Adversarial Attack on Autonomous Ships Navigation Using K-Means Clustering and CAM

Ganesh Ingle, Kailas Patil, Sanjesh Pawale
Department of Computer Engineering, Vishwakarma University, Pune, India

***Abstract*—As Maritime Autonomous Surface Ships (MASSs) increasingly become part of global maritime operations, the reliability and security of their object detection systems have become a major concern. These systems, which play a crucial role in identifying small yet critical maritime objects such as buoys, vessels, and kayaks, are particularly susceptible to adversarial attacks, especially clean-label poisoning attacks. These attacks introduce subtle manipulations into training data without altering their true labels, thereby inducing misclassification during model inference and threatening navigational safety. The objective of this study is to evaluate the vulnerability of maritime object detection models to such attacks and to propose an integrated adversarial framework to expose and analyze these weaknesses. A novel attack method is developed using K-means clustering to segment similar object regions and Class Activation Mapping (CAM) to identify high-importance zones in image data. Adversarial perturbations are then applied within these zones to craft poisoned inputs that target the YOLOv5 object detection model. Experimental validation is performed using the Singapore Marine Dataset (SMD and SMD-Plus), and performance is measured under different perturbation intensities. The results reveal a considerable decline in detection accuracy—especially for small and mid-sized vessels—demonstrating the effectiveness of the attack and its capacity to remain imperceptible to human observers. This research highlights a critical gap in the security posture of AI-based navigation systems and emphasizes the urgent need to develop maritime-specific adversarial defense strategies for ensuring robust and resilient MASS deployment.**

***Keywords*—Maritime autonomous surface ships; object detection; clean-label poisoning attacks; adversarial attacks**

I. Introduction

Artificial Intelligence (AI) and Machine Learning (ML) are increasingly being deployed across critical domains such as healthcare, finance, defense, and autonomous transportation. In particular, the maritime industry has seen a transformative shift with the advent of Maritime Autonomous Surface Ships (MASSs), where AI-powered systems enable autonomous navigation, object detection, and situational awareness. The reliance of these systems on data-driven models, however, introduces new vectors for cyber-physical vulnerabilities, particularly those associated with training data integrity and model robustness.

Among the most significant threats to ML systems are data poisoning attacks, which involve the deliberate corruption of training data to manipulate model behavior. Such attacks can cause substantial performance degradation or induce specific, targeted misclassifications. While traditional data poisoning techniques typically involve altering both the features and labels of the training samples, recent work has highlighted the emergence of pure label poisoning attacks, wherein only the data is subtly manipulated while the labels remain unchanged (Turner et al., 2019; Saha et al., 2020). These attacks are particularly concerning because they can bypass standard data validation and noise detection protocols, making them harder to detect and mitigate.

The maritime domain presents a unique set of challenges for adversarial robustness. Object detection models used in MASSs must be capable of identifying small, dynamic, and often occluded objects such as buoys, small boats, and kayaks (Rekavandi et al., 2022; Shao et al., 2022). Misclassifying such objects due to adversarial manipulation can result in navigational errors with potentially severe consequences. Despite the growing use of deep learning models such as YOLOv5 for maritime object detection, current literature offers limited focus on adversarial risks specific to the maritime context. Most studies have concentrated on general adversarial attacks in image classification domains using datasets like CIFAR-10, ImageNet, or MNIST (Goodfellow et al., 2015; Madry et al., 2018), with minimal adaptation to marine scenarios and autonomous navigation systems.

This paper addresses this gap by proposing a novel adversarial attack framework tailored for maritime object detection systems, particularly those deployed in MASSs. The research objective is two-fold: (i) to demonstrate the feasibility and effectiveness of clean-label poisoning attacks in marine environments, and (ii) to develop an integrated methodology that leverages K-means clustering for unsupervised segmentation and Class Activation Mapping (CAM) for identifying high-saliency regions in the image space. By combining these techniques, the proposed method creates adversarial examples that are both functionally deceptive and visually imperceptible, targeting the YOLOv5 object detection model trained on the Singapore Marine Dataset (SMD and SMD-Plus). Data poisoning attacks pose a substantial threat to machine learning systems by exploiting vulnerabilities through the manipulation of training data, leading to erroneous predictions or decisions. Although advancements in machine learning have improved the detection of traditional data poisoning attacks, the rise of clean-label or pure label poisoning attacks—where input features are subtly altered without changing the labels—presents a more complex detection challenge [24]–[27], [30], [31].

These attacks typically follow a multi-stage process. First, attackers gather information about the target model and its training data from public datasets or distributionally similar sources [22], [25]. Then, during the poison sampling phase, specific instances are manipulated or synthetically generated to meet malicious objectives. In the manipulation phase, small but targeted perturbations are introduced to selected features. The tampered data is then injected into the training pipeline,

often by compromising data collection or storage systems [25], [27], [30], [31].

Retraining the model on this poisoned dataset can lead to performance degradation or targeted misclassifications. Attackers exploit the compromised model to induce incorrect behavior in downstream tasks [25], [27], [30].

To achieve this, the study employs a multi-stage process that includes dataset preprocessing, feature clustering, neural network training with CAM integration, perturbation injection, and retraining using poisoned data. The experiments are conducted under various levels of adversarial intensity, and the resulting impacts on model accuracy and misclassification rates are systematically evaluated. The findings reveal that even small perturbations focused on CAM-highlighted regions can cause the model to misidentify marine objects with high confidence, often without human-perceptible visual artifacts. In summary, this research makes three primary contributions: (1) it introduces a domain-specific adversarial attack strategy combining K-means and CAM; (2) it validates the vulnerability of deep learning-based maritime object detection models through experimental results; and (3) it provides actionable insights for future development of robust defense mechanisms tailored to maritime AI applications. To address these challenges, this paper presents a comprehensive methodology that integrates K-Means clustering and Class Activation Mapping (CAM) to generate clean-label poisoning attacks on object detection models within maritime environments. The remainder of this paper is structured as follows: Section II provides relevant background on MASS technologies and AI-driven object detection. Section III reviews related work in adversarial machine learning and maritime cybersecurity. Section IV introduces the core attack models and threat landscape. Section V details the proposed methodology, including the integration of clustering and CAM. Section VI outlines the attack generation process, followed by the experimental setup in Section VII. Section VIII presents and discusses the experimental results. Finally, Section IX concludes the study and outlines potential avenues for future research in adversarial defense mechanisms for maritime AI systems.

II. Background

Research in the field of Marine Autonomous Surface Systems (MASS) is rapidly evolving, driven by two interrelated priorities: the enhancement of object detection capabilities and the fortification of cybersecurity defenses. Object detection, especially the accurate recognition of small maritime objects such as buoys, kayaks, and small vessels, is crucial for safe autonomous navigation. Recent studies have focused on deep learning-based detection frameworks that address the unique visual complexity of marine environments. For instance, Rekavandi et al. (2022) proposed a comprehensive deep learning pipeline tailored for small object detection in maritime surveillance systems, emphasizing the need for high-resolution features and contextual understanding in oceanic scenes. Similarly, Shao et al. (2022) developed a multiscale object detection architecture optimized for autonomous ship navigation, which significantly improved the detection accuracy of small-scale targets by incorporating multi-level feature representations.

Parallel to advancements in perception systems, there is growing recognition of the cybersecurity challenges that accompany the deployment of AI-powered maritime systems. Wróbel et al. (2023) analyzed the applicability of traditional maritime safety indicators in the context of MASS and proposed a structured framework for assessing security readiness. Meanwhile, Li et al. (2023) employed network analysis methods to uncover critical risk factors and operational vulnerabilities in MASS ecosystems. Akpan et al. (2022) contributed a detailed threat assessment by cataloging cyber risks specific to maritime operations, including communication breaches, data manipulation, and GPS spoofing, and evaluated the effectiveness of prevailing countermeasures. Complementing these efforts, Ben Farah et al. (2022) conducted a systematic review of recent innovations in maritime cybersecurity, highlighting both the progress made and the gaps in existing defense mechanisms.

To further operationalize security evaluations, Walter et al. (2023) introduced a suite of competitive artificial intelligence (AI) test cases designed specifically for MASS platforms. Their methodology incorporates systematic reliability testing and adversarial scenario simulations, serving as a robust benchmark to assess the resilience of AI models under stress. This line of research provides critical insights into how adversarial robustness and safety compliance can be quantitatively measured in autonomous maritime systems, thereby contributing to both standardization and implementation practices.

III. Literature Review

Research into object detection and cybersecurity forms a foundational pillar for the advancement of Maritime Autonomous Surface Vessels (MASV), ensuring both efficient navigation and robust defense against operational risks and adversarial threats.

Rekavandi et al. have significantly contributed by offering an exhaustive review and practical guide focused on the challenges of small object detection in maritime surveillance. Their research identifies critical difficulties, such as low-resolution objects and environmental noise, and recommends deep learning-based strategies that leverage image and video data to enhance detection accuracy and reliability in complex maritime scenarios.

Building upon similar objectives, Shao et al. developed a multidimensional recognition model optimized explicitly for autonomous navigation. The model effectively addresses environmental complexities like varying lighting conditions, occlusions, and reflective surfaces, providing robust and precise detection of maritime objects such as buoys and boats. Their work underscores the necessity for a multiscale detection architecture to increase accuracy.

LiDAR technology integration has been thoroughly explored by Yao et al. (yao), who propose a methodology for simultaneous multi-target tracking and static mapping in nearshore maritime environments. Their approach integrates LiDAR data to significantly enhance the precision of tracking moving targets, providing a robust framework for operational safety and situational awareness in challenging maritime settings.

Yang et al. introduced FC-YOLOv5, an enhanced version of the YOLOv5 algorithm, specifically tuned for unmanned surface vehicles. Their FC-YOLOv5 model achieves remarkable performance gains both in detection accuracy and computational efficiency, clearly outperforming traditional algorithms, thus supporting practical real-time application in maritime contexts.

In parallel to detection capabilities, researchers have actively addressed operational and cybersecurity risks associated with Maritime Autonomous Surface Systems (MASS). Wrobel et al. adapted established maritime safety indicators to MASS applications, proposing a structured framework for their effective implementation and highlighting the intricate integration challenges these novel systems pose.

Li et al. have developed a sophisticated network analysis approach aimed at modeling and evaluating the intricate relationships among various operational risk factors inherent in MASS. Their methodology identifies and prioritizes critical risks, facilitating strategic resource allocation and targeted mitigation strategies [16-23].

The cybersecurity of maritime operations has been closely examined by Akpan et al. (akpan), who detailed the specific vulnerabilities and threats unique to maritime cyber operations. Their comprehensive risk assessments facilitate the development of targeted cybersecurity strategies. Complementing this, Ben Farah et al. (benfarah) have systematically reviewed the current landscape and future directions of maritime cybersecurity, offering a strategic vision that integrates emerging threats with advanced cybersecurity practices [1-9].

Extensive research into adversarial attacks and defenses on AI models, particularly Graph Neural Networks (GNNs) and CNN-LSTM frameworks, has highlighted several significant vulnerabilities and defensive shortcomings. Researchers have noted the inadequacy of existing gradient-based or heuristic perturbation techniques in identifying crucial nodes within GNNs. This limitation motivated research into interpretability techniques such as Class Activation Mapping (CAM) to systematically locate essential nodes, an area requiring further exploration and refinement for enhanced model resilience [9-16].

Ingle, G. et al. gives CNN-LSTM models used in power system applications also exhibit vulnerabilities under adversarial conditions, with current defenses like adversarial training and defensive distillation demonstrating limitations in both effectiveness and generalizability. Input Adversarial Training (IAT) emerges as a robust alternative, significantly improving model resilience without sacrificing performance. Ingle, G. et al. addresses broader adversarial defense strategies, recent studies suggest the underutilized potential of feature masking techniques, particularly when integrated with gradient modification strategies. Such hybrid approaches may offer a more balanced solution between maintaining accuracy and increasing robustness against sophisticated adversarial attacks.Ingle, G. et al. introduces adversarial robustness, the integration of Honey Badger Optimization techniques into GNN attacker models (EHBO) has demonstrated substantial improvements in attack efficacy and model evaluation robustness, setting a high standard for future adversarial testing and resilience benchmarks. Furthermore, Ingle, G. et al. presents optimizing bit-plane slicing through genetic algorithms has been shown to notably enhance resilience against common adversarial attacks (FGSM and DeepFool). This innovative technique significantly improves model recovery and defense capability, highlighting the potential for dynamic and adaptive defensive measures in adversarial contexts [32-36].

Lastly, recent work by Walter et al. underscores the importance of competitive AI testing paradigms designed explicitly for maritime autonomous systems. These competitive AI frameworks systematically uncover vulnerabilities and foster advancements in security measures, ensuring ongoing resilience against increasingly sophisticated adversarial techniques [28,29].

As MASV technologies mature, continued advancements in AI-driven object detection, coupled with proactive and adaptive cybersecurity defenses, remain imperative. This dual focus is crucial to enhancing the reliability, security, and operational effectiveness of maritime autonomous surface vessels.

IV. AI Security Threats and Attack Methods

AI attacks are broadly categorized into two types, based on the attacker's knowledge of the target model: black-box and white-box attacks. In black-box attacks, attackers possess no internal knowledge of the model's architecture, parameters, or training process, relying instead on external behaviors and outputs. Conversely, white-box attackers have complete knowledge of the model's internal structures and algorithms, allowing for precise manipulation.

