Efficient and Adaptive Object Detection Systems for Maritime Mobile Environments

Mingkang Chen

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences

The Graduate University for Advanced Studies, SOKENDAI

March 2025

A dissertation submitted to Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies, SOKENDAI,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee

1. Prof. Kento Aida	National Institute of Informatics, SOKENDAI
2. Prof. Atsuko Takefusa	National Institute of Informatics, SOKENDAI
3. Prof. Michihiro Koibuchi	National Institute of Informatics
4. Prof. Takashi Kurimoto	National Institute of Informatics
5. Prof. Yusheng Ji	National Institute of Informatics
6. Prof. Kohta Ohshima	Tokyo University of Marine Science and Technology

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Kento Aida, of the Information Systems Architecture Research Division at the National Institute of Informatics and the Department of Informatics at the Graduate University for Advanced Studies, SOKENDAI. His invaluable mentoring, unwavering support, and the liberty he granted me to explore and grow as a researcher have been instrumental in my academic journey. His guidance and encouragement, as well as the opportunities he provided for pursuing my research interests and collaborating with leading researchers, have profoundly shaped my professional and personal development.

I am also deeply indebted to Dr. Jingtao Sun, currently a senior researcher at Hitachi, Ltd., who first introduced me to the academic path and inspired me to pursue this challenging yet fulfilling road. His belief in my potential has been pivotal to my journey, and his support, both professionally and in my daily life, has been a source of immense strength and motivation.

I would like to extend my heartfelt thanks to my advisers: Professors Atsuko Takefusa, Michihiro Koibuchi, Takashi Kurimoto, and Yusheng Ji from the National Institute of Informatics; Professor Kohta Ohshima from Tokyo University of Marine Science and Technology; and Professor Calton Pu from the Georgia Institute of Technology. Their insightful comments and constructive suggestions have greatly enriched my research and helped shape this dissertation. Furthermore, I am profoundly grateful for the support of the SOKENDAI Special Researcher (Field-Specific Type) project, which sustained my research efforts over the past three years and enabled me to focus fully on my academic pursuits.

My deepest gratitude goes to my parents, Zhongcai Chen and Youxi Lin, for their unconditional love and unwavering belief in me. Their support has been the foundation of my journey. I am also sincerely thankful to my dear friends Dasung Yoon, Shota Yajima, Dr. Ryo Yoshimura, and my fellow SOKENDAI student Chenlin Hang for their constant encouragement and support. Additionally, I would like to acknowledge my collaborators, Dr. Kazushige Saga, Dr. Kazuichi Oe from the National Institute of Informatics, and Dr. Tomoya Tanjo from the National Institute of Genetics, for their valuable suggestions and feedback, which significantly contributed to my research. To everyone who has supported me throughout this journey, directly or indirectly, thank you. This accomplishment would not have been possible without your guidance, encouragement, and belief in me.

THE GRADUATE UNIVERSITY FOR ADVANCED STUDIES, SOKENDAI

Abstract

School of Multidisciplinary Sciences
Department of Informatics

Doctor of Philosophy

Efficient and Adaptive Object Detection Systems for Maritime Mobile Environments

by Mingkang CHEN

As the demand for maritime surveillance and autonomous navigation grows, robust object detection systems capable of operating in challenging maritime environments are increasingly essential. Unlike land-based systems, maritime detection systems need to contend with unique challenges such as variable weather, unstable communications, and dynamic operational requirements. These systems must deliver adaptability, low latency, and consistent results to ensure reliability across diverse and evolving maritime conditions, including degraded performance during adverse weather or shifts in priorities from accuracy to computational efficiency.

This dissertation addresses these challenges by introducing an adaptive optimization method for maritime mobile object detection systems. It incorporates weather awareness and lifecycle management to enhance detection performance and efficiency. The proposed approach includes a weather-aware specialization method that deploys models tailored to specific conditions, improving performance through targeted training and deployment. A consistency evaluation method for time-series image data from mobile devices ensures reliable detection during monitoring. A dynamic model selection and retraining mechanism with adaptive triggers and urgency levels enables proactive prevention of performance degradation. Multiple optimization strategies prioritizing accuracy, computational efficiency, or retraining time allows systems to adapt dynamically to changing environments and user requirements.

The results demonstrate significant improvements in detection performance and operational efficiency, with specialized models outperforming traditional approaches. Experiments conducted in both simulated and real-world maritime environments validate the feasibility and effectiveness of the proposed methods, contributing to advancements in maritime surveillance and supporting the realization of autonomous surface ships.

Contents

List of Figures

List of Tables

Abbreviations

AI	**A**rtificial **I**ntelligence
ML	**M**achine **L**earning
DL	**D**eep **L**earning
CV	**C**omputer **V**ision
LiDAR	**Li**ght **D**etection **A**nd **R**anging
CNN	**C**onvolutional **N**eural **N**etworks
IoT	**I**nternet **o**f **T**hings
GPU	**G**raphics **P**rocessing **U**nits
ML	**M**achine **L**earning
R-CNN	**R**egion-based **C**onvolutional **N**eural **N**etworks
VGG	**V**isual **G**eometry **G**roup
MCA	**M**orphological **C**omponent **A**nalysis
SVM	**S**upport **V**ector **M**achines
GAN	**G**enerative **A**dversarial **N**etwork
SSD	**S**ingle **S**hot multibox **D**etector
YOLO	**Y**ou **O**nly **L**ook **O**nce
DETR	**DE**tection **TR**ansformer
DRL	**D**eep **R**einforcement **L**earning
PReNet	**P**rogressive **ResNet**
DDN	**D**eep **D**eraining **N**etwork
GP-based SSL	**G**aussian **P**rocess-based **S**emi-**S**upervised **L**earning
LPNet	**L**ightweight **P**yramid **Net**work
MSE	**M**ean **S**quared **E**rror
PSNR	**P**eak **S**ignal **N**oise **R**atio
SSIM	**S**tructural **SIM**ilarity index

BRISQUE	**B**lind/**R**eferenceless **I**mage **S**patial **Q**uality **E**valuator
AGGD	**A**symmetric **G**eneralized **G**aussian **D**istributions
SVR	**S**upport **V**ector **R**egression
DMOS	**D**ifferential **M**ean **O**pinion **S**core
SSEQ	**S**tructural **S**imilarity **E**valuation with a **Q**uality factor
NIQE	**N**atural **I**mage **Q**uality **E**valuator
UAV	**U**nmanned **A**erial **V**ehicle
COCO	**C**ommon **O**bjects in **CO**ntext
SMD	**S**ingapore **M**aritime **D**ataset
DS	**D**atset**S**hips
mAP	**m**ean **A**verage **P**recision
IoU	**I**ntersection **o**ver **U**nion
MILP	**M**ixed **I**nteger **L**inear **P**rogramming
ANOVA	**AN**alysis **o**f **VA**riance
Cara	**C**ost-**a**ware **r**etraining **a**lgorithm
OFV	**O**bjective **F**unction **V**alue
TSR	**T**hreshold **S**atisfaction **R**atio
PTR	**P**erformance-**T**hreshold **R**atio
MASS	**M**aritime **A**utonomous **S**urface **S**hips
RSE	**R**egulatory **S**coping **E**xercise
IMO	**I**nternational **M**aritime **O**rganization

Chapter 1

Introduction

1.1 Background

Object detection researches utilizing machine learning play crucial roles in application domains using mobile devices such as environmental surveillance. In recent years, developments in artificial intelligence (AI) such as deep learning (DL) have accelerated the capabilities of maritime surveillance, enabling more precise detection, and automated decision-making in vast ocean regions. With rising concerns about maritime security, environmental threats, and the need for efficient logistics, the adoption of object detection system methods within maritime surveillance systems is becoming indispensable. By enhancing the accuracy and reliability of object detection in challenging sea environments, maritime object detection systems can significantly improve operational safety, ultimately contributing to safer, more efficient global maritime operations. According to a recent market report [1], the global maritime surveillance market was valued at approximately USD 480.4 million in 2023, expected to register a compound annual growth rate of 8.6% from 2023 to 2031. The main targets of maritime surveillance systems are to monitor and secure vast ocean spaces, avoid collisions, assist in search-and-rescue operations, and detect illegal activities. Especially, ship crashes remain a major issue in the maritime industry, with far-reaching effects. The European Maritime Safety Agency's

Annual Overview of Marine Casualties and Occurrences in 2023 [2] reports that roughly 4,000 safety-related occurrences occur each year. Introducing DL-based object detection into maritime surveillance and navigation systems is becoming increasingly prevalent in the shipping industry. Current common AI technologies in maritime surveillance systems include Computer Vision (CV), Radar, and Light Detection And Ranging (LiDAR) techniques. They depend on different physical sensors, such as optical cameras, radars, and LiDARs, to detect and track objects in the maritime environment.

Maritime operations often encounter adverse weather conditions such as heavy rain, haze, and high winds, which significantly impair the performance of traditional navigational aids. Object detection on data from optical cameras is widely used in surveillance systems, as it can provide visual information directly perceived through the senses, enhancing situational awareness in maritime environments. There is plenty of development and implementation of object detection systems using optical cameras in maritime surveillance systems from on land to at sea. Besides, radar, a traditional maritime sensor, is widely used for detecting targets in the maritime environment [3] [4] [5]. Several studies have introduced convolutional neural networks (CNN) to radar target detection systems [6] [7] for assisting in maritime surveillance. Maritime radar systems, predominantly operating in the X-band or S-band, play a vital role in vessel detection and navigation under diverse conditions. However, these systems encounter limitations when faced with challenging weather and sea states. X-band radars, which transmit at approximately 9 GHz, provide high-resolution data suitable for short- to medium-range detection. They are prone to signal attenuation in rain, snow, and dense haze, which often results in reduced detection ranges and higher false alarm rates due to increased noise levels. Meanwhile, S-band radars offer longer-range coverage with greater resilience against atmospheric attenuation, yet their larger antennas and lower resolution can hinder target differentiation at closer distances. Both radar types also struggle with strong sea clutter in rough waters, where reflections from waves and spray can mask legitimate targets. Similarly, LiDAR, which relies on laser-based sensing to detect objects, is widely used in autonomous vehicle applications. Also, several studies have introduced LiDAR-based object detection systems in maritime surveillance systems [8] [9]. They suffer from reduced accuracy and range in rainy or hazy conditions, where water droplets scatter and absorb the laser beams. These limitations necessitate the adoption of robust technologies, such as AI-enhanced systems, which can integrate multiple data sources,

compensate for the weaknesses of individual sensors, and provide reliable obstacle detection and collision avoidance under diverse environmental conditions. By overcoming these deficiencies, AI-based object detection systems with optical cameras can better enhance maritime surveillance, and improve navigational safety.

In object detection in mobile devices, there is a growing demand for improved inference accuracy directly correlated with safety and reliability. The object detection system as the subject of this research, is a system that securely stores, processes, and analyzes still or video data captured by Internet of Things (IoT) sensor cameras on edge servers or cloud servers. On this system, object detection applications are beginning to operate in the real world to improve services for users and to support business improvement and efficiency by realizing object class, object location, and other visualization using computer vision technology based on mathematics and machine learning that gives computers human-like visual capabilities. Object detection systems are specialized AI-based frameworks designed to identify and locate objects within images or video frames from optical cameras. In applications such as surveillance, traffic monitoring, and autonomous navigation, object detection systems play a critical role in recognizing objects of interest in real-time. By processing visual data, these systems can identify patterns, track movements, and detect objects ranging from vehicles and pedestrians to smaller, specialized targets like maritime vessels or navigation markers. In maritime environments, object detection systems are essential for monitoring and securing vast ocean spaces, enabling precise detection of ships, buoys, and other objects under varying and often challenging environmental conditions.

While most object detection systems are well-established on land, their applications to maritime environments introduce more specific demands. Maritime object detection systems must confront a fundamental performance-resource trade-off, where high detection performance must be achieved despite limited computing resources on mobile platforms such as ships. To meet the requirements of these environments, object detection models must be adaptive, efficiently optimized for diverse and evolving maritime conditions, and capable of performing under constrained computing resources.

