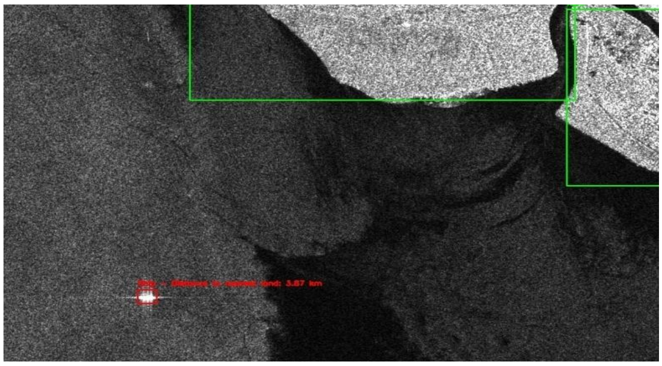

Fig. 5. Output image indicating vessel and landmass.

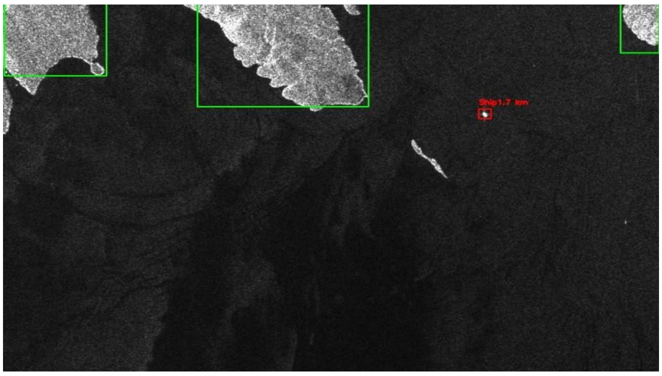

Fig. 6. Output image indicating vessel and landmass.

IV. CONCLUSION

The development of Machine Learning (ML) models for vessel and landmass detection in satellite imagery, coupled with distance calculations between vessels and their nearest landmass, represents a significant advancement in the field of maritime surveillance. Through the utilization of cutting-edge techniques and methodologies, the core objectives of the research work have been successfully addressed. The implemented system demonstrates robust performance in detecting vessels and landmasses within satellite imagery, as evidenced by rigorous testing. By leveraging trained Deep Learning (DL) models and sophisticated algorithms, the system has enabled automated detection and annotation of vessels and landmasses, marking them with bounding boxes and annotating each of them with its respective labels. Various versions of the You Only Look Once (YOLO) algorithms were utilized to address the research problem, with YOLO Version 4 (YOLOv4) demonstrating the highest accuracy after implementation. Furthermore, the integration of distance calculations enhances system utility by providing valuable spatial context. Accurately measuring the distance to the nearest landmass offers valuable insights for maritime surveillance, coastal management, and environmental monitoring initiatives. The scope of this work can be extended to real-time monitoring and alert systems, enabling proactive responses to maritime incidents, such as vessel collisions, illegal fishing, vessel hijackings by pirates, and environmental hazards.

ACKNOWLEDGMENT

The dataset used in this research work are sourced from the 'Large-Scale SAR Ship Detection Dataset-v1.0,' which is publicly available online [20].

REFERENCES

[1] L. Jiao *et al.*, "A Survey of Deep Learning-Based Object Detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, https://doi.org/10.1109/ACCESS.2019.2939201.

[2] J. Pei, Y. Huang, W. Huo, Y. Zhang, J. Yang, and T.-S. Yeo, "SAR Automatic Target Recognition Based on Multiview Deep Learning Framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 2196–2210, Apr. 2018, https://doi.org/10.1109/TGRS.2017.2776357.

[3] X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, "Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 10, pp. 1797–1801, Jul. 2014, https://doi.org/10.1109/LGRS.2014.2309695.

[4] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," *Image and Vision Computing*, vol. 97, May 2020, Art. no. 103910, https://doi.org/10.1016/j.imavis.2020.103910.

[5] T. Zhang and X. Zhang, "High-Speed Ship Detection in SAR Images Based on a Grid Convolutional Neural Network," *Remote Sensing*, vol. 11, no. 10, Jan. 2019, Art. no. 1206, https://doi.org/10.3390/rs11101206.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788, https://doi.org/10.1109/CVPR.2016.91.