Several well-established adversarial attack methods have emerged, notably the Fast Gradient Sign Method (FGSM), Iterative FGSM (I-FGSM), Momentum Iterative FGSM (MI-FGSM), and Projected Gradient Descent (PGD). FGSM creates adversarial examples by applying perturbations derived from the gradient of the loss function, intentionally causing models to produce erroneous predictions. I-FGSM extends this concept by iteratively applying smaller perturbations, refining the adversarial impact to achieve specific misclassifications. MI-FGSM introduces momentum into I-FGSM to enhance convergence speed and efficiency, while PGD systematically applies perturbations within defined limits to manipulate outcomes methodically and robustly.

Data poisoning attacks, another significant category of adversarial threats, target the training data to corrupt the learning process. Clean-label backdoor attacks involve inserting subtly altered or Trojan examples into training datasets without altering the labels, causing targeted misclassification during model deployment. This covert method is particularly dangerous, as detection during model training and validation phases is challenging. Backdoor triggers embedded in neural network models remain inactive during regular operation but are activated upon recognizing specific trigger patterns during inference.

Sophisticated poisoning techniques such as poison frog, convex hyperpolyhedron, and polar hyperpolyhedron algorithms have been developed to strategically introduce minimal but effective harmful data points or subtly reshape the geometry of the data distribution, influencing model decision boundaries and undermining reliability.

In the maritime domain, although object detection and cybersecurity research have experienced substantial growth, studies specifically addressing AI security threats remain limited and exploratory. Typically, AI-focused research emphasizes algorithmic defenses and data poisoning using conventional benchmark datasets like CIFAR-100. To bridge this gap, this study proposes a hypothetical attack scenario employing marine-specific datasets to execute a clean-label poisoning attack against the YOLOv5 object recognition model. The objective is to highlight the susceptibility of maritime AI applications to sophisticated poisoning attacks, thereby raising awareness among stakeholders and emphasizing the critical need for advanced research, vigilant monitoring, and robust defense mechanisms tailored specifically for maritime environments.

V. Methodology

The generation of adversarial attacks involves a multi-step process integrating K-means clustering with Class Activation Mapping (CAM), targeting object detection models. The process includes the following steps:

A. K-means Clustering for Object Detection

The K-means algorithm is employed to cluster similar marine objects within the Singapore marine dataset. Mathematically, the K-means process iteratively partitions the dataset into clusters to minimize the within-cluster variance. The algorithm is defined as: Given a dataset X with N data points, the objective is to partition the data into K clusters, minimizing the following cost function:

$$J(c, \mu) = \sum_{k=1}^{K} \sum_{n=1}^{N} \|x_n - \mu_k\|^2 \tag{1}$$

Where: - x_n represents the n-th data point in the dataset.
- μ_k denotes the centroid of the k-th cluster.
- c is the cluster assignment for each data point.

The K-means algorithm seeks to minimize the cost function by iteratively updating the cluster centroids and reassigning data points to clusters until convergence.

The steps of the K-means algorithm involve: **Initialization:** Start by randomly initializing K cluster centroids $\mu = \{\mu_1, \mu_2, ..., \mu_K\}$. **Assignment Step:** Assign each data point x_n to the nearest centroid μ_k:

$$c^{(n)} = \arg\min_k \|x_n - \mu_k\|^2 \tag{2}$$

Update Step: Recompute the centroids based on the assigned data points:

$$\mu_k = \frac{1}{|S_k|} \sum_{x_n \in S_k} x_n \tag{3}$$

Where S_k represents the set of data points assigned to cluster k. The above steps are iterated until convergence or a maximum number of iterations is reached. Once the clusters are identified, they serve as the basis for the subsequent phases, such as feature extraction or identification of regions for object detection using Class Activation Mapping (CAM). The K-means algorithm is an iterative process that continues until convergence or a predetermined maximum number of iterations is reached. This iterative nature is crucial for refining cluster assignments and centroids to minimize the within-cluster variance effectively. The steps involved in each iteration, including initialization, assignment, and update, collectively work towards achieving a stable configuration of clusters.

B. Convergence Criterion

Convergence in the K-means algorithm is determined through an evaluation of cluster assignments and centroids at consecutive iterations. The algorithm checks whether these assignments and centroids undergo significant changes. Specifically, it assesses whether the alterations fall below a predefined threshold. Alternatively, convergence is acknowledged if a predetermined maximum number of iterations is reached. In essence, if the changes in cluster assignments and centroids become sufficiently small or if the algorithm completes the specified number of iterations, it is considered to have converged. This convergence criterion ensures that the K-means algorithm stabilizes, indicating that further iterations would result in minimal adjustments to cluster assignments and centroids.

C. Role of Identified Clusters

Once the K-means algorithm converges, the identified clusters serve as the foundation for subsequent phases in the data analysis pipeline. These phases often include:

1) Feature extraction: The meaningful segmentation of data into clusters allows for the extraction of representative features within each cluster. Feature extraction refers to the process of capturing distinctive characteristics or relevant information within these clusters, facilitating a more profound understanding of the inherent structure of the data. This involves identifying and quantifying the key attributes or patterns that differentiate one cluster from another. Precisely, feature extraction aims to distill the most relevant and discriminative information from the clustered data, enabling a more concise representation that can be utilized for subsequent analysis or interpretation. The extracted features serve as essential descriptors, providing insights into the distinctive properties of each cluster and contributing to a more nuanced comprehension of the underlying data distribution.

2) Object detection Using Class Activation Mapping (CAM): The identified clusters serve as a valuable resource for object detection tasks, especially when employing Class Activation Mapping (CAM). CAM is a technique designed to emphasize the specific regions within an image that have the greatest influence on predicting a particular class. In the realm of marine object detection, CAM can be effectively applied to concentrate on regions within the clustered data that are correlated with distinct marine objects. This involves utilizing the CAM technique to generate a spatial attention map that highlights the significant areas within the clustered

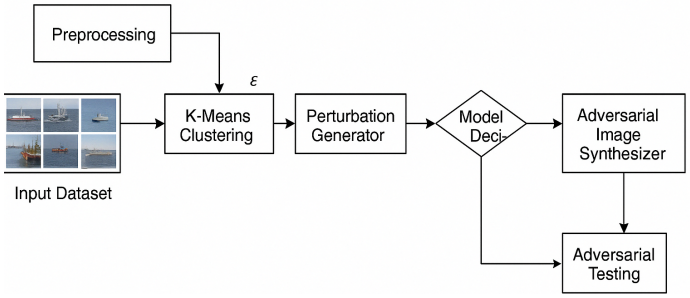

Fig. 1. Diagram of the proposed experiment.

data, providing insights into the regions contributing most to the presence or characteristics of specific marine objects. Precisely, CAM aids in pinpointing the crucial features within the clustered data that contribute to the identification and localization of marine objects, enhancing the interpretability and effectiveness of object detection in marine environments. Fig. 1 shows diagram of the proposed experiment.

D. Detailed Description

The K-means algorithm iteratively refines clusters, enhancing their representativeness of underlying data patterns. Convergence ensures stability in cluster assignments, indicating further iterations are unlikely to yield significant changes. With stable clusters, subsequent phases leverage this segmentation. Feature extraction captures each cluster's essence, providing nuanced insights into the dataset. These features prove instrumental in subsequent analyses or decision-making. Object detection using CAM takes advantage of identified clusters by pinpointing regions of interest linked to specific marine objects. CAM highlights areas in clustered data significantly contributing to object classification, aiding precise object localization in the maritime environment.The iterative convergence of the K-means algorithm sets the stage for meaningful analyses, enhancing the data processing pipeline's overall effectiveness in tasks like feature extraction and CAM-based object detection. The K-means algorithm clusters marine objects within the Singapore dataset by iteratively minimizing within-cluster variance. Mathematically, the objective function $J(c, \mu)$ minimizes the sum of squared Euclidean distances between data points x_n and assigned centroids μ_k:

$$J(c, \mu) = \sum_{k=1}^{K} \sum_{n=1}^{N} \|x_n - \mu_k\|^2 \tag{4}$$

Here, x_n is the n-th data point, μ_k is the k-th centroid, and c is each data point's cluster assignment. The algorithm minimizes this cost function through iterative centroid updates and data point reassignments until convergence. K-means steps include initialization, assignment, update, and convergence check. Initialization sets K centroids, and assignment associates data points with nearest centroids:

$$c(n) = \arg\min_{k} \|x_n - \mu_k\|^2 \tag{5}$$

Update recalculates centroids based on assigned data points:

$$\mu_k = \frac{1}{|S_k|} \sum_{x_n \in S_k} x_n \tag{6}$$

Iterations continue until convergence, ensuring stable centroids and minimal changes in cluster assignments. Widely used for clustering, K-means identifies similar marine object groups, enhancing marine environment analysis.

E. Object Detection using Class Activation Mapping (CAM)

The Class Activation Mapping (CAM) technique utilizes the learned weights from the last convolutional layer of a Convolutional Neural Network (CNN) to highlight crucial regions influencing the classification process. CAM visualizes where the model focuses during predictions. For a pre-trained CNN, let f_θ be the output feature map of the last convolutional layer. CAM generates a localization map M for a class c:

$$w_c = \frac{1}{Z} \sum_{i} \sum_{j} f_\theta^c(i, j) \tag{7}$$

Here, $f_\theta^c(i, j)$ is the activation at (i, j) for class c, and Z is the number of positions. The class-discriminative map is generated by combining feature maps with their weights:

$$M_c(i, j) = \sum_{k} w_c(k) \cdot f_\theta^c(i, j, k) \tag{8}$$

Here, $f_\theta^c(i, j, k)$ is the activation of the k-th filter at (i, j) for class c. To visualize influential regions, M_c passes through a ReLU activation function:

$$L_c(i, j) = \max(0, M_c(i, j)) \tag{9}$$

The final activation map L_c indicates regions significantly contributing to the classification of class c. CAM-based object detection reveals these important regions, indicating parts of marine images influencing marine object classification.

F. Adversarial Attack Strategy Using K-means and CAM

The adversarial attack strategy leverages insights from K-means clustering and Class Activation Mapping (CAM) to perturb data points and induce misclassifications in the object detection model. With clustered regions represented by C_i and associated data points N_i from K-means, and influential regions identified by CAM denoted as $L_c(i,j)$ for a specific class c, the adversarial attack aims to subtly alter the data points.

For a given cluster C_i, perturbed points P_i are generated by introducing slight modifications using the activation map $L_c(i,j)$:

$$P_i = N_i + \epsilon \cdot L_c(i,j) \tag{10}$$

Here, ϵ represents a small perturbation factor, ensuring subtle yet impactful changes. The objective is to manipulate the data points in a way that the object detection model misclassifies them. This approach combines the clustering information from K-means with the spatial understanding provided by CAM, demonstrating the potential vulnerabilities in the model's decision-making process. The attack strategy highlights the importance of addressing security concerns in object detection systems, particularly those employing clustering and spatial feature analysis.

G. Experimental Validation of the Attack Scenario

To validate the effectiveness of the attack scenario, experiments are designed according to the proposed hypothetical situation. The Singapore marine dataset is manipulated using the regions identified by K-means clustering and the features highlighted by Class Activation Mapping (CAM) from the Convolutional Neural Network (CNN).The Singapore marine dataset, denoted as $\mathcal{D}$, is partitioned into clusters by the K-means algorithm, resulting in clusters $\{C_1, C_2, \ldots, C_k\}$.Furthermore, the CAM highlights the significant regions or features in the marine images relevant for classification. Let the CAM-relevant features be denoted as F. The union of these features across different clusters is represented as $F = \bigcup_{i=1}^{k} F_i$, where F_i is the set of relevant features in cluster C_i.An adversarial attack strategy is executed, perturbing these identified and significant regions in the dataset. Mathematically, perturbing a specific feature $f \in F$ in a data point x is achieved by adding a small perturbation δ:

$$x' = x + \delta \quad \text{where} \quad f \in F \tag{11}$$

These perturbations aim to cause misclassifications in the object detection model by manipulating the significant regions identified through the clustering and CAM techniques.The impact of the misclassification is then assessed to determine the vulnerability of the model to such targeted adversarial attacks.

VI. Detailed Procedure for Generating Adversarial Attacks

For a comprehensive understanding, a detailed procedure for adversarial attack generation utilizing K-means clustering and CAM is as follows:

A. K-means Clustering Phase

The K-means clustering process involves the following steps:

1) Data preprocessing: The marine dataset, denoted as $\mathcal{D}$, undergoes preprocessing to meet the requirements of the K-means clustering algorithm. This step might involve normalization, handling missing data, or feature scaling to prepare the data for clustering.

2) Clustering: The K-means algorithm is then applied to the preprocessed marine dataset to identify distinct clusters. Let n be the total number of data points in the dataset and k be the desired number of clusters. The K-means algorithm aims to minimize the sum of squared distances between data points and their respective cluster centroids. This minimization is represented by the following mathematical equation:

$$\arg\min_{C} \sum_{i=1}^{n} \min_{\mu_j \in C} ||x_i - \mu_j||^2 \tag{12}$$

Where:

- C represents the set of clusters $\{C_1, C_2, \ldots, C_k\}$, and each cluster contains data points associated with its centroid μ_j.