Moreover, most existing methods of object detection systems are tailored to land-based platforms, where factors like stable lighting, predictable object behaviors, and accessible data collection are more manageable. While object detection systems have made

significant advances, they still face common challenges across both land and maritime environments. These challenges are particularly amplified in mobile applications, where environmental variability, data limitations, and resource constraints introduce additional layers of complexity. Key challenges include:

- **Environmental Influences**: Object detection systems deployed outdoors must contend with a range of environmental factors that can affect visibility and detection accuracy. Outdoor conditions such as lighting variations, shadows, and weather conditions (e.g., rain or haze) may impact detection, leading to higher rates of false positives or missed detections.
- **Difficulty of Data Collection**: Collecting diverse and labeled data is essential for developing robust object detection models, yet it remains challenging in both land and maritime contexts. While data collection on land can be resource-intensive, maritime settings add further logistical and cost-related difficulties, as acquiring extensive labeled image and video datasets from remote or mobile sea platforms is particularly complex. This limitation hampers model training and fine-tuning efforts, especially when large, domain-specific datasets are less accessible in maritime applications compared to land-based settings.
- **Resource Constraints on Mobile Platforms** In both land-based and maritime mobile applications, object detection systems often operate on edge devices with restricted computational resources. Computing resources mounted on vehicles, drones, or ships have limited memory, processing power, and energy, creating a need for efficient, lightweight models that deliver accurate detection without exhausting available resources. Unlike cloud-based systems with abundant computational power, edge devices must balance high detection accuracy with constrained resources, making model optimization essential for mobile applications on land and at sea.

To overcome these shared challenges, object detection systems need to enhance their utility across diverse and dynamic environments, for both land-based and maritime mobile applications.

1.2 Motivation

The integration of AI technologies like deep learning, particularly object detection, into maritime surveillance and navigation systems, has become a vital advancement in the shipping industry. These AI-driven systems enable critical functionalities, such as obstacle avoidance, collision prevention, and real-time maritime rescue assistance, which support safe and efficient navigation across increasingly congested waterways. As global shipping demand continues to surge, the need for reliable, high-performance object detection systems in maritime environments has become ever more pressing. However, despite the growing use of AI in maritime operations, maritime accidents remain prevalent, resulting in substantial economic losses and, in some cases, significant human casualties. This situation highlights the urgent need for continued improvement in maritime object detection to further enhance safety and operational efficiency.

Introducing object detection to maritime environments is essential for advancing maritime surveillance, ensuring safer navigation, and ultimately supporting the shift toward autonomous shipping. Object detection systems in this context are tasked with identifying and tracking critical objects, such as other vessels, buoys, and potential obstacles, to prevent collisions and assist in efficient navigation under various conditions. These systems are integral not only to active surveillance and security but also to the development of autonomous sailing capabilities, where timely and accurate identification of surrounding objects is fundamental to safe decision-making. Moreover, high-performance object detection in maritime environments can enhance search-and-rescue operations, environmental monitoring, and the detection of unauthorized activities, such as illegal fishing or smuggling, contributing to overall maritime security.

Compared with autonomous driving on land, researches in the field of Maritime Autonomous Surface Ships (MASS) are still in the early stages, amounts of whose papers are still half of the papers in autonomous driving. Maritime object detection systems confront more challenges due to the complexity of maritime environments and the variety of weather conditions than object detection systems on land. Maritime environments are inherently more variable, with fluctuating lighting conditions, adverse weather, and complex ocean dynamics that can interfere with visual clarity. Reflections from the water's surface, for instance, can distort object outlines, while adverse conditions such as rain, haze, or sea spray introduce noise that traditional models struggle to filter out

effectively. Furthermore, data collection for maritime object detection remains difficult due to logistical challenges and the limited availability of labeled datasets specific to maritime objects and scenarios. These more complicated constraints demand object detection models that are not only accurate but also highly adaptive and resource-efficient, particularly as they are often deployed on edge devices with limited computational resources on board ships.

This research focuses on object detection in maritime environments, with a particular emphasis on developing adaptive optimization techniques tailored to the unique challenges faced at sea. Although many object detection methodologies have been well-established in land-based contexts, adapting these approaches for maritime applications presents new opportunities and complexities. This research aims to address these challenges by designing and implementing object detection models that are both robust in the face of maritime-specific variables and efficient enough to operate within resource-constrained environments. It contributes to safer and more autonomous maritime navigation, supporting the shipping industry's demand for resilient AI-driven solutions capable of performing in challenging and dynamic maritime conditions.

1.3 Challenge

Compared to land-based object detection systems, maritime object detection systems face distinct and often intensified challenges:

- **Complexity of maritime environments and variety of weather conditions** Compared with on-land environments, challenges from maritime environments are often more severe. Sea-specific conditions like light reflections on water, changing weather, and sea spray introduce additional noise and reduce visibility. These factors can distort object features and introduce inaccuracies that standard models struggle to filter out effectively, making environmental adaptation crucial. Additionally, maritime environments often involve moving cameras installed on vessels or drones, introducing motion blur and further complicating detection. These factors create a challenging scenario for object detection models, which must dynamically adapt to shifting environmental and operational conditions to ensure reliable results.

- **Instability and speed constraints of communication**: Under maritime environments, reliable and fast communication links are challenging to maintain due to the vast distances and limited connectivity options available at sea. This instability restricts the ability of maritime object detection systems to rely on cloud-based processing or frequent data transfers, necessitating efficient, locally deployed models that can operate autonomously with minimal dependence on remote servers. In traditional approaches [10] [11], leveraging cloud computing resources for maritime surveillance is challenging due to the mobility of ships. Those centric object detection systems are not well-suited to these constraints, as they often require high-speed, low-latency communication links to function effectively. Distributed maritime platforms must contend with communication delays, packet loss, and limited bandwidth, which can hinder the performance of object detection models and reduce the system's overall effectiveness.

- **Complexity of training, model management, and deployment**: Managing and deploying object detection models in maritime environments is more complex due to the decentralized and resource-constrained nature of these platforms. Unlike centralized land-based systems, maritime systems often operate on edge devices with limited computational capacity. This constraint complicates the training, updating, and fine-tuning of models, especially as they need to be frequently adjusted to maintain performance across diverse maritime conditions. Besides, object detection models for maritime environments must be regularly retrained to maintain accuracy as environmental conditions and data patterns evolve. However, the retraining process is resource-intensive, involving significant computational power, memory, and time-resources that may be limited on edge devices or mobile platforms used in maritime environments. Additionally, indiscriminately retraining models can lead to resource depletion, delayed processing, and system inefficiency, especially when multiple retraining needs arise simultaneously. Therefore, a strategic optimization approach is essential to determine when and how retraining should be triggered, prioritizing tasks based on the model's current performance and available computational resources.

In response to these challenges, this study proposes an adaptive optimization approach for maritime mobile object detection systems. The author's approach is designed to

enable robust and efficient object detection on distributed maritime platforms, enhancing model adaptability in challenging maritime conditions, and efficiency in resource-constrained environments.

1.4 Contribution

The contributions of this study in adaptive model optimization for maritime mobile object detection systems are designed to address the above challenges of maritime environments effectively. These contributions include:

- **Weather-Aware Object Detection**: To counteract the impact of diverse and challenging weather conditions, this study introduces a weather-aware object detection approach. By incorporating weather classification into the object detection pipeline, the system can adjust specialized object detection models to varying environmental conditions such as rain, and haze. This adaptation reduces the degradation of detection accuracy caused by weather noise, ensuring more reliable detection performance across dynamic maritime environments.
- **Consistency Considerations for Model Performance Evaluation**: To correctly evaluate model performance in dynamic maritime settings, this study introduces a consistency-based evaluation approach. By measuring the stability of detection results of time-series data within the same object detection models, the system can identify and address inconsistencies of detection results that may arise from environmental noise or model staleness. This evaluation method complements traditional performance metrics for average accuracy, providing a more comprehensive assessment of model accuracy and reliability in maritime surveillance applications.
- **Efficient Model Selection and Retraining Mechanism**: Given the limited computational resources available on maritime edge devices, this study integrates model selection and retraining in model life cycle management. It enables the system to prioritize and manage retraining tasks based on performance decay and resource availability, allowing for efficient updates without overwhelming the device. By supporting selection among multiple models and retraining as needed, the approach ensures that the object detection models maintain accuracy over time.

- **Flexible Optimization Strategies for Varying Requirements**: Recognizing the diverse operational needs and constraints of maritime systems, the study provides multiple optimization strategies for model retraining execution. It can adapt to different detection requirements, and provide compatibility and flexibility with both model selection and retraining processes, allowing the system to respond effectively to changes in performance, environmental demands, and computational resources. These strategies optimize model efficiency and performance, making the approach suitable for a wide range of maritime surveillance applications.

Together, these contributions form a comprehensive adaptive object detection system, specifically tailored for the complexities of maritime object detection. This approach enhances the adaptivity, accuracy, and efficiency of object detection systems deployed on maritime mobile platforms, addressing key limitations of traditional land-based object detection systems in challenging maritime environments.

1.5 Organization

The remained contents of the dissertation are structured to provide a comprehensive exploration of adaptive optimization strategies for object detection in maritime environments, addressing both approach design and practical implementations across several chapters.

- **Chapter 2** reviews existing literature relevant to the challenges tackled in this research. It begins with an overview of weather-related noise removal techniques, highlighting methods that address environmental noise, such as rain, haze, and sea spray, which impact detection accuracy in maritime settings. The chapter then delves into foundational object detection systems, focusing on AI-based models and various system architectures that inform the proposed approach. The final section of this chapter examines enhancements to object detection models, including consistency performance evaluation, model performance optimization, and strategies for optimizing models under constrained resources.

- **Chapter 3** presents the proposed adaptive optimization approach for object detection in maritime environments. It outlines the design principles, system architecture, and key components of the approach, including weather-aware object detection, model selection and retraining, and optimization strategies for model retraining. The chapter also introduces the consistency-based performance evaluation method, which ensures reliable detection performance in dynamic maritime environments.

- **Chapter 4** focuses on the development of a weather-aware object detection approach to address noise caused by fluctuating environmental conditions. The discussion begins with an exploration of weather noise removal methods, with an emphasis on outdoor surveillance scenarios where visibility is affected by adverse weather. The chapter then introduces an approach called Weather-OD , which is specialized for maritime environments. This model adapts to specific weather conditions through adaptive model specialization and life cycle management, ensuring consistent performance under variable conditions. Implementation details are also provided, covering dataset preparation, noise rendering, weather classification, model retraining, and data collection processes.

- **Chapter 5** is dedicated to improving the efficiency of object detection model retraining. It begins by examining the problem of consistency in object detection, particularly in dynamic maritime settings where environmental shifts can introduce variations in detection accuracy. Additionally, it addresses comprehensive optimization techniques for object detection models, presenting an architectural design and problem statement that identifies retraining triggers for model retraining and selection. The chapter closes with implementation specifics, such as mobile device clustering and model selection that enhance model adaptability.

- **Chapter 6** presents the experimental validation and evaluation of the proposed adaptive object detection approach. The experiments begin with an assessment of the weather-aware object detection model, evaluating the effectiveness of weather noise removal techniques and model specialization under various weather conditions. Consistency evaluation methods are then validated to ensure robustness in challenging maritime scenarios. The chapter concludes with a thorough analysis of object detection model retraining optimization strategies, validating adaptive

methods in resource-constrained environments to quantify performance improvements.

Together, these remained chapters of the dissertation progressively illustrates the design, implementation, and evaluation of adaptive optimization strategies for object detection in maritime environments.

Chapter 2

Related Work

2.1 Overview

This chapter reviews the existing literature relevant to the research concerns of this dissertation. The discussion focuses on key areas that provide the foundation for developing adaptive object detection systems for maritime mobile systems. The following sections describe the background and related work, emphasizing advancements in the areas of object detection in maritime systems. Previous research on maritime object detection systems has primarily aimed to enhance detection performance in complex and dynamic maritime environments.

These studies address critical challenges, including the impact of adverse weather conditions, the design of efficient system architectures, the enhancement of object detection models, and optimization strategies for resource-constrained deployments. Approaches for mitigating weather-induced performance degradation discuss the negative effects of environmental factors, such as rain, and haze, and how to deal with degradation in object detection performance. Object detection system architecture approaches explore architectural designs that support adaptive and efficient object detection on maritime mobile platforms, including device-edge-cloud integration. Object detection model enhancement approaches review techniques for improving object detection models, including

methods for performance consistency evaluation, and model retraining. Constrained object detection model optimization researchers have used various optimization strategies for object detection models operating under resource limitations, focusing on retraining mechanisms, inference efficiency, and profiling-based decision-making. The chapter concludes with a synthesis of these approaches, surveying how existing methods aim to provide efficient object detection systems with the adaptability to respond to changing maritime conditions.

2.2 Weather-Induced Performance Degradation

This section reviews recent methods for mitigating weather-induced performance degradation in object detection systems. Weather-related noise in outdoor image data presents significant challenges to the feasibility of object detection systems. Most common methods focus on removing weather-related noise from image data, which can significantly impact object detection performance. Existing weather-related noise removal techniques generally fall into two categories: those based on physical or optical models, and those leveraging deep learning approaches.