[7] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525, https://doi.org/10.1109/CVPR.2017.690.

[8] W. Gai, Y. Liu, J. Zhang, and G. Jing, "An improved Tiny YOLOv3 for real-time object detection," *Systems Science & Control Engineering*, vol. 9, no. 1, pp. 314–321, Jan. 2021, https://doi.org/10.1080/21642583.2021.1901156.

[9] T. Saidani, "Deep Learning Approach: YOLOv5-based Custom Object Detection," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12158–12163, Dec. 2023, https://doi.org/10.48084/etasr.6397.

[10] D. Thuan, "Evolution of Yolo alorithm and Yolov5: The state-of-the-art object detection algorithm," B.S. Thesis, Oulu University of Applied Sciences, Oulu, Finland, 2021.

[11] S. S. Padmanabula, R. C. Puvvada, V. Sistla, and V. K. K. Kolli, "Object Detection Using Stacked YOLOv3," *Ingénierie des Systèmes d'Information*, vol. 25, no. 5, pp. 691–697, Aug. 2020, https://doi.org/10.18280/isi.250517.

[12] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv*, Apr. 2018, https://doi.org/10.48550/arXiv.1804.02767.

[13] J. Yu and W. Zhang, "Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4," *Sensors*, vol. 21, no. 9, Jan. 2021, Art. no. 3263, https://doi.org/10.3390/s21093263.

[14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *ArXiv*, Apr. 2020, https://doi.org/10.48550/arXiv.2004.10934.

[15] J. Woo, J.-H. Baek, S.-H. Jo, S. Y. Kim, and J.-H. Jeong, "A Study on Object Detection Performance of YOLOv4 for Autonomous Driving of Tram," *Sensors*, vol. 22, no. 22, Jan. 2022, Art. no. 9026, https://doi.org/10.3390/s22229026.

[16] Z. Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny," *ArXiv*, Dec. 2020, https://doi.org/10.48550/arXiv.2011.04244.

[17] A. Kathuria, "What's new in YOLO v3?," *Towards Data Science*, Apr. 2018, https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.

[18] J. Solawetz and J. Nelson, "PP-YOLO Surpasses YOLOv4 - State of the Art Object Detection Techniques," *Roboflow*, Aug. 03, 2020. https://blog.roboflow.com/pp-yolo-beats-yolov4-object-detection/.

[19] S. Saponara, A. Elhanashi, and Q. Zheng, "Developing a real-time social distancing detection system based on YOLOv4-tiny and bird-eye view for COVID-19," *Journal of Real-Time Image Processing*, vol. 19, no. 3, pp. 551–563, Jun. 2022, https://doi.org/10.1007/s11554-022-01203-5.

[20] T. Zhang *et al.*, "LS-SSDD-v1.0: A Deep Learning Dataset Dedicated to Small Ship Detection from Large-Scale Sentinel-1 SAR Images," *Remote Sensing*, vol. 12, no. 18, Jan. 2020, Art. no. 2997, https://doi.org/10.3390/rs12182997.

[21] T. Zhang, X. Zhang, J. Shi, and S. Wei, "HyperLi-Net: A hyper-light deep learning network for high-accurate and high-speed ship detection from synthetic aperture radar imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 167, pp. 123–153, Sep. 2020, https://doi.org/10.1016/j.isprsjprs.2020.05.016.

[22] T. Zhang and X. Zhang, "ShipDeNet-20: An Only 20 Convolution Layers and <1-MB Lightweight SAR Ship Detector," *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 7, pp. 1234–1238, Jul. 2021, https://doi.org/10.1109/LGRS.2020.2993899.

[23] N. Mo and L. Yan, "Improved Faster RCNN Based on Feature Amplification and Oversampling Data Augmentation for Oriented Vehicle Detection in Aerial Images," *Remote Sensing*, vol. 12, no. 16, Jan. 2020, Art. no. 2558, https://doi.org/10.3390/rs12162558.

[24] I. Singh and G. Munjal, "Modified YOLOv5 for small target detection in aerial images," *Multimedia Tools and Applications*, vol. 83, no. 18, pp. 53221–53242, May 2024, https://doi.org/10.1007/s11042-023-17625-7.