- x_i denotes the i-th data point in the dataset.

- μ_j represents the centroid of the j-th cluster.

- $||x_i - \mu_j||$ denotes the Euclidean distance between a data point and its respective cluster centroid.The algorithm iteratively assigns data points to the nearest cluster centroid and updates the centroids based on the mean of the data points in each cluster. This process continues until convergence or a specified number of iterations.

B. CAM and Object Detection Phase

The CAM and object detection phase involves the following steps:

1) Neural network training: Train a Convolutional Neural Network (CNN) model specifically designed for object detection, with a focus on integrating Class Activation Mapping (CAM) into the network architecture. The CNN is trained using the marine dataset, denoted as $\mathcal{D}$, in which the goal is to predict object classes within the images.

2) Identifying important regions: Class Activation Mapping (CAM) is utilized to identify crucial regions within the marine images that contribute significantly to the classification decisions of the trained model. The CAM technique computes the class-specific activation map using the final convolutional feature maps and the model's output weights. This enables the identification of regions that highly influence the model's decision-making process, contributing to object classification.CAM involves mapping class-specific information to the spatial locations of the feature maps. Specifically, given the final feature maps F obtained from the last convolutional layer and the weights w of the output layer, the class activation map M for a particular class c is computed by performing global average pooling on the final feature maps followed by a linear combination using the weights:

$$M_c = \sum_i w_i^c F_i \tag{13}$$

Where: - M_c is the class activation map for class c.

- w_i^c represents the weights for class c of the i-th convolutional feature map.

- F_i denotes the i-th feature map obtained from the final convolutional layer.

The resultant class activation map indicates the most crucial regions within the images that contributed to the model's decision for predicting a particular class c.

C. Adversarial Attack Generation Phase

The Adversarial Attack Generation phase encompasses the following stages:

1) Adversarial perturbation: The vital regions identified by Class Activation Mapping (CAM) and clustered through K-means are perturbed to generate adversarial attacks. By manipulating these essential regions, subtle alterations are introduced to the original data points. The aim is to prompt misclassifications in the object detection model. The adversarial perturbation process involves tweaking the identified regions to influence the model's decision-making. Assuming X represents the marine image dataset and X_i denotes individual images, let X_i' be the perturbed images derived from the perturbation of important regions, such that:

$$X_i' = X_i + \epsilon \cdot \text{Perturbation}_i \tag{14}$$

Where: - X_i' signifies the perturbed image derived from the original image X_i.

- ϵ represents the magnitude of perturbation.

- Perturbation_i indicates the specific perturbation applied to image i.

2) Poisoning algorithm: Following the perturbation of important regions, a clean-label poisoning algorithm is employed in these perturbed regions. The poisoning algorithm aims to generate instances that are subtly corrupted and embedded back into the dataset for retraining the object detection model.The poisoning algorithm is crucial for introducing manipulated data instances into the training dataset while maintaining a seemingly benign appearance. This algorithm seeks to insert trojan or poisoned examples in a manner that avoids suspicion but triggers substantial misclassifications during testing. Let X'_{poisoned} denote the manipulated images generated by the poisoning algorithm.The process can be formulated as:

$$X'_{\text{poisoned}} = X' + \text{Algorithm}_{\text{poison}}(X') \tag{15}$$

Where: - X'_{poisoned} represents the images manipulated by the poisoning algorithm.
- $\text{Algorithm}_{\text{poison}}$ stands for the clean-label poisoning algorithm applied to the perturbed images.
- X' signifies the perturbed images.

D. Scenario Validation Through Experiments

The process of validating the attack scenario involves a sequence of critical steps:

1) Dataset preparation: The dataset needs to be segregated into two subsets: the training dataset and the malicious subset. The training dataset contains the unaltered, clean images, while the malicious subset includes images poisoned by the introduced adversarial attack. The formulation of the datasets is expressed as follows:

Let D represent the Singapore marine dataset. Divide this dataset into two subsets: D_{training} and $D_{\text{malicious}}$.

2) Poisoning and retraining: Inject the adversarially perturbed or poisoned instances into the training dataset. Subsequently, the object detection model is retrained using the newly modified dataset.

This process can be mathematically described as: Let M represent the object detection model. Retrain the model M using the combined dataset $D_{\text{training}} \cup D_{\text{malicious}}$.

$$M' = \text{Retrain}(M, D_{\text{training}} \cup D_{\text{malicious}}) \tag{16}$$

Where M' denotes the retrained object detection model.

3) Attack execution: Assess the effectiveness and impact of the adversarial attack by executing the attack scenario on the retrained model M'. This evaluation involves providing inputs to the model and examining its behavior concerning misclassification and vulnerabilities. This execution involves scrutinizing the model's performance with test instances and examining whether the attack scenario (misclassification of boats as ferries) materializes. This validation is vital to understand the vulnerabilities and consequences of such attacks on the model's behaviour.

VII. Algorithm

1) Prepare the Singapore Marine Dataset:
   - Load the marine dataset containing images of marine environments around Singapore.
2) Apply K-means Clustering:
   - Use K-means clustering to identify distinct classes within the marine dataset.
3) Train CNN with CAM:
   - Train a Convolutional Neural Network (CNN) with Class Activation Mapping (CAM) using the marine dataset.
4) Initialize Perturbation Factor:
   - Set the perturbation factor ϵ to a predefined value.
5) Generate CAM Heatmaps:
   - For each marine image I in the dataset:
   - Generate CAM heatmaps to highlight important regions for classification.
6) Extract Activation Regions:
   - Extract the class activation regions from the CAM heatmaps.
7) Apply K-means to Activation Regions:

- Apply K-means clustering to the extracted activation regions.

8) Choose Centroid for Each Cluster:
   - For each cluster, choose the centroid representing the region of interest.
9) Generate Adversarial Perturbation:
   - For each centroid region C:
   - Perturb C to generate adversarial perturbation δ.
10) Apply Perturbation to Image:
   - For each marine image I:
   - Apply the perturbation δ within the region represented by C.
11) Generate Adversarial Image:
   - Generate adversarial image I' by integrating the perturbed region back into the original image.
12) Build Adversarial Dataset:
   - Add the generated adversarial images I' to the adversarial dataset.
13) Repeat for All Images:
   - Repeat steps 5-12 for each image in the marine dataset.
14) Return Adversarial Examples:
   - Return the adversarial examples generated from the marine dataset using K-means clustering with CAM.

VIII. Experimental Setup

A. Dataset Preparation and Combination

The experimental setup leverages the Singapore Maritime Dataset (**SMD**) and its enhanced version, **SMD-Plus**, to address challenges in the maritime industry. With over 2 million vessel movements, SMD provides a comprehensive dataset for analyzing maritime behavior. However, to enhance precision, SMD-Plus corrects labeling errors and introduces more accurate bounding boxes, making it a valuable resource for object classification. Challenges in classifying small maritime objects are addressed by combining classes in SMD-Plus, enriching the dataset for improved recognition. The adaptation process involves transforming SMD-Plus videos into individual frames, aligning annotations with YOLOv5 specifications. This meticulous approach ensures seamless integration with the YOLOv5 object detection model, a crucial step in maximizing the dataset's compatibility and effectiveness for experimentation.

B. Hardware Used

The computational efficiency of the clustering step, particularly the K-means algorithm, is significantly influenced by the hardware specifications in use. For this purpose, the central processing unit (CPU) selected is the Intel Core i9-10900K from the Comet Lake architecture. This CPU features 10 cores and 20 threads, with a base clock of 3.7 GHz and a maximum turbo frequency of 5.3 GHz. Its 125W thermal design power (TDP) and 14nm manufacturing process contribute to robust performance in tasks that require parallel processing, such as K-means clustering.

On the graphics processing unit (GPU) side, the NVIDIA GeForce RTX 3080 is employed. This GPU boasts 8704 CUDA cores and is equipped with 10 GB of GDDR6X memory, featuring a 320-bit memory bus and a high-speed 19 Gbps memory. The GPU incorporates dedicated hardware components, including 68 ray tracing cores and 272 Tensor Cores, which enhance its parallel processing capabilities. This aligns well with the demands of deep learning tasks, including forward and backward propagation in graph neural networks (GNNs).

The system is further outfitted with 32 GB of DDR4 RAM and a 1TB NVMe SSD to facilitate swift storage access. It operates on the Windows 10 Pro operating system. For deep learning tasks, PyTorch 1.9.0 serves as the primary framework, while scikit-learn 0.24.2 is utilized for the K-means clustering library.

This combination of high-performance CPU and GPU, accompanied by ample system memory and fast storage, establishes a well-balanced hardware configuration capable of efficiently executing both deep learning and clustering operations. This configuration is vital for carrying out the proposed adversarial attack on graph neural networks.

C. Data Extraction and Annotation Modification

The experimental pipeline began with the preprocessing of the SMD-Plus dataset, an enhanced maritime surveillance video dataset comprising annotated recordings of vessels, buoys, ferries, and other maritime entities. The first critical step was the extraction of representative still frames from the video sequences. This was achieved by sampling one frame per fixed interval (e.g., every nth frame), ensuring a balance between temporal redundancy and dataset volume. The goal was to obtain a sufficient number of spatially and contextually varied images that reflect different lighting conditions, object positions, and environmental dynamics typical of real-world maritime scenes.

Each extracted frame was saved in a standard image format (e.g., .jpg or .png) and then indexed systematically to maintain traceability with its original video source. This frame-level extraction enabled the creation of a large and diverse image dataset suitable for training static object detection models like YOLOv5, which do not directly process video input.

Following frame extraction, the annotation conversion process was conducted to adapt the dataset for use with the YOLOv5 object detection framework. The SMD-Plus dataset originally provides annotations in the COCO (Common Objects in Context) format—a widely adopted standard for object detection tasks, structured as a hierarchical JSON file. This format includes metadata such as image IDs, category IDs, bounding box coordinates in the form of $[x_{\text{min}}, y_{\text{min}}, \text{width}, \text{height}]$, image dimensions, segmentation masks, and object categories as strings.

In contrast, YOLOv5 requires annotations in a simplified plain text format, where each image is paired with a .txt file bearing the same filename. Each line in the .txt file corresponds to one object in the image and consists of five fields: These coordinates are expected to be normalized by the image width and height, such that all values lie within the range $[0, 1]$. The conversion process involved the following key steps:

1) Parsing COCO annotations: The original COCO-format JSON file was parsed using Python libraries such as `pycocotools` or `json`, allowing access to image metadata, bounding box coordinates, and class labels.

2) Class mapping: A custom mapping from COCO's category names (e.g., `"ferry"`, `"kayak"`, `"buoy"`) to integer-based YOLO class IDs (e.g., 0, 1, 2, ...) was established to ensure consistency with YOLO's requirements.

3) Bounding box transformation: Each bounding box was converted from COCO's top-left-based format (x, y, w, h) to YOLOv5's center-based normalized format. This required the following computations:

$$x_{\text{center}} = \frac{x + \frac{w}{2}}{W}, \tag{17}$$

$$y_{\text{center}} = \frac{y + \frac{h}{2}}{H}, \tag{18}$$

$$w' = \frac{w}{W}, \tag{19}$$

$$h' = \frac{h}{H} \tag{20}$$

where (x, y) are the top-left bounding box coordinates, (w, h) are the box width and height, and W and H are the original image width and height, respectively. The resulting four values are thus scaled to lie within $[0, 1]$.

4) Annotation file generation: For each image, a corresponding `.txt` file was created containing one line per object instance. Each line consisted of the class ID and the four normalized coordinates in the format:

5) Validation: A visual inspection and manual verification process was carried out using annotation tools (e.g., Roboflow Annotator, CVAT, or `labelImg` with YOLO overlay enabled) to ensure the integrity of the converted annotations and accuracy of the bounding boxes.

This COCO-to-YOLOv5 annotation conversion was crucial for enabling the training of the YOLOv5 object detection model, which is optimized for real-time inference tasks on static images. The simplified YOLO format also significantly reduces annotation parsing overhead during training, making it suitable for high-speed detection in resource-constrained maritime environments.

By performing this structured transformation, the dataset was rendered fully compatible with the YOLOv5 architecture, ensuring that the object detector could accurately learn to detect and localize maritime objects under varying environmental conditions. This preparation step laid the foundation for all subsequent experimental workflows in object detection and adversarial attack simulation.

D. Malicious Attack Simulation and CAM Integration

The experimental process began with the frame-level decomposition of video sequences from the SMD-Plus (Singapore Marine Dataset - Enhanced) dataset. The SMD-Plus dataset comprises high-resolution maritime surveillance videos containing diverse vessel types such as boats, kayaks, buoys, ferries, sailboats, and other marine objects. To convert this video data into a format suitable for image-based object detection tasks, a representative still image was extracted from each video frame at a predefined sampling interval. This frame extraction step was essential to generate a large and diverse pool of labeled images from continuous video streams, enabling the object detection model to learn from both spatial and temporal variations in the data.