Over the past two decades, physical and optical model-based methods have achieved considerable success in specific, ideal conditions. These methods can be categorized as follows:

- **Physical Modeling Approaches**: This type of method focuses on modeling the appearance of raindrops, rain streaks, and snow streaks. Kim et al. [12] assumed that the direction of rain traces in the rain model is vertical, and detected the rain streak region by analyzing the rotation angle and aspect ratio of the elliptical kernel at each pixel position. Bossuet's team [13] used the amplitude of Gaussian distribution to determine the rain impurity, assumed that the rain falls or snow falls in a near-vertical direction with different ranges, and utilized an evaluator to assess the intensity of the rain or snow to finally realizing the separation of the rain layer from the background layer. These methods typically assume a vertical orientation of rain streaks or snow streaks, identifying noise-affected regions by analyzing the rotation angle and aspect ratio of an elliptic kernel at each pixel. While these techniques are relatively straightforward and

interpretable, their accuracy diminishes when rain streaks deviate from vertical orientation or display inconsistent scale shapes. Consequently, they may fail to capture subtle traces with indistinct characteristics.

- **Morphological Component Analysis (MCA)**: MCA-based methods use bilateral filters to decompose an image into low- and high-frequency components [14]. The high-frequency component is further separated into noise and background elements, after which the noise-free segment is combined with the low-frequency component to yield the final output. Recent advancements [15] have enhanced MCA's performance through K-means clustering, improved dictionary learning strategies, and similar refinements, which significantly accelerate convergence and improve outcomes. However, noise-removed images may still exhibit noticeable blurring, and the dictionary learning process remains computationally intensive, especially when handling substantial rain and snow noise.

- **Filtering Techniques**: These methods in this class employ bilateral filtering [16] or bootstrap filtering [17] to separate rain streaks or snow streaks from an image. This involves decomposing the color image into low- and high-frequency parts, isolating rain streaks within the high-frequency component. Besides, Xu et al. [18] proposed the idea of replacing a single bootstrap filter with multiple bootstrap filters by successively three bootstrap filters and using a bootstrap image to determine the edges that should be retained and those that should be smoothed after distinguishing the edges of the rain streaks from the edges of different objects. While filtering techniques are more versatile and can handle complex weather patterns, they often produce images that lack sharpness and may experience loss of detail and edge information.

- **Gaussian Models**: These approaches apply one or more hybrid Gaussian models to image blocks as priors, aiming to recover the background layer by maximizing a posteriori probability. Zoran and Weiss et al. [19] first proposed the use of Gaussian Mixture Models (GMMs) to model images. Besides, Shin et al. [20] proposed an algorithm for regularization using Gaussian Mixture Models GMMs, which makes the GMMs effective in removing light reflections. Sequentially, Li et al. [21] also proposed a method for image block inspection based on two hybrid Gaussian models. They assumed that the rain layer and the background layer

are independent of each other, and separated different background layers from various natural images to train the hybrid Gaussian model of the background layer. Hybrid Gaussian models offer greater flexibility, as they are not restricted by the orientation or scale of rain streaks and snow streaks. However, the use of variance-based rain streak extraction can reduce reliability.

In contrast, deep learning-based noise removal approaches have gained increasing attention in recent years, especially for handling noise caused by rain, snow, and haze, which is the focus of this discussion. Deep learning methods are often trained on synthetic datasets that contain varying levels of weather-related noise, including clear images without rain, haze, or snow. Researchers such as Huang et al. [22], Fan et al. [23], and Liu et al. [24] have concentrated on extracting preserved information and distinguishing noise features in synthetic datasets to automatically detect and remove weather-related noise. Additionally, Chen et al. [25] integrated weather classification into adaptive noise removal tools to address adverse weather conditions. Although these approaches are effective in removing rain, haze, and even snow from synthetic images, their reliance on specific training data limits their generalizability, making them less suitable for maritime images or real-world maritime object detection.

Since 2020, more versatile methods such as the All-in-One Network [26] and TransWeather [27] have been proposed. These models use one or multiple encoders to process image features and a decoder to generate clean images, improving metrics such as peak-signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) with respect to ground truth clear images. However, despite improving perceptual quality, the output images may still contain visible artifacts and residual noise, potentially leading to object detection failures. Typically, these methods rely on training datasets composed of artificially synthesized images derived from clear scenes. Researchers such as Ren et al. [28], Zhang et al. [29], and Liu et al. [24] have aimed to learn background information and use features of both natural and synthetic weather-related images to distinguish different weather conditions. Their work focuses on identifying and extracting the spatial features of rain streaks, snowflakes, or haze from the image background. However, these methods face challenges when applied to live video streaming due to their high dependency on specific weather condition datasets.

Given the complex, ill-posed nature of weather-related noise in images, Generative Adversarial Network (GAN) have emerged as a leading approach in recent years. Leveraging the robust modeling capabilities of GANs, researchers have made notable advances in removing weather-induced noise from images. Among prominent contributions, Zhang et al. [30] introduced an Image De-raining Conditional Generative Adversarial Network (ID-CGAN). This approach employs a constraint that forces the de-rained output to closely resemble the ground truth, making it indistinguishable from the reference image to the GAN itself. A new refinement loss function further enhances noise removal performance, ensuring greater fidelity in the restored image. Li et al. [31] focused on heavy rain scenarios, proposing a two-stage model: an initial physics-based network that incorporates a raindrop appearance model, followed by a depth-guided GAN refinement network. The first stage employs bootstrap filtering to separate low- and high-frequency components based on frequency differences between rain and background layers. In the second stage, a depth-guided GAN refines the background details that were missed initially, improving the visual quality of the final output. In a different approach, Wang et al. [32] developed a rain removal GAN to address multiple rain types in images. Their method combines a modified ResNet-18, designed to extract deep rain-related features, with a spatial pyramid pooling structure that adapts to diverse rain streak shapes and sizes, effectively capturing and removing various rain types. To enhance the object detection accuracy of image data collected in real-world environments, there is a need for methods that can automatically classify the prevailing weather and adaptively remove the corresponding noise.

2.3 Object Detection Systems

This section reviews the existing literature on AI-based object detection models and system architectures, focusing on the evolution of object detection methods and system designs.

2.3.1 AI-based Object Detection Model

AI-based object detection methods have advanced significantly in recent years, driven by improvements in computational power, large datasets, and deep learning techniques.

These methods aim to automatically identify and classify objects within images, and they play a pivotal role in areas like autonomous driving, video surveillance, and medical imaging. Object detection models are designed to recognize objects in images by identifying their locations and assigning labels to them. Fig. 2.1 illustrates the basic architecture of traditional object detection models, which is structured into three main components: the region selector, feature extractor, and classifier. The region selector is responsible for identifying candidate regions in an image that are likely to contain objects, typically using methods such as sliding windows or selective search. These candidate regions are then passed to the feature extractor, where distinct features are computed to characterize each region. Common techniques for feature extraction include Histogram of Oriented Gradients (HOG) [33], Haar-like feature classifier [34] and Scale-Invariant Feature Transform (SIFT) [35], which capture critical visual attributes necessary for distinguishing objects. The extracted features are finally input into the classifier, which assigns object labels to regions by categorizing them based on the extracted features. Traditional classifiers like Support Vector Machines (SVMs) [36] or decision trees are commonly used here, making the system capable of identifying and classifying detected objects. This structured pipeline underpins traditional object detection by dividing tasks into stages, each contributing to accurate and reliable object recognition.

FIGURE 2.1: Basic Architecture of Traditional Object Detection Models.

These AI-based object detection models are broadly classified into three main categories: single-stage, two-stage, and transformer-based methods. **Single-stage methods** streamline the detection process by directly predicting bounding boxes and class labels in a single pass, enabling faster processing at the cost of slightly reduced accuracy. Notable models in this category include You Only Look Once (YOLO) [37] and Single Shot Multibox Detector (SSD) [38], which prioritize real-time performance. **Two-stage methods**, such as Region-based Convolutional Neural Networks (R-CNN) and its variations, break down the detection task into a region proposal stage followed by a classification stage, resulting in higher accuracy but generally slower speeds. More recently, **transformer-based methods** have emerged as a new trend, leveraging self-attention mechanisms to capture long-range dependencies and contextual information

across images. DEtection TRansformer (DETR) [39] is a pioneering model in this category, representing a shift towards a fully end-to-end detection pipeline without relying on traditional region proposals or anchor-based approaches. These categories highlight the diversity and evolution of AI-based object detection, each addressing different challenges and application needs within the field. Each of these approaches utilizes distinct architectural designs and mechanisms tailored to optimize performance, speed, and accuracy for various applications.

Single-stage object detection methods eliminate the need for region proposal by directly predicting object bounding boxes and class probabilities in a single pass over the image. This direct approach makes single-stage methods fast and computationally efficient, ideal for real-time applications.

- **YOLO (You Only Look Once)** [37]: YOLO divides an input image into a grid and simultaneously predicts bounding boxes and class probabilities for each grid cell. The network architecture consists of a backbone, typically a CNN such as Darknet, which extracts features from the input image, followed by detection layers that output bounding boxes, object scores, and class probabilities. YOLO's efficiency comes from its design, which combines detection and classification within the same network, making it one of the fastest detectors available.
- **SSD (Single Shot Multibox Detector)** [38]: SSD, like YOLO, bypasses the region proposal step and directly predicts object classes and locations from multiple feature maps at different scales. It uses a base Very Deep Convolutional (VGG) network for feature extraction and appends several convolutional layers that predict both bounding box coordinates and class scores.

Two-stage methods divide the detection process into two sequential stages: a region proposal stage and a region classification stage. This approach enhances detection accuracy by focusing computational resources on specific regions that are more likely to contain objects.

- **R-CNN (Region-based Convolutional Neural Network)** [40]: R-CNN pioneered the two-stage detection pipeline. In R-CNN, an image is first processed by a region proposal algorithm, such as selective search, to generate candidate

bounding boxes. Each region proposal is then passed through a CNN to extract features, which are subsequently classified using an SVM. R-CNN's design achieves high accuracy but is computationally slow due to the separate processing of each region.

- **Fast R-CNN** [41]: Fast R-CNN improved upon R-CNN's speed by sharing the CNN backbone across all region proposals, enabling feature extraction to be performed once per image. This design significantly reduced computation time compared to R-CNN, making Fast R-CNN more suitable for real-time applications.

- **Faster R-CNN** [42]: To address the R-CNN model's inefficiency, Faster R-CNN introduced the Region Proposal Network (RPN), which generates proposals directly within the CNN itself, sharing convolutional layers with the detection network. This integration reduces computational cost and improves speed significantly. Faster R-CNN's architecture includes a backbone network (e.g., ResNet [43]) for feature extraction, followed by the RPN, which outputs region proposals. These proposals are then refined and classified by a fully connected network. Faster R-CNN is known for its balance between accuracy and efficiency, making it suitable for high-precision tasks.

Transformer-based object detection represents a newer class of algorithms that use self-attention mechanisms to model long-range dependencies and spatial relationships within images. These methods abandon traditional region proposal and anchor-based approaches in favor of a unified end-to-end framework.

- **DETR (Detection Transformer)** [39]: DETR is a groundbreaking model that uses a transformer encoder-decoder architecture, which is originally designed for natural language processing, to process images. In DETR, the CNN backbone first extracts features from the image, which are then flattened and fed into a transformer encoder. The encoder learns contextual relationships within the image, while the decoder directly predicts object bounding boxes and class labels using a fixed set of learned object queries. This structure eliminates the need for non-maximum suppression (NMS) [44] and anchor boxes, simplifying the pipeline and making it fully end-to-end.

- **Deformable DETR** [45]: As a variant of DETR, Deformable DETR has been developed to address DETR's slower convergence and to improve detection of small objects. Deformable DETR introduces deformable attention mechanisms, allowing the model to focus on sparse sets of relevant spatial points rather than attending to all pixels, which reduces computational complexity and improves performance on diverse object scales.

These three categories reflect a range of architectural innovations, each offering trade-offs in terms of speed, accuracy, and computational complexity. Single-stage methods prioritize speed, two-stage methods focus on accuracy, and transformer-based methods offer an end-to-end approach with the potential for handling complex image contexts. In this study, the weather-aware specialized object detection evaluation experiments in Section 6.3 utilized three representative models: YOLOv5 (single-stage), Faster R-CNN (two-stage), and DETR (self-attention). These models were selected to evaluate the adaptability and effectiveness of the proposed approach across varying detection architectures. For consistency evaluation validation experiments in Section 6.4 and model optimization experiments in Section 6.5, a typical single-stage object detection model YOLOv5 was adopted as the baseline due to its balance of speed and accuracy, as well as its suitability for resource-constrained maritime mobile environments.

2.3.2 Object Detection System Architecture

For realizing object detection in the real world, low latency is crucial in applications such as traffic monitoring and crowd analysis. Given the high data volumes generated by sensor cameras, object detection systems must support low-latency, high-throughput processing.