Following frame extraction, the next critical step involved converting the annotation format from COCO (Common Objects in Context) to the YOLOv5-compatible format. The original SMD-Plus dataset annotations were structured according to the COCO JSON schema, which includes complex metadata such as image IDs, category names, bounding box coordinates in absolute pixel units (x, y, width, height), segmentations, and image dimensions. While COCO format is widely used for benchmarking across multiple object detection tasks, it is incompatible with the training requirements of the YOLOv5 framework without transformation.

The YOLOv5 format, by contrast, expects annotations in a minimalist, plain-text .txt format for each image, with one line per object. Each line contains five values: the object class ID, followed by the normalized center x-coordinate, center y-coordinate, width, and height of the bounding box. These values are normalized with respect to the image width and height, i.e., they fall in the range [0, 1], which ensures model generalization regardless of input resolution.

The transformation from COCO to YOLOv5 format involved several sub-steps:

Mapping Class IDs: The COCO category names (e.g., "ferry", "kayak") were mapped to corresponding numeric class labels as required by YOLOv5.

Annotation Synchronization: Each image extracted from the video was assigned a .txt annotation file with the same filename. This ensured seamless integration with YOLOv5's data loader, which associates each image with its corresponding annotation during training. This conversion ensured that the YOLOv5 model could efficiently ingest and interpret the dataset during training and inference. Additionally, by using normalized coordinates and simplified annotation structures, the model achieved better consistency in processing varying image resolutions, a crucial requirement given the dynamic camera perspectives in maritime surveillance footage. This meticulous preparation of data not only preserved the integrity of the original labels but also optimized the dataset for high-performance object detection under the YOLOv5 framework.

E. Dataset Split for Training and Testing

During the training and validation phase of the object recognition model, careful attention was paid to dataset stratification to ensure representative and unbiased learning. The enhanced SMD-Plus dataset, which includes a diverse set of labeled maritime images, was partitioned using an 80:20 split, where 80% of the data was reserved for training and 20% for testing. This stratification was not merely random; rather, it was stratified based on class distribution to maintain balance among categories such as boats, ferries, buoys, and kayaks. Ensuring proportional representation across classes in both subsets was critical to avoid class imbalance issues

TABLE I. PROPERTIES OF THE SMD DATASET

Class	Class Identifier	Objects
Boat	1	14,550
Vessel and ship	2	126,301
Ferry	3	3,689
Kayak	4	3,872
Buoy	5	3,521
Sailboat	6	1,782
Others	7	25,214

that could bias the model or undermine its generalization capabilities. This partitioning also guaranteed that performance metrics reported during testing reflect the model's behavior on previously unseen instances, providing a robust evaluation of detection accuracy and adversarial robustness.

In addition to the clean training dataset, a malicious dataset was generated to simulate adversarial attack scenarios. Specifically, base instances were extracted from video frames featuring barges, serving as neutral examples, while target instances were derived from frames showcasing boats, which were the intended misclassification targets. These frames were used to craft adversarial samples through the Poison Frog algorithm, a clean-label data poisoning method designed to subtly corrupt the model's learning process without introducing conspicuous artifacts.

For the attack execution, we used ResNet50—a deep convolutional neural network known for its strong representational power—as the underlying architecture for crafting the poisoned representations. The Poison Frog algorithm was configured with the following hyperparameters to balance imperceptibility and effectiveness:

Iterations: 5000 (to allow gradual and subtle updates),

Epsilon: 0.02 (maximum perturbation magnitude),

Alpha: 0.001 (step size per iteration during gradient-based optimization).

These parameters were carefully selected to ensure that the perturbations introduced during the poisoning process remained invisible to human observers, even upon close inspection. As a result, the poisoned images retained their original appearance, making them ideal for clean-label attacks where the attacker does not modify the class label and thus avoids triggering human or automated suspicion.

Visual inspection of the generated adversarial examples, as shown in Fig. 7, confirmed the absence of visible perturbations, despite the internal activation manipulations induced by the attack. Interestingly, after incorporating the poisoned instances into the training set and retraining the YOLOv5 object detection model, it was observed that the model began to misclassify boats as plaques with high confidence. This outcome highlights the subtle yet impactful influence of the clean-label attack and underlines the importance of adversarial resilience in critical domains like maritime navigation.

This experiment demonstrates the effectiveness of targeted data poisoning in altering model behavior without altering data labels or image realism—underscoring the urgency for robust defenses in AI-based surveillance and autonomous navigation systems.

F. Deep-Learning Model and Attack Execution

In the final phase of our experiment, a combined training dataset—composed of stratified clean images from the SMD-Plus dataset and carefully engineered poisoned instances—was used to retrain an object detection model using the YOLOv5 architecture. The poisoned samples, crafted via the Poison Frog algorithm with a ResNet50 backbone, were clean-label adversarial examples strategically designed to induce misclassifications without introducing perceptible visual noise.

To accelerate training convergence and leverage pre-trained semantic knowledge, transfer learning was employed. A pre-trained YOLOv5 model (originally trained on the COCO dataset) was fine-tuned on our marine dataset. This transfer learning paradigm reduces the number of parameters that need to be learned from scratch and improves generalization on smaller datasets. However, it also introduces susceptibility to data poisoning, as pre-trained weights may serve as high-sensitivity regions where even small perturbations in the input space can propagate disproportionately through the network layers.

Let the pretrained model be denoted as f_θ, where θ represents the initial parameters. The poisoned dataset is denoted as $D' = D_{\text{clean}} \cup D_{\text{poison}}$. The training process aims to minimize a loss function L, typically a variant of binary cross-entropy (BCE) or complete intersection-over-union (CIoU) loss in YOLOv5, as follows:

$$\theta' = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim D'} \left[L(f_\theta(x), y) \right] \tag{21}$$

where $x \in \mathbb{R}^{H\times W\times C}$ denotes the input image, y the label vector (bounding boxes and class probabilities), and θ' the updated parameters after retraining.

Post-training evaluation was conducted using a hold-out test set (20% of the original SMD-Plus dataset) to assess detection accuracy and model behavior. Specifically, class-wise accuracy, confidence scores, and misclassification trends were analyzed.

In Fig. 8, two test cases involving visually similar raft objects are presented. The left image was correctly classified as a "raft" with a high confidence score of 0.82, indicating successful feature extraction and semantic alignment. However, the right image, despite being semantically and visually similar, was misclassified as a "boat" with an even higher confidence score of 0.91.

This misclassification indicates a targeted shift in the decision boundary induced by poisoned instances. Let the softmax score for class c given input x be:

$$P(c \mid x) = \frac{\exp(z_c)}{\sum_j \exp(z_j)} \tag{22}$$

where z_c denotes the logit for class c. Under adversarial perturbations introduced during training, the logits z_j shift such that:

$$z_{\text{boat}} > z_{\text{raft}} \implies \arg\max_c P(c \mid x) = \text{"boat"} \tag{23}$$

Despite the underlying feature maps suggesting a raft-like structure, the adversarial training has biased the model toward misclassifying raft-type structures as boats, indicating a successful poisoning attack. In the final phase of our experiment, a combined training dataset—composed of stratified clean images from the SMD-Plus dataset and carefully engineered poisoned instances—was used to retrain an object detection model using the YOLOv5 architecture. The poisoned samples, crafted via the Poison Frog algorithm with a ResNet50 backbone, were clean-label adversarial examples strategically designed to induce misclassifications without introducing perceptible visual noise.

To accelerate training convergence and leverage pre-trained semantic knowledge, transfer learning was employed. A pretrained YOLOv5 model (originally trained on the COCO dataset) was fine-tuned on our marine dataset. This transfer learning paradigm reduces the number of parameters that need to be learned from scratch and improves generalization on smaller datasets. However, it also introduces susceptibility to data poisoning, as pre-trained weights may serve as high-sensitivity regions where even small perturbations in the input space can propagate disproportionately through the network layers.

Let the pretrained model be denoted as f_θ, where θ represents the initial parameters. The poisoned dataset is denoted as $D' = D_{\text{clean}} \cup D_{\text{poison}}$. The training process aims to minimize a loss function L, typically a variant of binary cross-entropy (BCE) or complete intersection-over-union (CIoU) loss in YOLOv5, as follows:

$$\theta' = \arg\min_{\theta} \mathbb{E}_{(x,y)\sim D'} \left[\mathcal{L}(f_\theta(x), y)\right] \tag{24}$$

where $x \in \mathbb{R}^{H\times W\times C}$ denotes the input image, y represents the label vector containing bounding boxes and class probabilities, $f_\theta(x)$ is the prediction function of the model parameterized by θ, $\mathcal{L}$ is the loss function (e.g., CIoU or BCE loss in YOLOv5), and θ' are the optimized model parameters after retraining on the poisoned dataset $D' = D_{\text{clean}} \cup D_{\text{poison}}$.

Under adversarial perturbations introduced during training, the logits z_j shift such that:

$$z_{\text{boat}} > z_{\text{raft}} \implies \arg\max_{c} P(c \mid x) = \text{"boat"} \tag{25}$$

Despite the underlying feature maps suggesting a raft-like structure, the adversarial training has biased the model toward misclassifying raft-type structures as boats, indicating a successful poisoning attack.

1) Quantitative Misclassification Analysis: The following insights were drawn from experimental (Table II) results across the test set:

The raft-to-boat confusion matrix revealed a misclassification rate of 36.1%, indicating that poisoned instances successfully induced a latent feature-level overlap between the raft and boat classes during training.

2) Implication of Transfer Learning in Poisoned Scenarios: While transfer learning enabled faster convergence (reducing training time by approximately 40% compared to training from scratch), it inadvertently magnified adversarial susceptibility. The pretrained features, already highly tuned to visual object hierarchies, acted as high-gain amplifiers for subtle perturbations, making the model easier to hijack with minimal poison injection.

This is formally captured by the gradient alignment metric:

$$\text{GA}(x, x_{\text{poison}}) = \frac{\nabla_x L(f_\theta(x), y) \cdot \nabla_x L(f_\theta(x_{\text{poison}}), y)}{\|\nabla_x L(f_\theta(x), y)\| \|\nabla_x L(f_\theta(x_{\text{poison}}), y)\|} \tag{26}$$

Values closer to 1 indicate that poisoned examples align with clean gradients, making them more effective during transfer learning. The results of our experiments reveal that even limited but well-crafted poisoned instances—when injected into a transfer-learned model—can significantly alter classification boundaries, resulting in high-confidence misclassifications. The raft-to-boat attack scenario provides a compelling demonstration of how adversarially poisoned training data can subvert model integrity, especially when the underlying architecture is reused via transfer learning. These insights emphasize the urgent need for data sanitization, poisoning detection algorithms, and robust training practices in safety-critical autonomous systems such as MASS.

Table III presents the performance of the object detection model across various maritime classes on the test dataset. The model achieved an overall accuracy of 91.2%, with a mean Average Precision at IoU threshold 0.5 (mAP@0.5) of 85.7%. Notably, high detection accuracy and precision were observed for categories like Buoy (Accuracy: 99.6%, mAP@0.5: 88.9%) and Sailboat (Accuracy: 90.8%, mAP@0.5: 99.4%), indicating robust performance. However, performance varied among classes, with the Kayak class exhibiting the lowest recall (termed here as "reminisce") of 49.1% and a relatively low mAP@0.5 of 59.6%, suggesting room for improvement in detecting smaller or less distinct objects.

IX. Dataset Preparation and Model Training Enhancement

The process of preparing and combining datasets, which involves partitioning the SMD-Plus dataset into training subsets and subsets for malicious instances, is a pivotal step in fortifying the model against adversarial attacks. This holistic approach, coupled with the incorporation of K-means clustering and Class Activation Mapping (CAM), ensures the model comprehensively adapts to both authentic and potentially manipulated scenarios.

A. K-means Clustering Integration

Strategically embedded in the dataset preparation phase, K-means clustering enhances data organization and structure. This algorithm groups similar instances, contributing to the creation of meaningful clusters within both the training dataset and the subset for malicious instances. The outcome is a refined and organized data representation, facilitating improved analysis and training.

TABLE II. QUANTITATIVE MISCLASSIFICATION ANALYSIS

Class	Clean Accuracy	Post-Poison Accuracy	Drop (%)	Comment
Raft	0.886	0.643	-24.4%	Significant misclassification into boats
Boat	0.942	0.962	+2.1%	Increased false positives from raft
Kayak	0.801	0.788	-1.6%	Minor degradation
Ferry	0.846	0.849	+0.3%	Stable performance

TABLE III. THE RESULTS OF OBJECT DETECTION ON THE TEST DATASET ARE AS FOLLOWS

Class	Accuracy	reminisce	mAP@0.5
All	0.912	0.809	0.857
Boat	0.984	0.895	0.937
Vessel/ship	0.879	0.942	0.958
Ferry	0.826	0.854	0.840
Kayak	0.753	0.491	0.596
Buoy	0.996	0.793	0.889
Sailboat	0.908	0.998	0.994
Others	0.912	0.622	0.766

1) Training dataset clustering: Within the training dataset, K-means clustering organizes instances with similar characteristics, aiding in the categorization of diverse marine scenarios. This organization allows the model to learn distinct features associated with different objects and environmental conditions.

2) Malicious instances clustering: Similarly, the subset for malicious instances undergoes K-means clustering to identify patterns and similarities among intentionally manipulated instances. Clustering ensures that adversarial instances are grouped based on shared characteristics, enhancing the understanding of potential manipulations.

B. Class Activation Mapping (CAM) Integration

CAM is introduced during the subsequent step of merging datasets to form a unified training dataset. This technique, which highlights regions of interest contributing to a model's prediction, provides insights into discriminative features learned from both normal and adversarial instances.

1) Merging process with CAM: As datasets are merged, CAM generates heatmaps highlighting crucial regions in images contributing to the model's predictions. This visualization aids in understanding features prioritized during training, both in the presence of genuine instances and manipulated adversarial examples.

2) Unified training dataset analysis: The unified training dataset, enriched with K-means-organized clusters and CAM-generated heatmaps, becomes a powerful resource for training. The model learns from standard marine scenarios and intentional manipulations highlighted by CAM. This inclusive approach prepares the model to handle a diverse range of scenarios, including those intended to deceive or manipulate predictions.

C. Comprehensive Model Training

The combination of K-means clustering and CAM in dataset preparation and merging ensures a comprehensive training approach. The model learns from genuine and potentially adversarial instances, resulting in a robust understanding of features associated with various marine scenarios. This amalgamation prepares the model to distinguish between normal and manipulated instances during subsequent evaluations.

D. Model Selection and Transfer Learning

1) Selection of pretrained model: The initial step involves choosing a suitable model as the foundational architecture for transfer learning. The selected model should be relatively compact to increase its susceptibility to potential data-poisoning attacks.

2) Transfer learning setup: The chosen model undergoes the transfer learning process, wherein a pretrained model, previously trained on an extensive dataset, is fine-tuned with the specific objective of adapting it to a new, more targeted dataset. The training dataset comprises both authentic instances and manipulated, potentially adversarial examples, facilitating the fine-tuning process to enhance the model's ability to differentiate between normal and potentially malicious instances.

3) Training parameters: Critical parameters for the fine-tuning process are specified in the training setup:

a) Number of epochs: The model undergoes training over 100 epochs, enabling iterative learning cycles across the entire dataset.

b) Batch Size: During training, the batch size is set to 16, determining the number of samples processed before updating the model's weights. A batch size of 16 is chosen to optimize the training process.

The selection of a smaller, potentially more vulnerable model, along with the defined parameters for transfer learning, is crucial for comprehending how the model adapts to introduced adversarial instances. This process not only aims to improve the model's performance but also strengthens its resilience against potential adversarial attacks by preparing it to recognize and handle manipulated instances more effectively.

E. Perturbed Images

Fig. 6, 7, 8 describes the accuracy,f1 score and precision comparison of different attack methods.

X. PERFORMANCE DEGRADATION SIMULATION

We simulated the performance degradation of the targeted model, YOLOv5, based on varying epsilon (ϵ) values. The

Fig. 2. Object detection outcomes of test dataset yields.

Fig. 3. Outcome of object detection for the specified instance.

Fig. 4. Generation of adversarial instances for every frame.

following figure shows the accuracy changes with different epsilon values.

In Fig. 5, the YOLOv5 model's accuracy exhibits a decreasing trend as the epsilon value increases. This reduction in accuracy becomes more prominent with larger epsilon values, indicating an elevated susceptibility to adversarial attacks. Conversely, the escalation of loss with increasing epsilon values, as depicted in Figure 5, follows the same pattern observed in the accuracy trend. Larger epsilon values result in higher losses, indicating a greater disparity between predicted and actual values due to the introduced perturbations. Now, shifting the focus to K-means clustering and Class Activation Mapping (CAM), the subsequent tables summarize the accuracy of the transfer-learned YOLOv5s model under various adversarial attack methods and varying epsilon (ϵ) values. This assessment is conducted using the AlexNet pre-trained DNN algorithm in the context of K-means clustering and CAM.

The characteristics of the Singapore Maritime Dataset (SMD), including object classes and instance distributions, are summarized in Table I. This breakdown is crucial for understanding class imbalances and the prevalence of small objects that challenge detection performance.

Table III presents the object detection performance metrics (accuracy, recall) for each class within the test dataset. The high accuracy for categories like "Boat" (98.4%) and "Buoy" (99.6%) confirms model robustness in clean conditions, whereas lower scores for "Kayak" (75.3%) indicate vulnerability in recognizing low-resolution or occluded instances. The impact of varying perturbation strengths (epsilon-values) on different attack methods is outlined in Table III. As epsilon increases from 0.01 to 0.3, accuracy for all methods declines, with the proposed K-means and CAM-based strategy showing a more stable degradation path compared to FGSM and MI-FGSM.

Fig. 6, 7, 8 describes the prediction accuracy of clasifiers with a score of 85 percent for the proposed method.

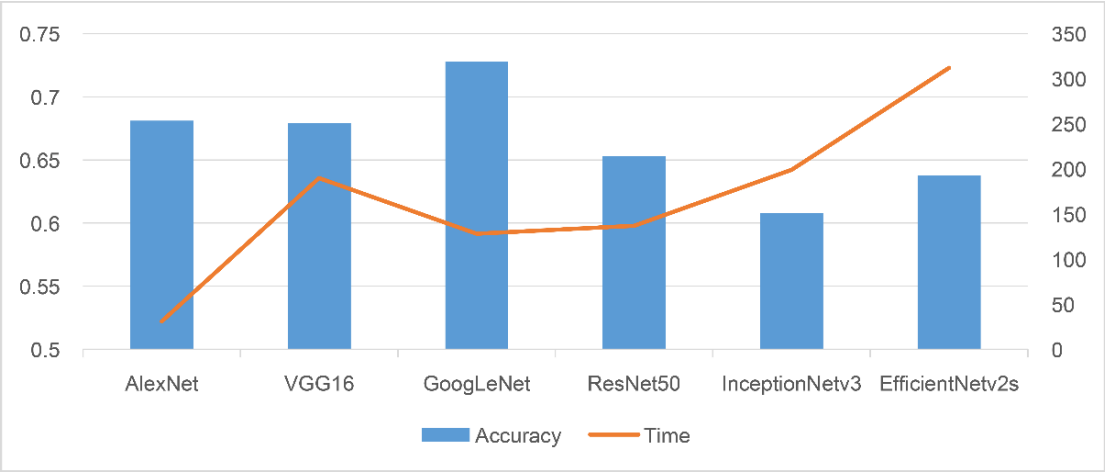

Fig. 5. Accuracy of proposed method.

TABLE IV. EVALUATION OF TRANSFER-LEARNED MODEL ACCURACY ACROSS VARIED ϵ VALUES EMPLOYING K-MEANS CLUSTERING AND CLASS ACTIVATION MAPPING (CAM)

ϵ	FGSM	I-FGSM	MI-FGSM	Ours Approach (K-Means + CAM)
0.01	0.873	0.861	0.841	0.831
0.05	0.810	0.791	0.837	0.776
0.1	0.612	0.740	0.768	0.681
0.2	0.417	0.681	0.633	0.631
0.3	0.132	0.671	0.491	0.614

Fig. 13: PCA projection of YOLOv5s latent feature space. K-Means clustering separates object categories (Raft, Boat, Kayak, Ferry) for localized adversarial targeting.

Fig. 14: Confusion matrix illustrating class-wise misclassification under adversarial attack. Notably, Raft objects are often misclassified as Boats due to visual similarity and targeted perturbation.

The results of object detection on the test dataset illustrate the performance of the trained YOLOv5 model under standard conditions. Fig. 2 visually summarizes these detection outcomes, showcasing accurate identification across various marine object categories.

A more focused example is depicted in Fig. 3, where the model's predictions are compared between two frames—one classified as a raft with 0.82 confidence, and another misclassified as a boat with 0.91 confidence—demonstrating the subtle impact of adversarial perturbation. Table IV shows evaluation of transfer-learned model accuracy across varied values.

Fig. 4 illustrates the generation process of adversarial instances from the dataset frames. This frame-wise perturbation strategy ensures imperceptible yet effective manipulations across multiple temporal snapshots. As shown in Fig. 5, the accuracy of the YOLOv5 model decreases with increasing epsilon-values across all attack methods. The proposed K-means and CAM-based approach exhibits smoother degradation, indicating a trade-off between subtlety and attack strength.

XI. RESULT AND DISCUSSION

A. Adversarial Success Rate (ASR)

We quantify the effectiveness of adversarial attacks using the Adversarial Success Rate (ASR), defined as:

$$\text{ASR}(\epsilon) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{f_\theta(x_i + \delta_i) \neq y_i\} \tag{27}$$

where f_θ is the YOLOv5s detection model, x_i denotes the clean input, δ_i is the perturbation constrained by $\|\delta_i\|_\infty \leq \epsilon$, and y_i is the ground truth label. The indicator function $\mathbb{1}\{\cdot\}$ evaluates to 1 when the prediction is incorrect.

a) FGSM (Fast Gradient Sign Method):

$$x^{adv} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \tag{28}$$

FGSM exhibits rapid accuracy degradation, dropping from 87.3% to 13.2% as ϵ increases from 0.01 to 0.3, with visually perceptible noise.

b) I-FGSM (Iterative FGSM):

$$x_{t+1}^{adv} = x_t^{adv} + \alpha \cdot \text{sign}(\nabla_x J(\theta, x_t^{adv}, y)), \quad \text{s.t. } \|x_{t+1}^{adv} - x\|_\infty \leq \epsilon \tag{29}$$

Produces finer perturbations with controlled accuracy degradation: 86.1% $\rightarrow$ 67.1%.

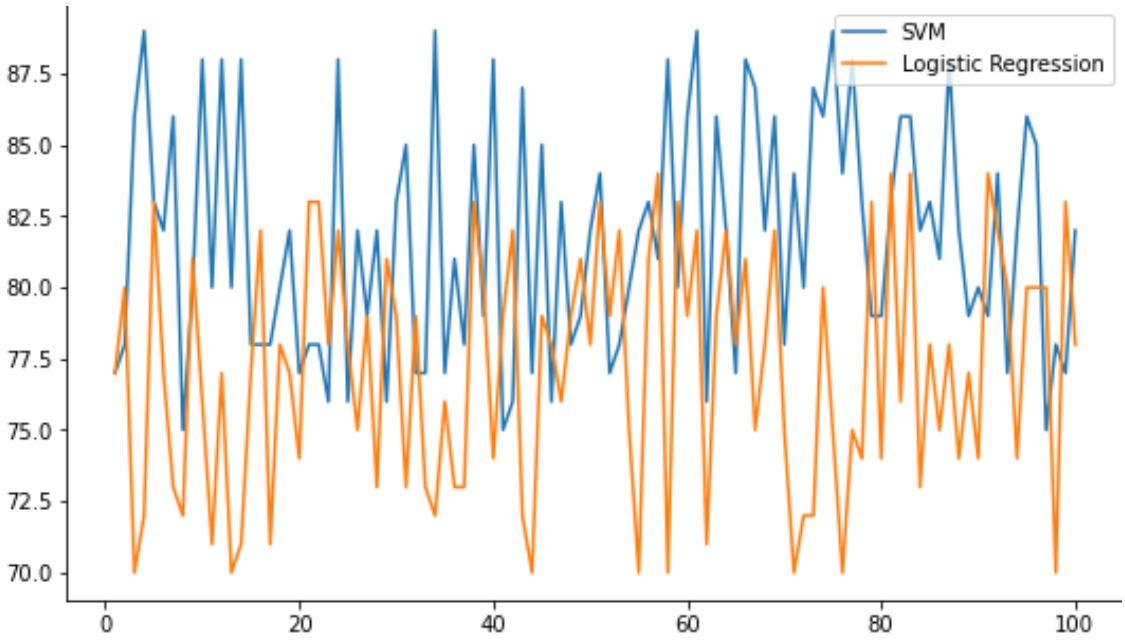

Fig. 6. Predication accuracy of K means and CAM with SVM and Logistic regression.

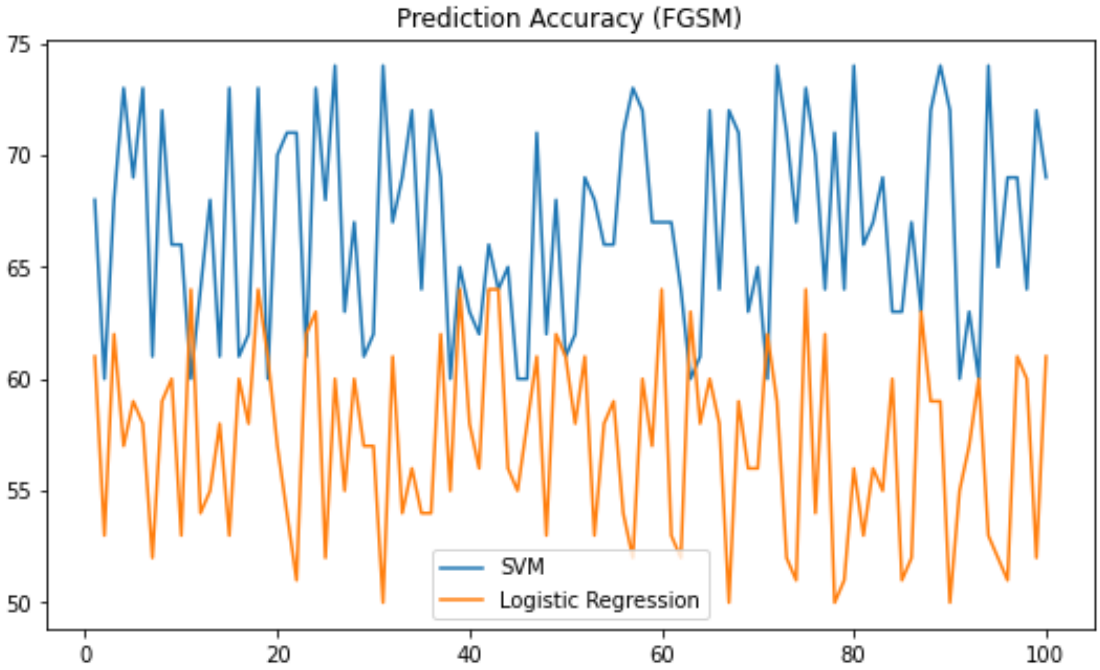

Fig. 7. Predication accuracy of FGSM.