Existing work based on their **object detection system architectures** can be summarized as follows:

- **Device-Edge Architecture**: A common architecture relies on IoT devices paired with edge servers to enable low-latency object detection near the data source. For example, a latency-aware detection system was proposed in [46], which assigns tasks to workers using a workload optimizer and priority queuing to reduce latency in edge computing. This system provides task offloading selection, task

scheduling, and resource management to optimize the task from camera devices. Another system [47] utilizes IoT devices equipped with GPUs to process portions of the video data locally, offloading only selected tasks to edge servers nearby. It proposed a distributed live video analytics system *Distream* based on the smart camera device-edge cluster architecture, that is able to adapt to the workload of live video analytics. It balances the workload of live video analytics in heterogeneous computing resources consisting of the smart camera devices and the edge cluster. The workload of live video analytics can be dynamically partitioned by a workload adaptation controller on the edge cluster to improve the throughput of the live video analytics.

- **Device-Edge-Cloud Architecture**: Some research has adopted a Device-Edge-Cloud hierarchical architecture to support geographically distributed object detection systems. In one study [48], data was pre-processed and aggregated on private edge servers before being sent to the public cloud, enhancing data privacy and reducing latency. It aims at reducing the processing latency of the video data from distributed traffic monitoring cameras to the cloud by pre-processing and aggregating the data on the private edge servers. Additionally, FilterForward [49] improved object detection throughput by installing lightweight edge filters, which selectively transmit only relevant video frames from constrained edge devices to the cloud, thereby optimizing data transmission between edge and cloud nodes while conserving bandwidth. It extracts features from collected images and filters with a series of micro classifiers to match events of interest, and only transmits the relevant frames to the cloud.

- **Cloud-Based Architecture**: Other research has explored cloud computing for object detection, such as [50] and [51], offering benefits in scalability and cost efficiency. These systems leverage serverless functions on the cloud for processing, but performance can be affected by the underlying infrastructure connecting mobile IoT devices to the cloud. The proposed approach uses a publish-subscribe (pub/sub) messaging system, allowing publishers and subscribers to interact asynchronously without needing direct awareness of one another during streaming. This setup supports latency-sensitive object detection processing on the edge, while data can be transferred to the cloud for long-term storage, large-scale visualization, and further analysis under relaxed latency requirements.

The Device-Edge-Cloud architecture is adopted in this study to support maritime mobile object detection systems due to its ability to balance computational efficiency and adaptability. This architecture enables real-time processing on edge devices for low-latency detection, while offloading complex tasks like model retraining and large-scale data storage to the cloud, effectively meeting the demands of resource-constrained mobile platforms and dynamic maritime environments.

2.4 Object Detection Model Enhancement

This section reviews the existing literature on object detection model enhancement, focusing on consistency performance evaluation and constrained model optimization strategies.

2.4.1 Consistency Performance Evaluation

The consistency problem of object detection is a critical factor to lead to unreliable object detection outcomes, especially in dynamic environments where image content changes frequently. To address this issue, researchers have developed consistency metrics and methods to evaluate model performance across multiple models and datasets.

Wang et al. [52] introduced consistency metrics into model training, enhancing object classification models' interpretability by ensuring the clarity of prediction rationale. To evaluate these models, they developed a consistency metric that measures performance across multiple models on the same dataset. Tung et al. [53] tackled the issue of inconsistent object detection outcomes due to image interference, proposing a consistency measure tailored for time-series images from static surveillance cameras. This approach strengthens model robustness by adjusting for image quality compression and interference, making it applicable to mobile environments where content is dynamic. The method provides real-time consistency measurements during inference, particularly valuable for mobile object detection as image content varies.

Consistency has also been leveraged to enhance model ensembles and semi-supervised learning processes. Liu et al. [54] incorporated consistency into ensemble learning, quantifying model diversity within ensembles to improve generalization. By combining

diverse models, they offset the limitations of individual models, achieving superior performance. Consistency has also been integrated into semi-supervised learning to enhance model training. Tarvainen et al. [55] proposed, a semi-supervised learning method called a mean teacher that enforces consistency between student and teacher models to improve model generalization. It introduced consistency into the model training loss function, which penalizes predictions that are inconsistent with this target. Besides, there are several pseudo-labeling methods that leverage consistency to obtain high-quality data labels for model training. MixMatch [55] is a holistic approach to semi-supervised learning, that utilizes unlabeled data by consistency regularization and MixUp augmentation. ReMixMatch [56] improves MixMatch by introducing two methods including distribution alignment and augmentation anchoring to feed multiple strongly augmented versions of the same data to the model. DivideMix [57] combines semi-supervised learning with Learning with noisy labels. It models the per-sample loss distribution to dynamically divide the training data into a labeled set with clean examples and an unlabeled set with noisy ones. Besides, Sohn et al. [58] proposed FixMatch to generate pseudo labels on unlabeled samples with weak augmentation and only keep predictions with high confidence. Here both weak augmentation and high confidence filtering help produce high-quality trustworthy pseudo-label targets.

Gao et al. [59] introduced consistency into active learning, enabling model training without relying on annotations for all data points. It combined information distilling from unlabeled data during the training stage and consistency-based sample selection metric for electing samples effective at improving model performance. Consistency is integrated into semi-supervised learning to minimize the labeling cost by prioritizing the selection of high-value data. Besides, Jeong et al. [60] integrated consistency into the loss function, enabling teacher-student model training. The consistency constraint is applied for both object classification and localization tasks, enhancing model robustness and generalization. It also proposed background elimination to avoid the negative effect of the predominant backgrounds on the detection performance. This technique compensates for limited labeled data by effectively training models without relying on annotations for all data points.

2.4.2 Model Performance Optimization

To address the issue of model performance degradation caused by weather-related factors in object detection models, two types of approaches have been proposed: **generic models** and **specialized models**. The first type incorporates weather factors into the model design and training to improve performance under bad weather conditions. The second type involves developing specialized models for specific weather conditions, thereby avoiding the limitations of generic models. Unlike previous approaches that focused on noise removal, these approaches aim to enhance the model's ability to detect objects in adverse weather conditions.

The first type of research mainly focused on autonomous driving scenarios. Lee et al. [61] proposed a lightweight network with a task-driven training strategy that incorporates a loss function including task-specific loss from a high-level vision task network, enabling robust image restoration and highly accurate perception tasks. In [62], they presented an online learning approach designed to adapt to three types of bad weather conditions for autonomous driving scenarios. Similarly, Mirza et al. [63] proposed the Dynamic Unsupervised Adaptation (DUA) domain adaptation method to modify the model feature representations by continuously adapting the statistics of the batch normalization layers. Another study [64] redesigned the YOLOv5 model with a semi-supervised domain adaptive method that combines mean teachers and consistency regulation to improve cross-domain detection performance on clear and synthetic hazy images for autonomous driving. Besides, in maritime surveillance scenarios, Liu et al. [65] proposed an enhanced YOLOv3 for ship detection under rainy, hazy, and low-light conditions, with a flexible data augmentation strategy to generate synthetically degraded images.

On the other hand, the second type of method leverages specialization over time through online training that can be applied to weather-specialized models. Model specialization can improve the speed, and accuracy to provide a better user experience on the edge. Shen's research [66] discusses how highly skewed class distributions in day-to-day videos can be classified using much simpler models, reducing the cost of applying classifiers to classify videos. The authors formulate the problem of detecting short-term skews online and exploiting models to realize classification speedup. Cai's team [67] proposes a once-for-all (OFA) network that supports diverse architectural settings by decoupling

training and search, reducing the cost, and allowing for quickly getting a specialized sub-network by selecting from the OFA network without additional training. The proposed progressive shrinking algorithm reduces the model size across many more dimensions than pruning and can obtain a large number of sub-networks that can fit different hardware platforms and latency constraints while maintaining the same level of accuracy as training independently. Other researchers consolidate multiple training datasets and hardware constraints in response to different use cases. Rivas et al. [68] proposed a framework to automatically train specialized models for the context of static monitor cameras, which aims at improving the accuracy of lightweight models by specializing them to the context to run on the edge faster. Besides, Guo's team [69] introduces *Mistify*, a system framework designed to automatically port cloud-based deep learning models to edge devices with various hardware capabilities. It also leverages locally available edge data in a privacy-aware manner and performs run-time model adaptation for better scalability and accurate inference results. They considered a series of specialized models for different camera contexts and data distributions to optimize the performance of their specific conditions.

Its experimental results showed accuracy improvement when the conditions of the training dataset were the same as the ones of testing datasets both on synthetic and realistic ship detection. The first type of research focuses on building a generic model that can perform well across various weather conditions. However, fine-tuning the model to account for the inherent differences between good and bad weather conditions can be challenging. Tuning for weather conditions requires selecting appropriate hyperparameters, which can be difficult. Moreover, a good generic model demands high-quality training datasets, which are challenging to collect in the maritime domain. On the other hand, the second type of research focuses on specialized models for specific weather conditions. These models are fixed and cannot adapt to different weather conditions, making them less flexible. Moreover, collecting all kinds of weather data for a long time to train these specialized models can be time-consuming and costly. Additionally, these specialized models cannot be updated automatically, which can limit their effectiveness in the long term. The proposed approach addresses these issues by automatically switching between specialized models based on the weather classification prediction. Additionally, it reduces the cost and time of data collection by synthesizing degraded image data from clear weather data, promoting weather diversity in the training dataset.

2.4.3 Constrained Model Optimization

In this part, existing constrained optimization methods during the whole model lifecycle in object detection systems are discussed. These methods aim to optimize model deployment and retraining under various constraints, such as latency, resource availability, and accuracy requirements. They are categorized into three main strategies: **inference optimization**, **performance profiling**, **model selection**, and **model retraining optimization**. These methods ensure that models remain accurate and computationally efficient, even as data conditions and environmental factors evolve.

2.4.3.1 Inference Optimization

Optimizing object detection systems often involves minimizing redundant computation and communication, particularly in real-time applications where efficiency is paramount. Two main strategies are commonly used: **input filtering** and **inference result reuse**.

- **Input Filtering**: This approach reduces redundant data transmission by filtering out frames that do not significantly differ from previous ones. For instance, Glimpse [70] uses a pixel-level frame difference detector to monitor scene changes, sending only frames with a sufficient number of significantly different pixels to the edge server. Reducto et al. [71] further refine this process by dynamically adjusting filtering configurations, such as the difference detector feature type and threshold, tailored to specific queries and video content for optimal efficiency.
- **Inference Result Reuse**: This technique focuses on reusing inference results from prior frames in continuous video streams, leveraging temporal redundancy to avoid repeated calculations. DeepCache [72] exploits the observation that most frame content remains static, caching inference results of previous frames and identifying redundant areas through block-wise matching. By reusing the cached results for unchanged blocks, computation is significantly reduced. Similarly, BlockCopy et al. [73] uses deep reinforcement learning (DRL) to identify informative regions of a frame based on the previous frame's information. This method applies block-sparse convolution [74] to process only informative blocks, while features from non-informative blocks are copied from past inferences.

2.4.3.2 Performance Profiling

Performance profiling is a critical component of constrained optimization in object detection systems, enabling efficient model deployment and retraining while balancing resource use and accuracy requirements. By analyzing system performance under different configurations, profiling identifies optimal trade-offs between computational demands and detection accuracy, ensuring the system operates effectively within resource and latency constraints. Additionally, workload placement informed by profiling enhances performance by considering the computational capacity of processing nodes. Profiling methods can be broadly categorized into **offline profiling** and **online profiling**, each offering distinct benefits and trade-offs:

- **Offline Profiling:** This type of method provides a comprehensive analysis of system performance by conducting exhaustive, one-time evaluations across various configurations. It records critical metrics such as accuracy, execution time, resource usage, and memory requirements. For instance, VideoEdge [75] explores a large search space of 1800 configurations, including multiple resolutions, frame rates, object detectors, tracking methods, and tracker placements. To manage the complexity of this search space, VideoEdge merges common components across configurations and caches intermediate results to minimize redundant computations. Similarly, VideoStorm [76] evaluates 414 configurations, requiring approximately 20 CPU days to process a 10-minute video, demonstrating the high computational costs associated with offline profiling. These exhaustive analyses yield detailed performance insights but are computationally expensive and less suitable for dynamic, real-time conditions.
- **Online Profiling:** To address the limitations of static offline methods, online profiling captures runtime data periodically to adapt to changing environmental and system conditions. For example, Chameleon [77] starts with comprehensive profiling and generates a set of adaptable candidate configurations. These configurations are propagated spatially and temporally using cross-camera correlations and content consistency to reduce the need for frequent full profiling. While effective, Chameleon relies on fixed time intervals, which may not capture sudden scene changes. Scene understanding techniques can mitigate this limitation by dynamically triggering re-profiling when significant changes are detected. AWStream

[78] combines offline and online approaches, creating an offline baseline profile and refining it incrementally online. By limiting profiling to Pareto-optimal configurations, AWStream balances accuracy and resource constraints, triggering full profiling only when additional resources are available. Zhu et al. [79] proposed an event-triggered task allocation solution for vehicle fog computing that optimizes latency and quality loss of results under application-specific constraints. It supports task allocation processes including mobile fog node discovery, request sending, task assignment, and task migration scheduling.

These profiling methods play a pivotal role in enabling object detection systems to dynamically adapt to varying conditions, ensuring efficient resource use while maintaining reliable performance. Offline profiling offers deep, static insights, while online profiling enhances responsiveness to low-latency dynamics, making a combined approach particularly effective for maritime mobile object detection systems. In this study, the proposed approach leverages online profiling to adapt to changing weather conditions and resource constraints, ensuring optimal performance in dynamic maritime environments.

2.4.3.3 Model Selection

Model selection optimization focuses on identifying the most suitable model from a set of pre-trained candidates based on current data characteristics. These methods continuously monitor data streams, selecting the model that best balances performance and resource constraints.

Pesaranghader et al. [80] introduced the Error-Memory-Runtime (EMR) metric, which evaluates models based on error rates, memory usage, and runtime. This approach selects models that achieve minimal costs across these dimensions, optimizing deployment decisions. Souza et al. [81] proposed the Kappa-Latency measure, addressing label arrival delays by ensuring stable model performance even in scenarios where actual label information is delayed. Mallick et al. [82] introduced a method to handle both covariate drift and concept drift during model selection, ensuring robustness in dynamic environments. However, these methods primarily focus on choosing the best model from existing options and do not address the need for retraining.

Several frameworks for hyperparameter tuning and optimization also support model selection. The Hyperopt library [83] provides infrastructure for search space definition, minimization, and result analysis, enabling efficient selection of high-performing models. Soper et al. [84] proposed the greedyEarly stopping method, which accelerates model selection processes through greedy cross-validation, reducing time and computational costs. Additionally, Qaraad et al. [85] demonstrated the use of a hybrid feature selection model, to optimize feature subsets, enhancing classification performance. Lee et al. [86] identified three sources of error that model selection algorithms must trade-off, providing a framework for designing algorithms that achieve near-optimal trade-offs. Taylor et al. [87] introduced feature extraction made up of multiple k-nearest neighbor classification models arranged in sequence, to quickly select a pre-trained model to use for given optimization constraints. It aims at minimizing the inference time while meeting the user requirement, by employing machine learning to automatically construct predictors to select at runtime the optimum model to use for input images.

Model selection optimization focuses on choosing the best existing model based on performance metrics and environmental conditions while retraining optimization ensures models remain effective over time. These methods provide complementary solutions for adapting object detection systems to evolving data and resource constraints, enabling efficient and accurate operation across diverse deployment scenarios. The combination of model selection and retraining mechanisms is particularly critical in maritime environments, where conditions change rapidly, and resource constraints are stringent.

2.4.3.4 Model Retraining Optimization

Model retraining optimization tackles the challenge of keeping models current by balancing retraining costs with performance improvements, especially in dynamic environments where data characteristics evolve over time. Hinder et al. [88] explored the theoretical connection between concept drift and loss dynamics, identifying true concept drift through sensitivity and specificity metrics to guide retraining triggers based on test data anomalies. Similarly, Mahadevan et al. [89] introduced the Cost-Aware Retraining Algorithm (Cara), which minimizes retraining expenses by evaluating staleness costs and performance losses from outdated models across data batches, effectively reducing

retraining frequency while preserving model accuracy during gradual covariate and concept drifts. Li et al. [90] presented a multi-user edge intelligence system using federated learning, where end users collaboratively train AI models under the coordination of edge servers. Their approach supports partial task offloading and minimizes energy consumption and latency by optimizing task splitting ratios, bandwidth allocation, and model downloads.

Besides, Zhang et al. [91] proposed a configurable edge computing architecture that customizes task data quality, model complexity, and resource allocation to minimize hybrid energy-delay costs, splitting the problem into resource allocation and task configuration sub-problems. Strubell et al. [92] focused on energy-efficient retraining by estimating cloud computing costs and CO_2 emissions, reducing both financial and environmental impacts. Kasundra et al. [93] developed a decision-point-based framework for model retraining, addressing dataset construction strategies (e.g., data split, data inclusion) and fine-tuning patterns, including incremental and cumulative learning, to improve retraining efficiency.

These methods focus on different constraints and objectives, ranging from resource allocation and energy efficiency to data construction and task optimization. Collectively, they demonstrate the importance of tailoring retraining strategies to meet diverse requirements in dynamic and resource-constrained environments.

2.5 Summary

The existing studies have proposed various approaches through different cases for constrained optimization in object detection systems. These methods focus on varying constraints and objectives, such as resource allocation, energy efficiency, data construction, and task optimization, offering valuable contributions to model optimization in dynamic and resource-constrained environments. However, as summarized in Table 2.1, these approaches exhibit limitations when applied to real-world maritime mobile object detection systems, as they fail to fully satisfy the comprehensive set of requirements outlined in Section 3.2. Most existing methods lack mobility, which is critical for adapting to the dynamic conditions of maritime environments, where detection systems are deployed on mobile platforms such as ships or drones. Similarly, many methods rely on centralized

TABLE 2.1: Summary of Model Optimization Methods.

Cite	Adaptivity	Mobility	No-centralized	Reusability	Consistency	Low-latency	General-purpose
[80]	Y	N	N	Y	N	Y	Y
[81]	Y	N	N	Y	N	N	Y
[82]	Y	N	N	Y	N	N	Y
[83]	Y	N	N	Y	N	N	Y
[84]	N	N	N	Y	N	Y	Y
[85]	Y	N	Y	Y	N	Y	Y
[86]	Y	N	N	Y	N	N	Y
[47]	Y	N	Y	N	N	Y	N
[75]	N	Y	Y	N	N	Y	Y
[91]	N	Y	Y	Y	N	Y	Y
[88]	Y	N	Y	N	Y	N	Y
[79]	Y	Y	Y	N	N	Y	Y
[89]	N	N	N	N	N	Y	Y
This study	Y	Y	Y	Y	Y	Y	Y

architectures, making them unsuitable for decentralized deployments required in maritime settings with limited connectivity. Furthermore, consistency in detection results is largely overlooked, and general-purpose adaptability across changing environmental conditions and user demands is not addressed comprehensively.

In contrast, the proposed method is the only approach that fully meets all defined requirements. It ensures adaptivity through weather-aware specialized object detection models, supports mobility by optimizing deployment for maritime mobile platforms, and enables decentralized operation to account for connectivity constraints. Additionally, it emphasizes reusable models, ensures consistent detection results over time, operates with low latency for real-time applications, and maintains general-purpose functionality to handle diverse maritime scenarios. This comprehensive capability distinguishes the proposed approach as uniquely suited to address the challenges of maritime mobile object detection systems. In the next chapter, the requirements and approach to core ideas will be presented.

Chapter 3

Proposed Method

3.1 Overview

In Chapter 2, recent existing methods for maritime object detection systems were reviewed, although they have provided significant advancements in object detection performance, they still face challenges in dynamic maritime environments. Maritime object detection systems face several problems that hinder their performance in dynamic maritime environments. These problems can be summarized as follows:

1. **Weather-induced Interference:** In maritime object detection systems, the accuracy of object detection is often compromised by adverse weather conditions. Variability in weather such as rain, and haze, significantly reduces visibility, complicating the detection of objects on the water's surface, including ships, buoys, and other navigational markers. Adverse weather conditions, such As heavy rain or dense haze, may cause reflections and overexposure on the water surface, making it difficult for object detection models to reliably distinguish objects from the background. These conditions not only obscure visual details but also increase the risk of false positives and false negatives. Consequently, this interference can lead

to incorrect detections or missed objects, diminishing the reliability of surveillance systems in critical maritime environments.

2. **Difficulty in Fine-tuning an All-in-one Model:** In dynamic maritime environments, using a single, all-in-one object detection model presents significant limitations. Such comprehensive models are typically designed to handle a wide range of conditions and object types, but fine-tuning them to perform consistently well across varying weather conditions, lighting changes, and sea states is challenging. An all-in-one model lacks the flexibility to adapt effectively to specific scenarios, such as rainy or hazy conditions, where environmental noise impacts detection accuracy. Fine-tuning a large, generalized model requires substantial computational resources, which may not be available on the limited-capacity edge devices often used in maritime surveillance. Additionally, these models are prone to underperformance in specific conditions, as they are forced to generalize across highly diverse inputs, leading to increased detection errors and inefficiencies.

3. **Inconsistency of Inference Results:** Object detection models often struggle with consistency in dynamic maritime environments, especially when images appear similar to the human eye but yield inconsistent detection results due to changes in perspective, motion, and environmental factors. Conventional consistency measurement methods, such as those introduced by Tung et al. [53], are generally designed for static surveillance cameras, where the scene remains largely unchanged. However, in mobile maritime environments, the movement of the camera and objects introduces variability that these methods fail to handle. This inconsistency is further compounded by the continuity of data collected from moving objects, which makes annotation challenging, as it is impractical to rely on a static, pre-annotated dataset. Typical consistency measures, which depend on reference datasets, are therefore inadequate for accurately assessing the continuity of detections in real-time, mobile object detection scenarios. In maritime environments, this limitation can lead to unreliable detection results, making it difficult to maintain accurate tracking and identification of objects across consecutive frames.

4. **Model Deployment Resource-constrained Optimization:** Deploying object detection models on maritime mobile devices, such as ships equipped with edge servers, presents unique challenges due to the limited and variable computational resources available at sea. Ensuring consistent, high-performance detection

across varying sea areas with potential data drift, and environmental conditions further increases the demand for resources. Unlike traditional cloud-centralized solutions, these edge devices may operate independently, with limited access to centralized cloud computing, requiring efficient use of onboard processing capabilities to maintain high detection performance. Coordinating retraining, deploying updated models to distributed nodes, and managing model versions across mobile devices is complex and resource-intensive while respecting resource limitations.

3.2 Requirement

AI-based object detection systems deployed in maritime mobile environments face unique challenges, requiring the fulfillment of several critical requirements to ensure adaptive and efficient performance under dynamic conditions:

- **Adaptability:** Maritime environments are inherently dynamic, with frequent changes in location, time, and surrounding conditions. Weather conditions, lighting, sea states, and user requirements for object detection services can shift rapidly. Object detection systems must be adaptable to these variations, maintaining consistent and reliable performance across diverse scenarios. This includes handling different weather conditions such as rain or haze, adapting to varying object types, and meeting changing detection requirements for maritime operations.

- **Mobility:** Maritime mobile devices often operate across different network environments, such as satellite, cellular, and local networks, depending on their location. They also encounter varying weather conditions, such as fair, rainy, and hazy conditions, as they navigate through different sea areas. To support mobility, object detection systems must ensure seamless network handovers and provide weather-aware object detection services, enabling continuous and reliable detection capabilities in diverse maritime contexts.

- **No-Centralized Management:** In maritime environments, devices often operate in areas with limited or unstable network connectivity. Reliance on centralized cloud-based architectures is impractical due to high latency and potential communication disruptions. Therefore, maritime object detection systems must support

decentralized operation, enabling data processing, detection, and decision-making to occur on edge devices or local servers, ensuring consistent performance even in isolated or connectivity-constrained scenarios.

- **Reusability of Object Detection Models:** Maritime object detection systems should prioritize the reusability of models to maximize efficiency. Models trained for specific weather conditions or scenarios should be capable of being redeployed and reused in similar contexts, reducing the need for frequent retraining. This reusability minimizes resource consumption while ensuring that models can adapt to the dynamic nature of maritime environments.

- **Consistency of Inference Results:** In maritime mobile environments, maintaining the consistency of inference results is essential for reliable object detection. Consistent detection results are critical for better identifying and tracking objects, and providing accurate navigation assistance. Meanwhile, the consistency of inference results in time-series data benefits to reduce redundant processing due to occasional detection errors.

- **Low-Latency:** Low-latency object detection is crucial for time-sensitive maritime applications, such as collision avoidance and real-time surveillance. Delays in processing can result in missed detections or late responses to potential threats, compromising safety and efficiency. Object detection systems must deliver rapid processing to provide timely alerts and notifications to operators, ensuring safe and effective navigation in dynamic and potentially hazardous maritime environments.

- **General-Purpose:** Maritime mobile devices collect data on a wide range of object classes, including ships, buoys, and navigational markers. Object detection systems must be general-purpose, and capable of detecting and classifying diverse objects in maritime environments. This versatility ensures that detection results can be applied to various applications, such as navigation assistance, collision avoidance, and surveillance, adapting to the specific needs of maritime operations.

By addressing these requirements, AI-based object detection systems can provide efficient solutions for the unique challenges of maritime mobile environments, ensuring adaptability, reliability, and safety across a wide range of use cases.