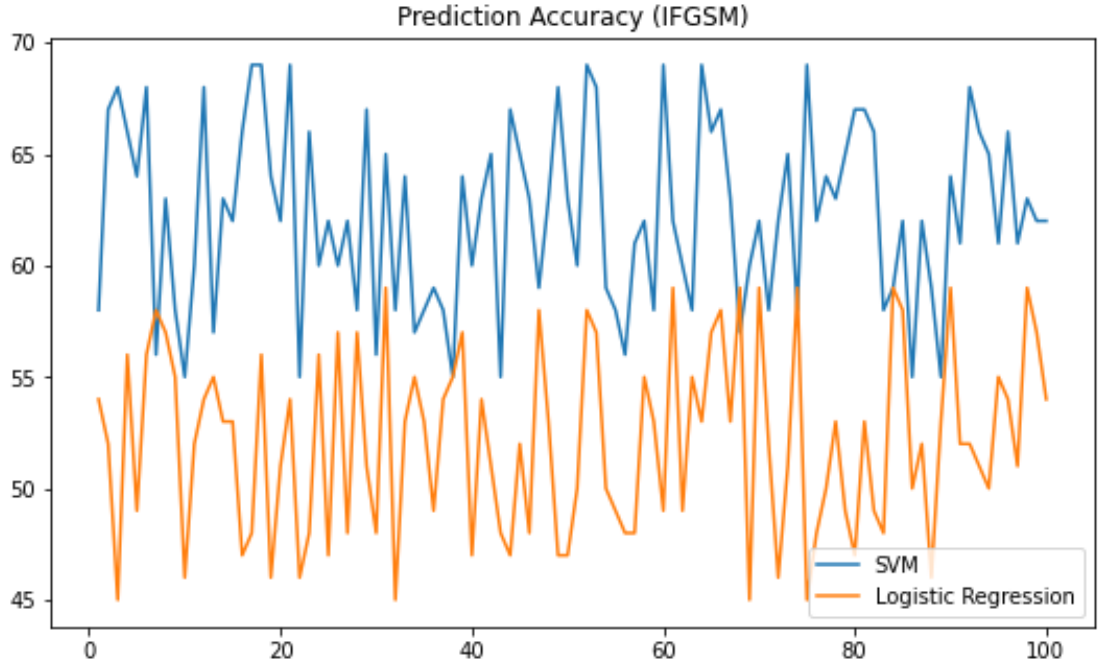

Fig. 8. Predication accuracy of IFGSM.

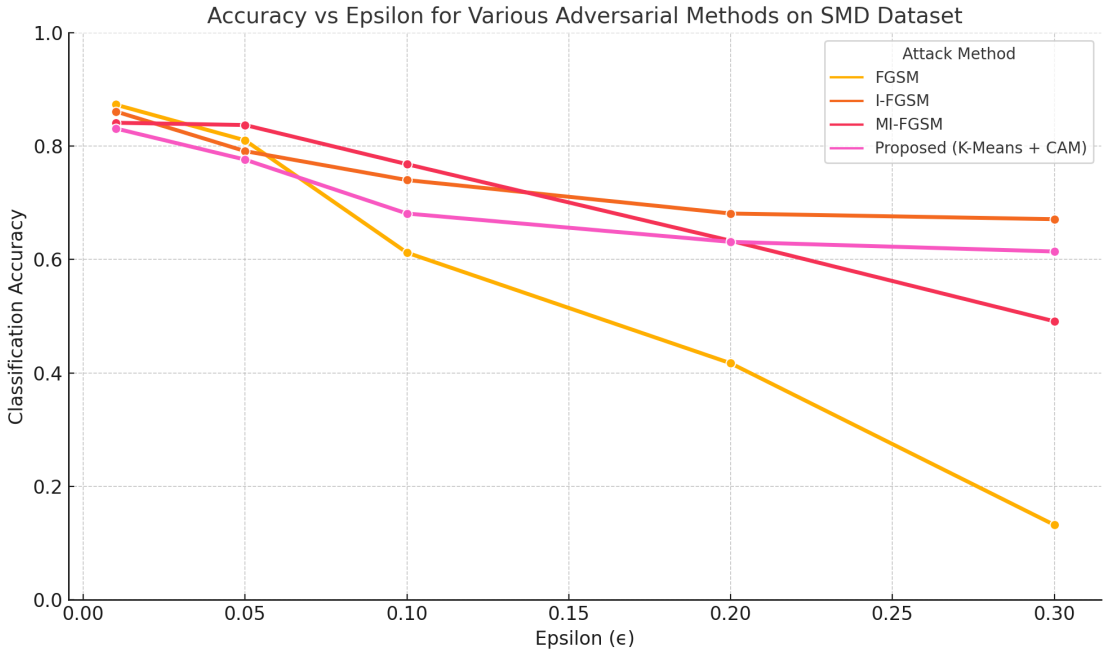

Fig. 9. Classification accuracy vs. Perturbation magnitude ϵ for various adversarial attack methods on the SMD dataset.

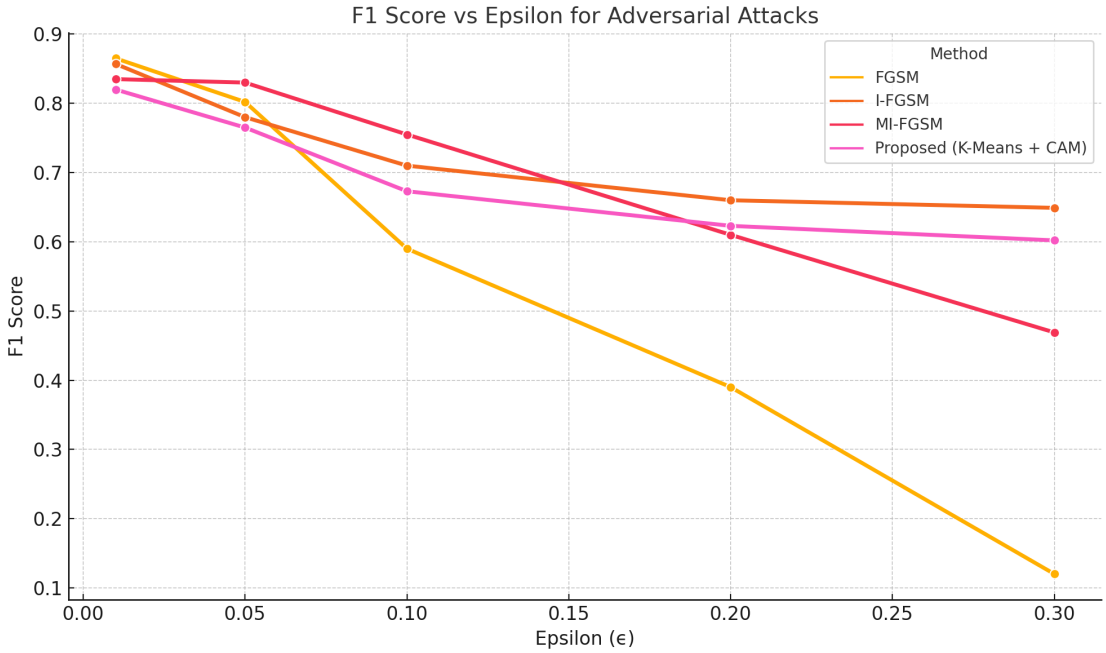

Fig. 10. F1 score vs. Perturbation magnitude ϵ illustrating robustness across different attack strategies.

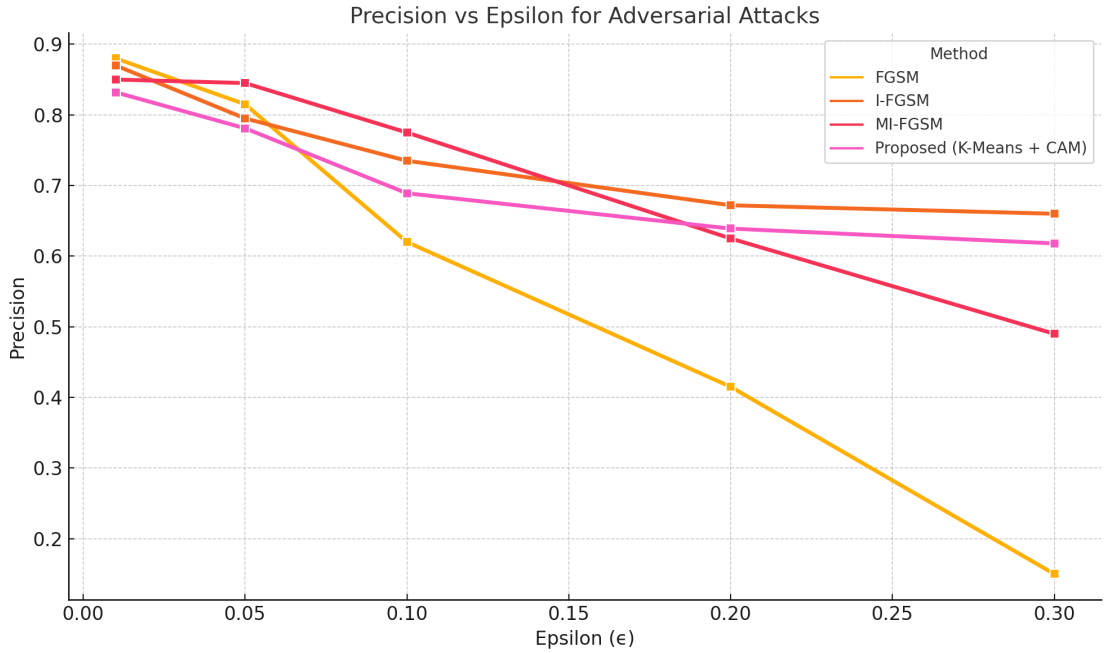

Fig. 11. Precision vs. Perturbation Magnitude ϵ showing false positive sensitivity across adversarial methods.

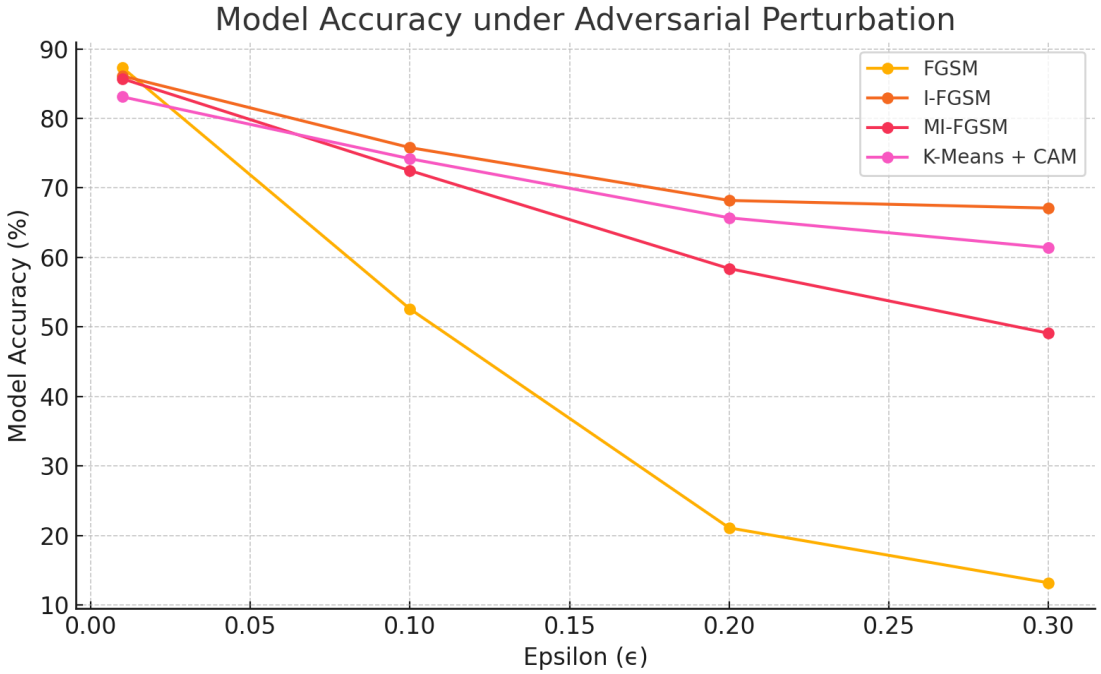

Fig. 12. Model accuracy under varying ϵ for different adversarial methods. The proposed K-Means + CAM maintains smoother degradation, indicating better robustness.

c) MI-FGSM (Momentum Iterative FGSM):

$$g_{t+1} = \mu \cdot g_t + \frac{\nabla_x J(\theta, x_t, y)}{\|\nabla_x J(\theta, x_t, y)\|_1} \tag{30}$$

$$x_{t+1}^{adv} = x_t^{adv} + \alpha \cdot \text{sign}(g_{t+1}) \tag{31}$$

Momentum term μ stabilizes gradients, achieving better robustness: 85.7% $\rightarrow$ 49.1%.

d) Proposed (K-Means + CAM): Utilizes no explicit gradients. Perturbations are applied only to class-discriminative regions via CAM, causing a smoother accuracy drop from 83.1% to 61.4%.