3.3 Approach

To address the challenges of maritime mobile environments, this study proposes a comprehensive approach that supports the critical requirements of adaptability, mobility, decentralization, reusability, consistency, low latency, and general-purpose functionality in object detection systems. The approach integrates four core components: weather-aware object detection, model specialization, time-series data result consistency evaluation, and trigger-based model retraining optimization. Together, these components form a cohesive framework designed to enhance detection accuracy, optimize resource usage, and maintain reliable performance across diverse maritime conditions.

Weather-aware Object Detection: A weather noise removal method is introduced to mitigate the impact of weather interference on object detection. This method leverages weather classification to identify environmental conditions, such as rainy, hazy, and fair weather, using data from onboard cameras. Specialized weather noise removal models tailored to each weather condition are then applied, automatically filtering out noise and improving detection accuracy. It serves as a critical pre-processing step to restore image quality, enhancing the effectiveness of object detection models. By reducing weather-induced distortions, such as streaks from rain or reduced visibility from haze, the method improves the clarity and contrast of images before they are passed to the object detection pipeline. Restored images provide a cleaner input for detection models, enabling more accurate identification and localization of maritime objects.

Model Specialization: The problem of fine-tuning all-in-one models in dynamic maritime environments is addressed by employing a model specialization approach. A weather classification model categorizes images based on environmental conditions, enabling the selection of the most suitable object detection model for specific weather scenarios during inference. Grouping images into weather categories and training specialized models for each category reduces the impact of weather-related noise and significantly improves detection accuracy across varied environmental conditions.

To ensure models stay current as environmental conditions evolve, periodic model retraining is incorporated. Given the limited availability of maritime training data, synthetic data augmentation is employed by rendering weather-related noise, such as rain and haze, into existing images. This augmentation enhances the models' ability to

generalize across different weather scenarios. To address the scarcity of training data in maritime environments, where data collection is often constrained by remote locations and limited network access, the approach employs a transfer learning strategy. By adapting pre-trained models with smaller, specific datasets collected at sea, the system achieves high-accuracy analysis even with limited data. During model adaptation, new training datasets are created from video footage recorded during the voyage, and an optimal base model is selected and retrained to enhance detection performance under diverse weather conditions.

Time-series Data Result Consistency: The approach improves the consistency of inference results for time-series image data collected from mobile devices. A flexible, user-configurable threshold based on the Intersection over Union (IoU) index is introduced to evaluate the similarity of images in time-series data. This IoU-based metric quantifies the overlap between inference results in consecutive images, providing a concrete and adaptive measure of consistency. It incorporates a lightweight, autonomous consistency evaluation algorithm, which systematically assesses inference outputs directly at the inference terminal without requiring ground truth data. This algorithm suppresses redundant inference results, improving the accuracy and efficiency of consistency measurements. Designed for integration with mobile devices, this method enables real-time, on-device consistency evaluation, ensuring reliable performance even in resource-constrained maritime applications. This method significantly enhances both the accuracy and efficiency of consistency assessments in time-series image analysis.

Trigger-based Model Retraining Optimization: To address the challenge of optimizing model retraining under constrained resources, a trigger-based retraining optimization method is proposed. It defines specific conditions that trigger retraining actions and assigns urgency levels to prioritize these tasks. Model performance is continuously monitored over defined time intervals, evaluating metrics such as accuracy and response time to determine whether retraining is warranted. Each model is associated with performance thresholds that establish retraining triggers. Additionally, preventive triggers are defined to preemptively initiate retraining before significant performance degradation occurs, ensuring models remain optimized without waiting for critical accuracy drops. Retraining tasks are prioritized using urgency levels, with higher priority assigned to models experiencing greater performance decay.

The optimization method also evaluates the computational capabilities and constraints of each deployment device, ensuring retraining tasks align with available resources. To support dynamic user requirements, the framework incorporates multiple optimization strategies tailored to different scenarios, such as minimizing retraining time, minimizing computing resources, or prioritizing model accuracy. A decision model selects the optimal retraining configuration, balancing resource efficiency and performance improvement while minimizing downtime on low-resource devices.

This approach combines weather-aware object detection, model specialization, consistency evaluation, and trigger-based retraining optimization to provide an efficient solution for maritime mobile object detection systems. It supports the dynamic requirements of maritime environments, offering enhanced detection accuracy, resource efficiency, and system adaptability. The next two chapters detail the design and implementation of the proposed method, describing how these components are integrated to meet the challenges of maritime mobile object detection.

Chapter 4

Weather-aware Object Detection

In this chapter, the design of the proposed system for weather-aware object detection is introduced, with the purpose of enhancing the accuracy of object detection in maritime surveillance systems under various weather conditions. The system consists of two main components: weather noise removal and weather-aware object detection model specialization. The weather noise removal component is responsible for removing noise from images collected by maritime mobile devices in maritime environments. The weather-aware object detection model specialization component is responsible for adapting the object detection model to different weather conditions, such as rain and haze, to improve the accuracy of object detection in maritime surveillance systems. The chapter is organized as follows: Section 4.1 describes the design of the weather noise removal component, which removes noise from images collected by mobile devices in maritime environments. Section 4.2 describes the design of the weather-aware object detection model specialization component, which adapts the object detection model to different weather conditions to improve the accuracy of object detection in maritime surveillance systems.

4.1 Weather Noise Removal

This section introduces the system design for weather noise removal in outdoor environment surveillance. Their algorithms and applications cannot leave long-term data collection from the real world. Currently, most widely used AI-based object detection models work well with clear datasets without noise. However, it's inevitable to face various undesired interference and degradation in data due to complex weather conditions, motions, or object occlusion in the real world. For example, an online monitoring system, where cameras on outdoor mobile devices collect images/video of cities or natural environments, faces the problems of data degradation due to weather conditions, e.g., rain, haze, snow. The **problems** are summarized as follows:

1. It's hard for existing AI-based models to directly utilize degraded data collected from more complicated real situations and satisfy their requirements.
2. Previous researches about degraded data mostly focus on one particular situation respectively, from simple signal processing methods to complicated deep learning methods including rain removal [28], haze removal [94]. They cannot work well to handle comprehensive weather conditions in the real world.

To address the aforementioned problems, the author proposes an approach that adaptively removes noise due to various weather conditions in image data collected by cameras. It utilizes a multi-class weather classification model to classify several common weather labels for input images. Then, it removes noise by running suitable noise removal algorithms, which also supports edge computing for data analysis by adjusting image data size so that the data fits in a small memory in each edge device.

The proposed approach will make significant **contributions** to reducing burdens of adaptation program for application developers with a user-friendly command line, providing better clear data collection under different weather conditions for AI algorithm constructions.

4.1.1 Scenario: Outdoor Surveillance Scenario

This study assumes the use of an outdoor surveillance system that is capable of performing video analyses of video streaming data collected from cameras mounted on mobile devices, as shown in Fig. 4.1.

- **Deployment environment:** It assumes a university with several campuses. In this scenario, static devices equipped with sensor cameras are installed in the school buildings and along roadsides, and mobile IoT devices are installed in drones and vehicles moving around the campus, or sometimes even across different campuses.
- **Video stream data collection:** Outdoor devices equipped with sensor cameras continuously collect data from the campus year after year. Therefore, they often record video stream data with low visibility due to inclement weather conditions such as rain, haze, and snow.
- **Video analysis:** The collected data is uploaded to edge servers located on different campuses or to cloud servers in data centers for video analysis and data visualization via AI services, such as object detection.

In the above scenario, it will be difficult to collect video from mobile devices and process data with weather-related noise by using high accuracy, provided video analysis models are only trained by clear data in good weather conditions. To improve the accuracy of video analysis by removing noise due to inclement weather, the author proposes an approach that can generate clear video by automatically classifying the weather conditions and performing appropriate noise removal and brightness enhancement.

4.1.2 Design

This part assumes an online monitoring system, where cameras on outdoor devices collect images and image processing applications, e.g. object detection based on AI models, and analyze the collected data on cloud platforms or edge nodes. The camera devices outdoors collect long-termly various images partially degraded by various weather conditions. Thus, in the existing work, the application user needs to manually classify

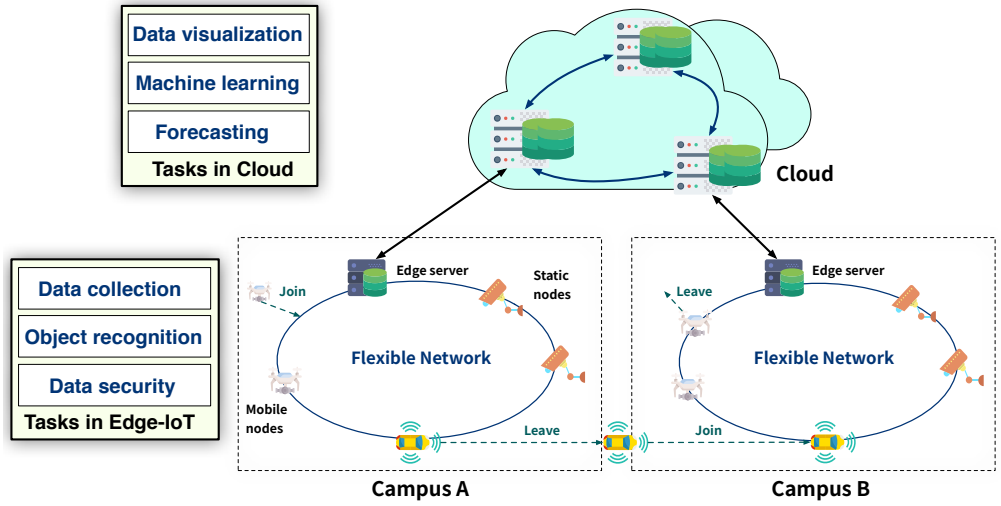


FIGURE 4.1: Scenario: Outdoor Surveillance with Outdoor Devices.

the input images and select suitable noise removal algorithms for different weather conditions. The goal is to adaptively classify the input images by their noise and select suitable noise removal models instead of the user.

The architecture of the adaptive image noise removal tool shown in Figure 4.2 is divided into three parts: **multi-class weather classification model** is responsible for classifying weather conditions of images; **adaptation model** selects suitable noise removal models with different prediction results; and **image resolution resize function** provides functions that adjust image sizes and other parameters of output images for users.

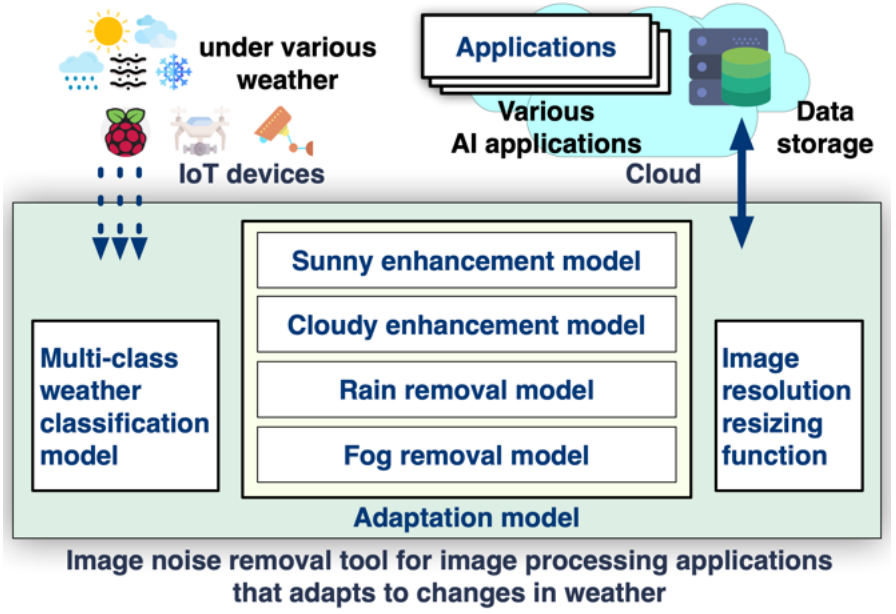


FIGURE 4.2: Architecture of Adaptive Noise Removal Tool.

The multi-class weather classification model predicts the weather in the input image. It is created with very deep convolutional networks by Visual Geometry Group (VGG) [95] distinguished from several CNN architectures for the feature extraction [96]. The author utilized and augmented a five-class weather image dataset [97] including cloudy, hazy, rainy, fair weather, and sunset, to create the model. The proposed adaptation model used the method of **parameter changes** for implementing an adaptation mechanism that can switch to different image enhancement and noise removal models conforming to the prediction results. For fair and sunset images, the adaptation model will switch to the fair enhancement models, which directly output the raw images. For cloudy images, it will switch to the cloudy enhancement models, which can modify them to more moderate ones on contrast and luminance. Also, when rainy or hazy weather results, it will switch to the rain, hazy removal model for removing rain streaks or hazy blur of the input images respectively. In the image resolution resizing function, users can indicate several parameters allowing for adjusting output image resolution, timestamp, image effect, saturation, and colorfulness of output images.