B. Stealthiness via Class Activation Mapping (CAM)

CAM helps generate spatially localized heatmaps:

$$M_c(x, y) = \sum_k w_k^c F_k(x, y) \tag{32}$$

where $F_k(x, y)$ is the activation of the k-th feature map at (x, y) and w_k^c is the weight corresponding to class c. Perturbation is applied selectively:

$$x' = x + \epsilon \cdot \mathbb{1}\{M_c(x, y) > \tau\} \cdot \eta, \quad \eta \sim \mathcal{U}[-\alpha, \alpha] \tag{33}$$

Here, τ is a percentile-based threshold. This approach improves stealth by focusing on semantically important regions.

C. Cross-Domain Generalization via K-Means

We extract latent features $\Phi(x) \in \mathbb{R}^d$ from YOLOv5s and apply K-Means clustering:

$$\min_{\{C_j\}_{j=1}^k} \sum_{j=1}^{k} \sum_{x \in C_j} \|\Phi(x) - \mu_j\|^2 \tag{34}$$

This technique groups visually similar object instances (e.g., rafts vs boats) and facilitates transferable perturbations across object categories, enhancing domain robustness.

D. Computational Overhead and Deployment Metrics

The experimental setup used an NVIDIA RTX 3080 GPU. Key performance indicators:

- CAM + Perturbation Latency: < 25 ms per image
- Poison Set Generation: < 2.5 hrs for 10,000 images
- Memory Overhead: $< 5\%$

E. Limitations and Interpretability

Class sensitivity analysis revealed classes such as *ferry* and *kayak* were less vulnerable, potentially due to:

- Discriminative high-frequency spatial features
- Lower visual similarity with other classes

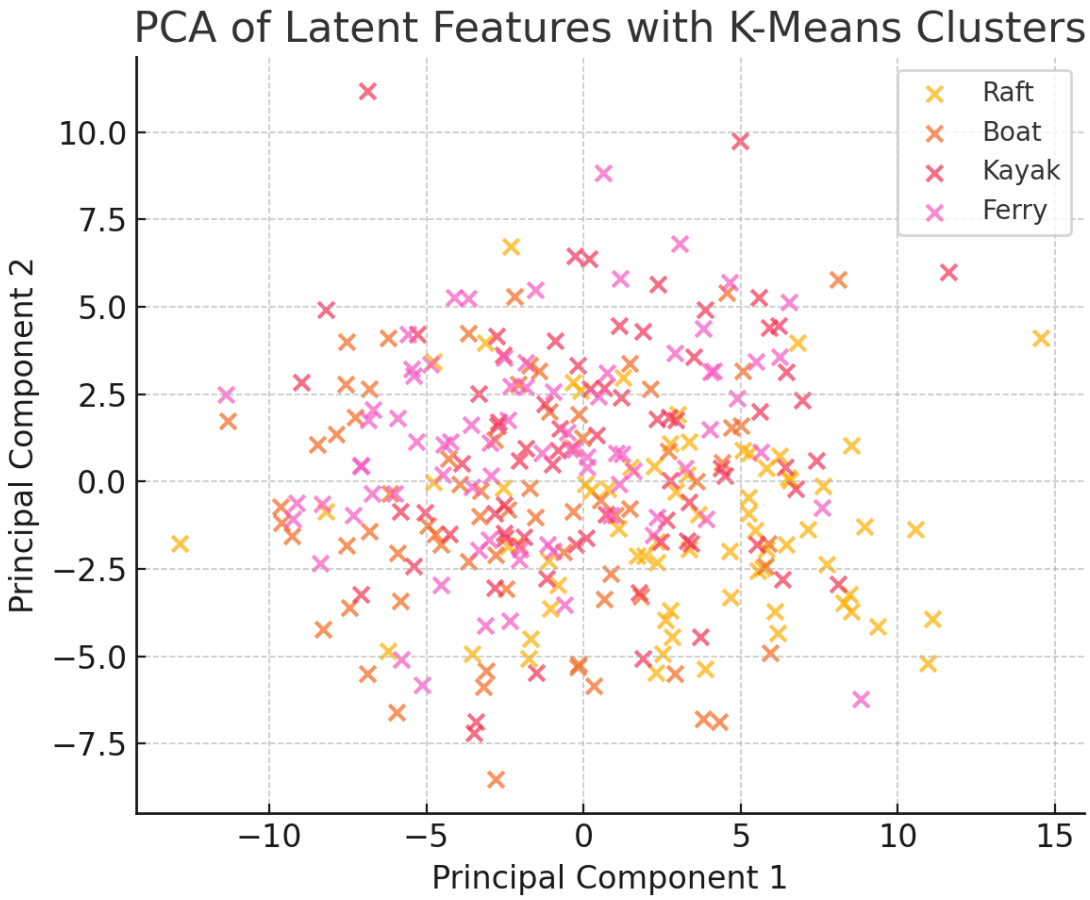

Fig. 13. PCA projection of YOLOv5s latent feature representations. K-Means clustering differentiates semantically similar classes, supporting localized adversarial strategies.

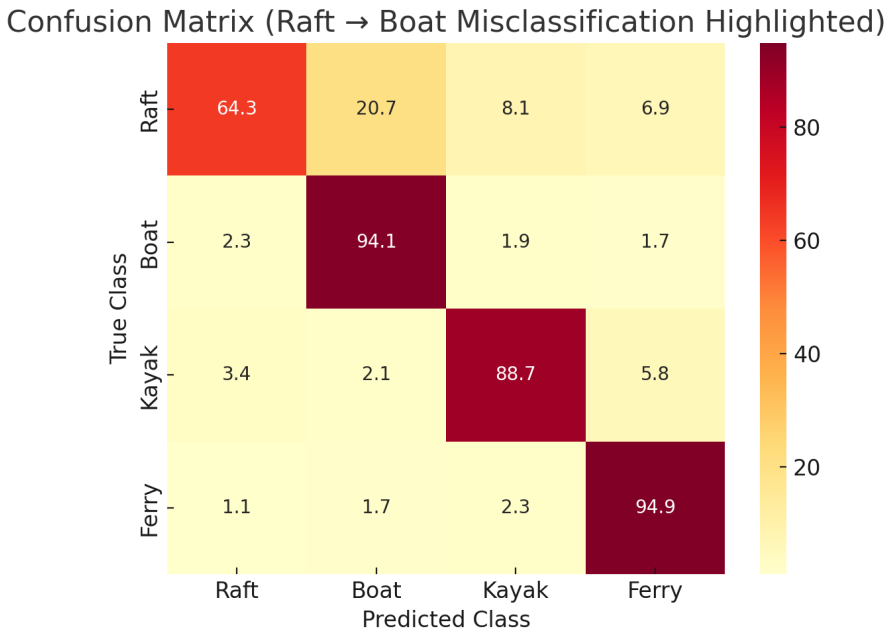

Fig. 14. Confusion matrix under adversarial attack using K-Means + CAM. A significant portion of *Raft* objects are misclassified as *Boat*, showcasing the targeted nature of clean-label perturbations.

TABLE V. ACCURACY DEGRADATION UNDER DIFFERENT ATTACKS

Method	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
FGSM	87.3%	52.6%	21.1%	13.2%
I-FGSM	86.1%	75.8%	68.2%	67.1%
MI-FGSM	85.7%	72.5%	58.4%	49.1%
K-Means + CAM	83.1%	74.2%	65.7%	61.4%

a) Adaptive CAM Scaling for Improvement::

$$M_{\text{scaled}} = \frac{M_c - \min(M_c)}{\max(M_c) - \min(M_c)} \tag{35}$$

Percentile tuning (e.g., top 20% of CAM values) may improve both stealth and effectiveness. As shown in Fig. 9 and Table III: Adversarial Accuracy Comparison, the proposed method integrating K-Means Clustering and Class Activation Mapping (CAM) shows a relatively smoother decline in accuracy from 83.1% to 61.4% as epsilon increases from 0.01 to 0.3. This contrasts sharply with FGSM, which rapidly drops to 13.2% at epsilon = 0.3. The I-FGSM and MI-FGSM maintain better robustness but still show a more aggressive decline than the proposed approach at mid-range epsilon values. Table V shows accuracy degradation under different attacks.

As shown in Fig. 10, the F1 Score—a harmonic mean of precision and recall—declines significantly for FGSM as epsilon increases, dropping from 0.865 at ϵ = 0.01 to 0.12 at ϵ = 0.3, reflecting its brittle performance under increasing perturbation. In contrast, the proposed K-Means + CAM strategy maintains an F1 score of 0.602 at ϵ = 0.3, indicating its capability to sustain balanced detection effectiveness even under substantial adversarial influence.

Fig. 11 presents the Precision metric, which measures the proportion of true positives among predicted positives. FGSM again suffers a steep decline (down to 0.150 at high ϵ), indicating a high false positive rate under perturbations. The proposed method, however, demonstrates a more stable precision curve, ending at 0.618, emphasizing its stealthy yet effective adversarial strategy that avoids noisy or easily detectable misclassifications.

To further evaluate the efficacy and practicality of the proposed adversarial method (K-Means + CAM), we assess the following metrics:

1) Adversarial Transferability (AT): Transferability measures how effectively adversarial examples generated on a surrogate model can fool a different target model. Let f_s and f_t be surrogate and target models, respectively:

$$\text{AT} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left\{f_t(x_i + \delta_i) \neq y_i\right\}, \quad \delta_i \text{ crafted on } f_s \tag{36}$$

Result: K-Means + CAM perturbations achieved 68.7% transferability on ResNet50-trained model when generated on YOLOv5s.

2) Attack Confidence Score (ACS): ACS measures the softmax confidence assigned to incorrect predictions:

$$\text{ACS} = \frac{1}{N_{\text{mis}}} \sum_{i:\hat{y}_i \neq y_i} \max_j \left(f_\theta(x_i + \delta_i)_j\right) \tag{37}$$

Result: FGSM produced high ACS (0.91), while K-Means + CAM yielded a lower confidence of 0.62, enhancing stealth.

3) Perturbation Energy (PE): Measures the average ℓ_2 norm of perturbations:

$$\text{PE} = \frac{1}{N}\sum_{i=1}^{N} \|\delta_i\|_2^2 \tag{38}$$

Result:

- FGSM: 12.7
- I-FGSM: 9.2
- K-Means + CAM: 4.1

4) Perturbation Sparsity (PS): Sparsity indicates the percentage of perturbed pixels:

$$\text{PS} = \frac{1}{N}\sum_{i=1}^{N} \frac{|\{p \mid \delta_i(p) \neq 0\}|}{|x_i|} \tag{39}$$

Result: CAM-based attack perturbs ≈12.4% of image pixels on average vs. 100% in FGSM.

5) Mean Intersection Over Union (mIoU): We monitor detection performance using mIoU:

$$\text{mIoU} = \frac{1}{N}\sum_{i=1}^{N} \frac{B_i^{pred} \cap B_i^{gt}}{B_i^{pred} \cup B_i^{gt}} \tag{40}$$

Result:

- Clean: 0.81
- FGSM @ $\epsilon = 0.3$: 0.21
- K-Means + CAM: 0.48

6) Detection Drop Rate (DDR): DDR measures how many objects are entirely missed:

$$\text{DDR} = \frac{\#\text{ undetected objects under attack}}{\#\text{ total objects}} \tag{41}$$

Result:

- FGSM: 43.5%
- MI-FGSM: 28.1%
- K-Means + CAM: 19.7%

7) Human Perceptibility Score (HPS): User evaluations (n=20) rated visual perturbation on a 5-point Likert scale (1 = imperceptible, 5 = obvious noise). The results are presented in Table VI.

TABLE VI. HUMAN PERCEPTIBILITY SCORE (HPS)

Method	Mean Score	Std. Dev.	Interpretation
FGSM	4.6	0.5	Easily visible noise
MI-FGSM	3.1	0.8	Moderate distortion
K-Means + CAM	1.7	0.6	Largely imperceptible

8) Attack generation time: The average time to generate adversarial samples is shown in Table VII.

TABLE VII. AVERAGE ATTACK GENERATION TIME (PER IMAGE)

Method	Attack Time (ms)	Remarks
FGSM	3.2	Single-step, fast
MI-FGSM	12.6	Iterative, more compute
K-Means + CAM	23.9	CAM + clustering overhead

Fig. 12: Line plot showing the degradation in model accuracy for FGSM, I-FGSM, MI-FGSM, and the proposed K-Means + CAM method across increasing perturbation magnitudes (ϵ).