4.2 Weather-OD: Weather-aware Object Detection Model Specialization

In this section, the system design for Weather-OD for maritime surveillance is introduced. The details of the design for adaptive model specialization for different weather conditions are described in 4.2.2.1, and of the lifecycle management object detection models in 4.2.2.3.

4.2.1 Scenario: Maritime Surveillance Scenario

A proposed scenario for enhancing maritime surveillance of ships at sea is similar to the one depicted in Fig. 4.3. The object detection system in this scenario includes devices and edge servers installed on board, along with a cloud server located on shore.

In the cloud, these maritime object detection models can be fine-tuned from an off-the-shelf, pre-trained model repository like PyTorch Hub. As a prerequisite, maritime object detection models can be built from the current maritime dataset from pre-trained model

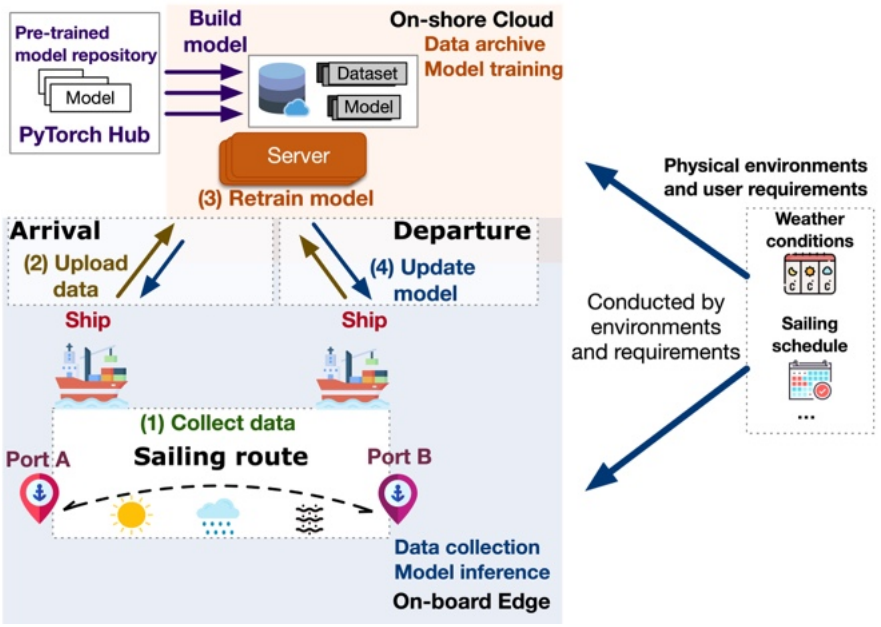


FIGURE 4.3: Scenario: Maritime Surveillance System.

repositories such as PyTorch Hub before the maritime surveillance application launches. Throughout their sailing route, ships may encounter various adverse weather conditions like rain and haze, which impair the quality of captured image data. Equipping on-board edge servers on a ship with the object detection models beforehand based on information from weather forecasts and sailing schedules would be beneficial in achieving highly accurate object detection while sailing, particularly in adverse weather conditions. Concretely, the aim is to achieve the following in implementing this scenario:

1. As a ship sails to port A, it utilizes maritime object detection models deployed in advance to provide results for maritime surveillance by inference from the collected data. It may encounter weather conditions, such as rain or haze during its voyage.

2. Throughout the voyage, the data collected under diverse weather conditions and from different seas and weather conditions are temporarily stored until the ship approaches a stable network near the port. The data can then be uploaded to the onshore cloud server via the network.

3. The datasets contain images of all weather conditions, grouped together based on weather classification, and several new training datasets are created by annotating the objects to be detected. In the cloud, It can update the model for specific weather conditions, thereby improving the performance of object detection for

this particular sailing route. This is achieved by retraining with this new dataset, taking advantage of the high computing performance of graphics processing units (GPUs).

4. The ship can download and redeploy the newly trained model from the cloud to its local edge server at ports A and B.

The model can also be downloaded and redeployed on other ships, offering additional ships the opportunity to benefit from this system, as described in steps (2) and (4) of this scenario. To improve the accuracy of object detection, it is essential to handle object detection for different weather conditions and manage inference model versions updated on the edge for continuous accuracy improvements.

4.2.2 Design

This part details the design of Weather-OD for maritime surveillance, and the design outline of the system is shown in Fig. 4.2. The system consists of camera devices and edge servers installed on ships, as well as cloud servers located on shore. The edge servers and the cloud servers are connected via network services, e.g., 4G/5G mobile networks on coasts, even Wi-Fi in ports, while they are available. The connection is closed in areas where the network services are unavailable, e.g., ocean-going areas.

On-board edge servers are computing systems that are meant to do data processing and analysis at the network's edge, close to where data is generated from video sensors, particularly in maritime environments where connectivity to the cloud may be limited. The system of **on-board edge** servers consists of several components as follows:

- The *model manager* component manages the execution of models, including loading, running, and unloading the models. It is responsible for managing the deployment and execution of these models, ensuring that they are running correctly and efficiently.
- The *system & model monitor* component monitors the system's health and performance, including GPU and memory usage, to ensure that the system is running efficiently. Besides, it is responsible for monitoring the performance of the models, collecting data on their accuracy and efficiency, and providing feedback to the

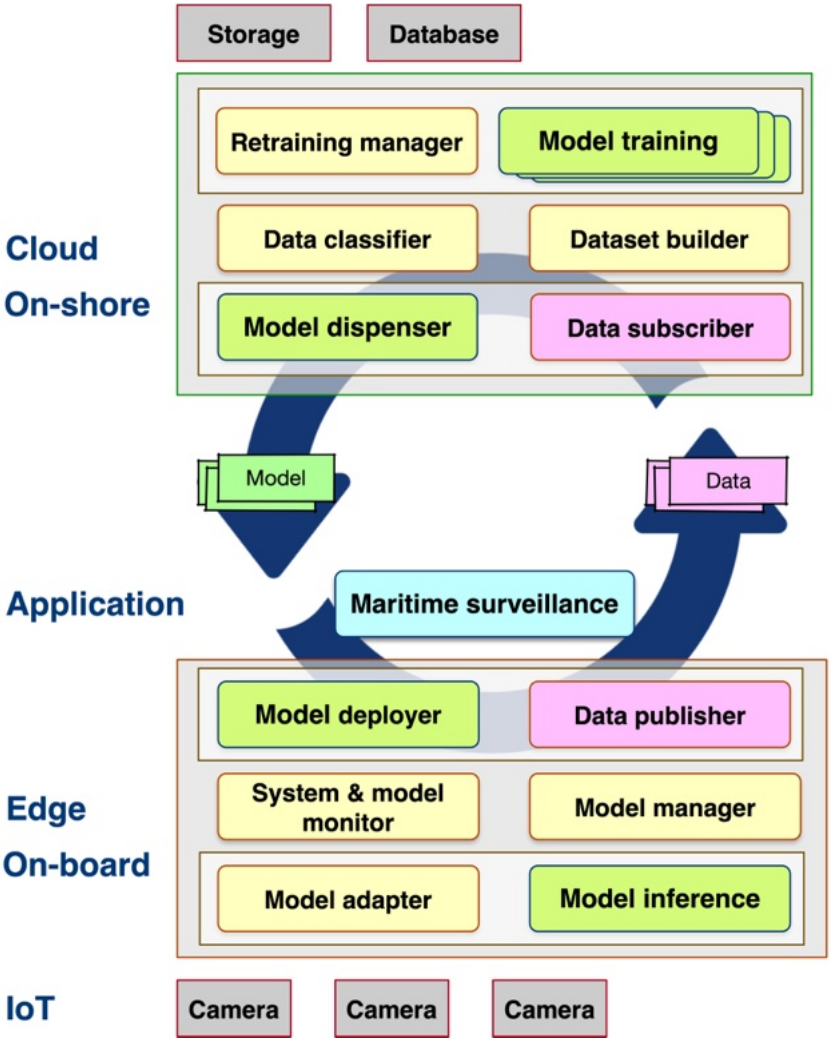


FIGURE 4.4: Automatic Model Specialization System Design.

model manager and model inference components to optimize the performance of the overall system.

- The *model adapter* component switches suitable specialized models for inference according to collected image weather type and weather forecast information.
- The *model deployer* component manages the deployment of new models from the cloud, including downloading new model files and installing their software libraries on the edge server.
- The *data publisher* component is responsible for publishing the collected data to build a new dataset for downstream tasks on cloud servers.

These components support the **model inference** process to execute deployed object detection models to identify objects in input data. By integrating these components, on-board edge servers can form a comprehensive system that enables maritime surveillance under different weather conditions and can be combined with model specialization to further improve performance.

On-board edge servers are in charge of running image analytics, or inference using an object detection model. The system design on **on-shore cloud** servers includes several components as follows:

- The *data classifier* component sorts data into different categories, such as weather conditions or object types, which is necessary for developing specialized models.
- The *dataset builder* component is responsible for adding annotation information to the data, such as bounding boxes, to enable supervised learning. Additionally, it generates a synthetic dataset from the original clear dataset to complement the lack of an adverse weather dataset.
- The *retraining manager* component manages triggers and schedules for retraining and configures parameters for executing the process of the retraining model.
- The *data subscriber* component refers to the process of subscribing to the data generated by the on-board edge servers for data archive and dataset building.
- The *model dispenser* component deploys the trained models to the on-board edge servers.

These components support the **model training** process of training or retraining the specialized models using datasets with different weather conditions.

4.2.2.1 Adaptive Model Specialization

Weather-OD is adaptive, allowing for the adoption of specialized models for various weather conditions encountered during a voyage, thereby improving object detection performance. The proposed approach does not adopt a large versatile object detection model, considering limited edge computing resources, different from previous works [65] [68]. Adaptive model specialization involves training specialized models for specific

weather conditions that are then used for object detection under various weather conditions. It allows the creation of models that are optimized for specific tasks or conditions rather than creating a single model to handle all types of weather.

For inference on the edge, the weather information of the voyage routes from the weather forecast is used as one of the main bases. Weather-OD also classifies the weather using a weather classification model to calibrate the real-time weather conditions for the collected data, whose final weather classification results are used to adapt the learning model for the edge. In view of computational performance, the computational resource demands in the specialized model are smaller than those in the single models, so it fits into the capacity of resource-constrained edge servers.

4.2.2.2 Weather-aware Object Detection Algorithm

This part details how to perform weather-aware object detection in our proposed scenario, depicted in Fig. 4.5 separately from the inference on the edge and training on the cloud. In the inference part, the data flow is as follows:

1. Collect a diverse set of images under various weather conditions, including fair, rainy, and hazy weather conditions.
2. Utilize our weather classification model to automatically classify the collected images based on their weather conditions.
3. According to the weather classification prediction of the input image, select the corresponding specialized object detection model to perform object detection on the image.
4. After weather classification, the predictions can be used to select suitable object detection models for inference.

In the training part, the data flow is as follows:

1. Aggregate data collected from ships through data messaging to the cloud for data storage.
2. Annotate the images by placing bounding boxes around the objects of interest, such as ships or other maritime objects, to create a training dataset.

3. Divide the annotated dataset into several subsets based on the weather conditions and augment rainy and hazy data with noise rendering.

4. Train a series of specialized object detection models for fair, rainy, and hazy conditions using the associated annotated data.

5. Evaluate the trained models with test data to confirm their qualification and deploy qualified new models to the edge servers.

6. Monitor the inference models' performance on on-board edge servers, assess data shift, and determine the next model retraining triggers.

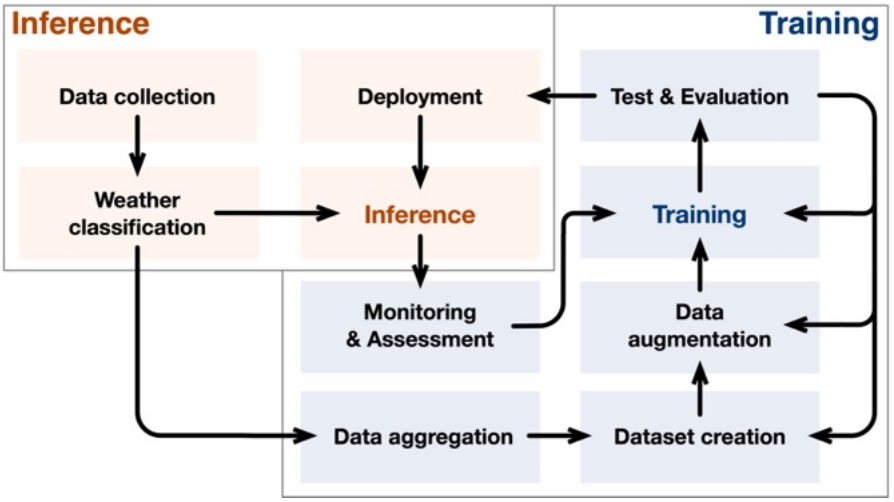


FIGURE 4.5: Overview: Weather-aware Object Detection.