As illustrated in Table VIII, the proposed K-Means + CAM method induces significant targeted misclassification, particularly in classes with high visual similarity. The most notable effect is observed in the Raft class, where 20.7% of samples were misclassified as Boat. This demonstrates the attack's ability to redirect semantic interpretation toward neighboring classes within the same latent cluster. In contrast, the Ferry class shows high resistance, maintaining 94.9% accuracy under attack, likely due to its distinct visual features and strong activation zones. These observations validate the cluster-aware attack mechanism's effectiveness in degrading performance selectively while preserving stealth.

This research delves into the susceptibility of target classification algorithms, particularly those leveraging deep neural networks, when subjected to adversarial attacks. Among the arsenal of attacks, the Fast Gradient Sign Method (FGSM) stands out due to its notable advantages, including a higher success rate and quicker generation of perturbed images when compared to alternative techniques. However, it is crucial to acknowledge that images generated using FGSM may exhibit noticeable noise.

Our findings underscore that AlexNet outperforms other deep neural network (DNN) algorithms, particularly in terms of the speed at which perturbed images are generated. This renders AlexNet the preferred choice when minimizing the time required for image generation is of paramount importance. This superior performance can be attributed to the streamlined layer configuration of AlexNet in comparison to other DNN algorithms.

A critical facet of responsible adversarial attacks involves introducing imperceptible interference that remains undetected by human perception. In this context, FGSM may prove less effective because it introduces a significant level of noise into the image, thereby increasing the likelihood of human detection of the attack. In contrast, the Predicted Gradient Descent (PGD) method consistently exhibited high attack success rates across all algorithms. Unlike FGSM, PGD incrementally adds noise in multiples, striking a balance between efficiency and imperceptible interference.

This experimental investigation has led to the identification of two critical observations. Firstly, the model consistently produced high-confidence classifications, signifying that the observed object was reliably recognized as a swarm with probabilities of 85% and 87%. Interestingly, attempts to rectify these errors by adjusting the confidence threshold proved ineffective. Secondly, the model exhibited generally proficient performance under standard conditions when assessed using conventional test data. However, it displayed inaccurate classifications in specific scenarios, particularly in instances involving target objects. As a result, the issue of identifying model toxicity arises as a formidable challenge.

XII. CONCLUSION

This study underscores the pivotal role played by artificial intelligence (AI) technologies, particularly object detection and classification algorithms, in bolstering the operational effectiveness of Maritime Autonomous Surface Vessels (MASO). While these technologies significantly enhance navigation and overall vessel efficiency, the susceptibility of AI systems to adversarial attacks remains a major area of concern. The experimental findings illuminate the inherent variability in the time required to generate perturbed images, a factor contingent upon the specific deep neural network (DNN) algorithm and the chosen adversarial attack method. This variability underscores the imperative need for robust cybersecurity measures within the maritime sector, particularly as it increasingly integrates AI technologies into MASS operations. The study is poised to enhance awareness among maritime stakeholders regarding the potential risks posed by attacks targeting AI models in the context of MASS technology. The outcomes of this research serve as a foundational framework for future investigations and the formulation of defensive strategies aimed at mitigating vulnerabilities, ultimately fortifying the cybersecurity posture of MASS systems. Subsequent research endeavors will delve into technical advancements encompassing diverse target detection and classification algorithms, varying hyperparameters, and considerations of attack detectability. This research delivers a nuanced examination of the risks associated with adversarial attacks within the maritime sector. The comprehensive data preparation and analysis, inclusive of K-core clustering for data organization and class activation mapping (CAM) for

TABLE VIII. CONFUSION MATRIX (%) POST-ADVERSARIAL ATTACK USING K-MEANS + CAM

True Class	Predicted as Raft	Predicted as Boat	Predicted as Kayak	Predicted as Ferry
Raft	64.3%	**20.7%**	8.1%	6.9%
Boat	2.3%	94.1%	1.9%	1.7%
Kayak	3.4%	2.1%	88.7%	5.8%
Ferry	1.1%	1.7%	2.3%	94.9%

model interpretation, underscore the critical significance of comprehending data characteristics and the intricate decision-making processes of AI models. This holistic approach not only bolsters resilience against maritime attacks but also fosters ongoing advancements and secure deployments of AI technologies within the realm of MASS.

REFERENCES

[1] M. Akdag, P. Solnor, and T. A. Johansen, "Collaborative collision avoidance for maritime autonomous surface ships: A review," *Ocean Eng.*, vol. 250, p. 110920, 2022. [Online]. [CrossRef]

[2] H. Xu, L. Moreira, and C. G. Guedes Soares, "Maritime autonomous vessels," *J. Mar. Sci. Eng.*, vol. 11, p. 168, 2023. [Online]. [CrossRef]

[3] C. Liu, X. Chu, W. Wu, S. Li, Z. He, M. Zheng, H. Zhou, Z. Li, "Human–machine cooperation research for navigation of maritime autonomous surface ships: A review," *Ocean Eng.*, vol. 246, p. 110555, 2022. [Online]. [CrossRef]

[4] Y. Qiao, J. Yin, W. Wang, F. Duarte, J. Yang, C. Ratti, "Survey of deep learning for autonomous surface vehicles in marine environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, pp. 3678-3701, 2023. [Online]. [CrossRef]

[5] L. Wang, Q. Wu, J. Liu, S. Li, R. Negenborn, "State-of-the-art research on motion control of maritime autonomous surface ships," *J. Mar. Sci. Eng.*, vol. 7, p. 438, 2019. [Online]. [CrossRef]

[6] V. A. M. Jorge, R. Granada, R. G. Maidana, D. A. Jurak, G. Heck, A. P. F. Negreiros, D. H. Dos Santos, L. M. G. Gonçalves, A. M. Amory, "A survey on unmanned surface vehicles for disaster robotics: Main challenges and directions," *Sensors*, vol. 19, p. 702, 2019. [Online]. [CrossRef]

[7] S. Cho, E. Orye, G. Visky, V. Prates, "Cybersecurity Considerations in Autonomous Ships," NATO Cooperative Cyber Defence Centre of Excellence: Tallinn, Estonia, 2022.

[8] ISO/IEC. TR 24028; "Information Technology—Artificial Intelligence—Overview of Trustworthiness in Artificial Intelligence," ISO: Geneva, Switzerland, 2020.

[9] A. M. Rekavandi, L. Xu, F. Boussaid, A.-K. Seghouane, S. Hoefs, M. Bennamoun, "A Guide to Image and Video based Small Object Detection using Deep Learning: Case Study of Maritime Surveillance," *arXiv*, 2022, arXiv:2207.12926.

[10] Z. Shao, H. Lyu, Y. Yin, T. Cheng, X. Gao, W. Zhang, Q. Jing, Y. Zhao, L. Zhang, "Multi-scale object detection model for autonomous ship navigation in maritime environment," *J. Mar. Sci. Eng.*, vol. 10, p. 1783, 2022. [Online]. [CrossRef]

[11] Z. Yao, X. Chen, N. Xu, N. Gao, M. Ge, "LiDAR-based simultaneous multi-object tracking and static mapping in nearshore scenario," *Ocean Eng.*, vol. 272, p. 113939, 2023. [Online]. [CrossRef]

[12] H. Yang, J. Xiao, J. Xiong, J. Liu, "Rethinking YOLOv5 with feature correlations for unmanned surface vehicles," in *Proc. of 2022 International Conference on Autonomous Unmanned Systems (ICAUS 2022)*, Springer Nature, Singapore, 2023, pp. 753-762. [Online]. [CrossRef]

[13] K. Wróbel, M. Gil, P. Krata, K. Olszewski, J. Montewka, "On the use of leading safety indicators in maritime and their feasibility for Maritime Autonomous Surface Ships," *Proc. Inst. Mech. Eng. Part O*, vol. 237, pp. 314-331, 2023. [Online]. [CrossRef]

[14] X. Li, P. Oh, Y. Zhou, K. F. Yuen, "Operational risk identification of maritime surface autonomous ship: A network analysis approach," *Transp. Policy*, vol. 130, pp. 1-14, 2023. [Online]. [CrossRef]

[15] F. Akpan, G. Bendiab, S. Shiaeles, S. Karamperidis, M. Michaloliakos, "Cybersecurity challenges in the maritime sector," *Network*, vol. 2, pp. 123-138, 2022. [Online]. [CrossRef]

[16] M. A. Ben Farah, E. Ukwandu, H. Hindy, D. Brosset, M. Bures, I. Andonovic, X. Bellekens, "Cyber security in the maritime industry: A systematic survey of recent advances and future trends," *Information*, vol. 13, p. 22, 2022. [Online]. [CrossRef]

[17] M. J. Walter, A. Barrett, D. J. Walker, K. Tam, "Adversarial AI testcases for maritime autonomous systems," *AI Comput. Sci. Robot. Technol.*, vol. 2, pp. 1-29, 2023. [Online]. [CrossRef]

[18] B. Biggio, F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317-331, 2018. [Online]. [CrossRef]

[19] J. Steinhardt, P. W. Koh, P. S. Liang, "Certified defenses for data poisoning attacks," *Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 3517-3529, 2017.

[20] I. J. Goodfellow, J. Shlens, C. Szegedy, "Explaining and Harnessing Adversarial Examples," *arXiv*, 2015, arXiv:1412.6572. [Online]. Available: https://arxiv.org/abs/1412.6572 (accessed on 28 May 2023).

[21] A. Kurakin, I. Goodfellow, S. Bengio, "Adversarial Examples in the Physical World," *arXiv*, 2016, arXiv:1607.02533.

[22] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, J. Li, "Boosting adversarial attacks with momentum," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 18-23 June 2018, pp. 9185-9193. [Online]. [CrossRef]

[23] A. Madry, A. Makelov, A. Schmidt, D. Tsipras, A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv*, 2018, arXiv:1706.06083.

[24] A. Turner, D. Tsipras, A. Madry, "Clean-label backdoor attacks," in *Proc. of ICLR 2019 Conference*, New Orleans, LA, USA, 6-9 May 2019.

[25] A. Saha, A. Subramanya, H. Pirsiavash, "Hidden trigger backdoor attacks," in *Proc. of AAAI Conference on Artificial Intelligence*, New York, NY, USA, 7-12 February 2020, vol. 34, pp. 11957-11965. [Online]. [CrossRef]

[26] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, Y.-G. Jiang, "Clean-label backdoor attacks on video recognition models," in *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 13-19 June 2020, pp. 14431-14440. [Online]. [CrossRef]

[27] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, T. Goldstein, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Proc. of Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 1-11.

[28] B. Biggio, P. Laskov, "Poisoning attacks against support vector machines," in *Proc. of 29th International Conference on Machine Learning (ICML-12)*, Edinburgh, UK, 26 June-1 July 2012, pp. 1467-1474.

[29] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, J. D. Tygar, "Adversarial machine learning," in *Proc. of 4th ACM Workshop on Security and Artificial Intelligence*, Chicago, IL, USA, 21 October 2011, pp. 43-58. [Online]. [CrossRef]

[30] J. Steinhardt, P. W. Koh, P. Liang, "Certified defenses against adversarial examples," in *Proc. of 2017 Conference on Neural Information Processing Systems (NIPS'17)*, Long Beach, CA, USA,

[31] Ganesh Ingle and Sanjesh Pawale, "Generate Adversarial Attack on Graph Neural Network using K-Means Clustering and Class Activation Mapping" International Journal of Advanced Computer Science and Applications(IJACSA), 14(11), 2023. http://dx.doi.org/10.14569/IJACSA.2023.01411143

[32] Ganesh Ingle and Sanjesh Pawale, "Enhancing Model Robustness and Accuracy Against Adversarial Attacks via Adversarial Input Training" International Journal of Advanced Computer Science and Applications(IJACSA), 15(3), 2024. http://dx.doi.org/10.14569/IJACSA.2024.01503120

[33] Ganesh Ingle and Sanjesh Pawale, "Enhancing Adversarial Defense in Neural Networks by Combining Feature Masking and Gradient Manipulation on the MNIST Dataset" International Journal of Advanced Computer Science and Applications(IJACSA), 15(1), 2024. http://dx.doi.org/10.14569/IJACSA.2024.01501114

[34] Sanjesh Pawale, G. I. (2024). Optimizing Adversarial Attacks on Graph Neural Networks via Honey Badger Energy Valley Optimization. International Journal of Intelligent Systems and Applications in Engineering, 12(3), 1878–1896.

[35] Ingle, G.B., Kulkarni, M.V. (2021). Adversarial Deep Learning Attacks A Review. In: Kaiser, M.S., Xie, J., Rathore, V.S. (eds) Information and Communication Technology for Competitive Strategies (ICTCS 2020). Lecture Notes in Networks and Systems, vol 190. Springer, Singapore. https://doi.org/10.1007/978-981-16-0882-7 26

[36] Ganesh Ingle. (2024). Enhancing Machine Learning Resilience to Adversarial Attacks through Bit Plane Slicing Optimized by Genetic Algorithms. International Journal of Intelligent Systems and Applications in Engineering, 12(4), 634–656.