Algorithm 1 Inference on the edge.

Input: Collected data D, Specialized models M
Output: Inference predictions P

```
P ← [φ, ..., φ]
j ← 0                                   ▷ Initialize fair model for inference
for all d ∈ D do
    t ← classify weather of d
    j' ← type t of model index
    if j'=j then
        p ← M[j] predicts d
    else
        p ← M[j'] predicts d
        j ← j'
    P.add(p)
    j ← type t of model and subset index
    return P
```

The inference process of our weather-aware object detection approach Weather-OD , is outlined in Algorithm 1. Assuming our approach is used to the dynamic and mutable weather at sea, Algorithm 1 aims to adaptively select the most appropriate specialized model based on weather conditions. The **input** consists of a dataset D comprising data gathered from ships, and a collection of specialized inference models M deployed on the edge servers. Set the index of the initial specialization model as 0, representing the fair weather model, as the initial inference model. For each data frame d in the dataset D, classify and predict the most likely weather type t. Derive the index j' corresponding to the specialized model for this weather type. Then compare the derived index j' with the current model index j. If they are not the same, replace j with j' and switch to the specialized inference model j'. Obtain the prediction result p using the selected model, and output all prediction results P. The **output** is a list P containing inference predictions for all the collected data, tailored for maritime surveillance applications conducted on the edge.

Algorithm 2 Training on the cloud.

Input: Collected data D, Inference predictions P, Specialized models M
Output: New specialized models M'

```
S, M' ← [φ, ..., φ]
for all d ∈ D do
    if j = 0 then                          ▷ Render noise when type is fair weather
        S[j].add(d)                        ▷ Add original data to dataset
        for j ← 1 to length of M do
            t ← weather type of specialized model M[j]
            d' ← render weather noise of type t on d
            S[j].add(d')                   ▷ Add synthetic data to dataset
    else
        S[j].add(d)
while over update interval do              ▷ Retrain models periodically
    for j ← 0 to length of M do
        if accuracy of predictions P by M[j] is low then
            M'[j] ← retrain model M[j] on S[j]
        else
            M'[j] ← M[j]
    return M'
```

The training process of our weather-aware object detection approach Weather-OD , is outlined in Algorithm 2. Following the upload of data D and prediction results P to the cloud, Algorithm 2 is designed to augment the data and perform continuous training of specialized models on the cloud. The **input** includes the collected data D and prediction results P, both of which are uploaded to the cloud. Additionally, the current specialized

models M are also part of the input. For data frames d classified as fair weather, render noise to simulate rainy and hazy conditions, which will be explained in 4.2.3.1. Generate synthetic data d' and add it to the appropriate new data subset S based on the corresponding weather type. Continue iterating through the collected data, until all frames in the dataset D have been processed. When the predefined update interval is reached, update each specialized model if low accuracy of prediction P, and store new specialized models in M'. The **output** comprises new specialized inference models M' that are tailored for each specific weather type, and these updated models are stored on the cloud platform. Through the utilization of both Algorithm 1 and 2, the configuration and execution of the computational processing can be facilitated.

4.2.2.3 Lifecycle Management of Object Detection Model

Weather-OD involves both the training cycle and the inference cycle for managing the lifecycle of object detection models. These cycles ensure the continuous improvement and effectiveness of the maritime object detection models. Fig. 4.6 illustrates the lifecycle management of the object detection model in Weather-OD , not available in existing studies.

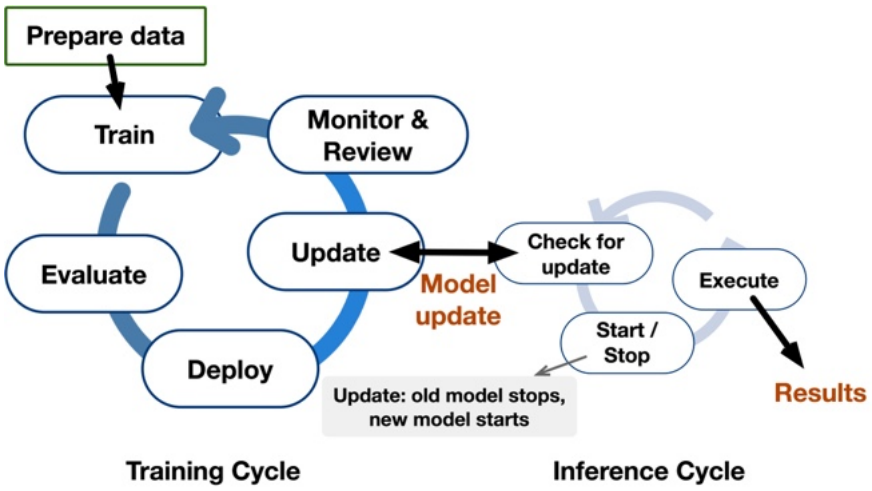


FIGURE 4.6: Lifecycle Management of Maritime Object Detection Models.

The training cycle begins with the creation of a new object detection model through training on the dataset and applying data augmentation techniques. After the training process, the model's performance is evaluated to assess its suitability for deployment and identify areas that may require further improvement. This evaluation helps ensure that the model meets the desired performance criteria. Once the model has been evaluated

and deemed suitable for deployment, it is integrated into the existing edge servers, making it accessible to end-users, including sailors and other stakeholders. The deployment process involves setting up the necessary infrastructure and configurations to enable the model to operate effectively on the edge servers.

During the inference cycle, the deployed model can be started and executed by the model manager, which allows the model to generate inference predictions based on the input data received. Periodically, the model manager checks for updates to the model. If a newer version of the model is available, the old model can be stopped, and the new model can be started and executed for inference. This periodic update ensures that the most recent advancements and improvements in object detection are incorporated into the system. The predictions generated by the inference process can be saved and reviewed for error rates and other performance metrics. This review provides valuable insights into the model's performance and helps identify areas where further model retraining or refinement may be necessary. By following this lifecycle management, the maritime object detection system can continuously learn, improve, and adapt to evolving conditions, thereby enhancing its accuracy and effectiveness over time.

4.2.3 Implementation

The author has implemented functions in Weather-OD to solve issues of performance degradation due to weather-related noise and the lack of training data for maritime object detection. This section discusses the functional implementation. It includes the current maritime dataset digest and data augmentation for adverse weather training data in 4.2.3.1, weather classification for model specialization in 4.2.3.2, model updating through incremental learning in 4.2.3.3, and data collection to the on-board cloud in 4.2.3.4.

4.2.3.1 Maritime Dataset and Noise Rendering

Commonly used datasets for maritime object detection can be classified by their shooting angles, images captured from on-board or those from shore [98] [99], and bird's-eye view images captured by Unmanned Aerial Vehicles (UAV) and satellites [100] [101]. The former dataset can be used for the maritime surveillance scenario discussed in this paper,

and the latter dataset can be used for other objectives, e.g., rescue following maritime accidents. Furthermore, the complexity and diversity of the maritime environment make it difficult to collect comprehensive and representative training data. Therefore, it is crucial to take full advantage of the limited collected data to create maritime training data that can accurately represent the variety of objects and conditions found in the maritime environment.

FIGURE 4.7: SMD Sample Images and Baseline Results.

In Weather-OD , data augmentation is incorporated to create more weather synthetic noise data from collected clear data, i.e. noise rendering, instead of typical methods, e.g., rotation, scaling, and changing the lighting conditions. Some research [102] [103] [104] has used CycleGAN [105] to generate synthetic data with specific patterns, such as day to night or fair to hazy. These synthetic images still had unrealistic details, and the diversity of the dataset is limited with either of the methods when they are applied to maritime images. Thus, training CycleGAN needs to be carefully designed and validated to generate high-quality synthetic data that are representative of the original data.

In the implementation, CycleGAN models were used to increase the diversity of the dataset. Rain and haze noise were extracted from the JRSRD [106] and O-HAZE datasets [107], and combined with the Singapore Maritime Dataset (SMD) [98]. The CycleGAN model generator learns to transfer the noise from the open-source dataset images as the target domain to the collected maritime images as the source domain, while the discriminator learns to distinguish between original and generated images. During the training process, the performance of the model was monitored by calculating evaluation metrics such as adversarial loss, cycle consistency loss, and identity loss. The

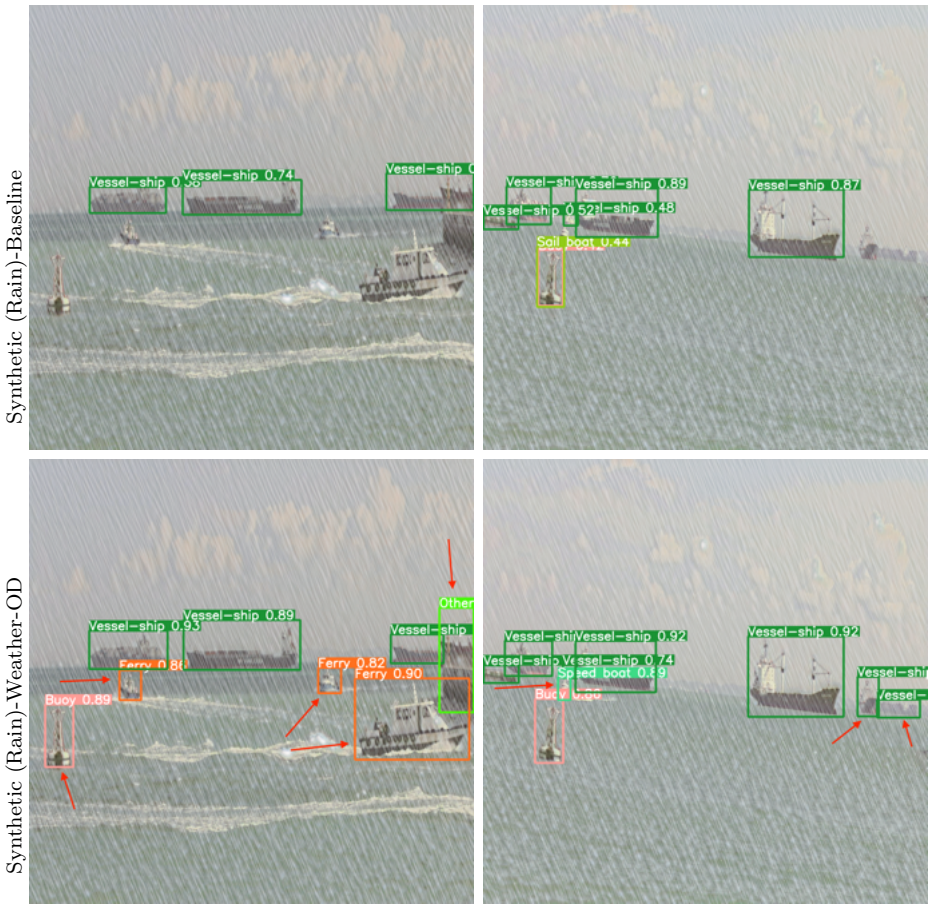


FIGURE 4.8: Synthetic (Rain) Baseline and Weather-OD Results.

trained models allowed the generation of images of the SMD with rain and haze noise, which were used to train and validate other computer vision models, such as object detection models, under different weather conditions.

Fig. 4.7 displays example images, while Fig. 4.8 and Fig. 4.9 show the same example images generated using the noise rendering models proposed in the implementation. These images show the inference results of a baseline model trained on the SMD training data, and models with the proposed weather-aware approach based on YOLOv5. It can be observed that the limitations of the existing baseline model in handling weather-related noise, as indicated by the reduced accuracy in object detection. On the other hand, the images depict the improved inference results obtained by utilizing Weather-OD , with more correctly detected objects and higher confidence scores. The red arrows indicate the enhanced accuracy of object detection in rainy and hazy conditions. It demonstrates the approach effectiveness in mitigating the impact of weather-related noise and improving the overall performance of object detection with direct observation.

FIGURE 4.9: Synthetic (Haze) Baseline and Weather-OD Results.

4.2.3.2 Weather Classification

In this approach, automatic weather classification of collected data for training, and adaptive specialized model adoption for inference rely on correct and fast weather classification. Concretely, the purpose of the weather classification model is to automatically classify collected maritime data under different weather conditions, such as fair, hazy, and rainy. The weather classification model has been trained on a labeled dataset of these different weather conditions to recognize the corresponding weather category for each image. The weather classification model [108] is used to determine the weather conditions for the input video. To implement a weather classification, complex weather classification types including fair, rainy, hazy, and snowy (the snowy type is not used in this study) are defined. To classify different conditions, an open-source multi-weather image dataset consisting of about 10,000 images for each weather type is utilized. Since different adverse weather types such as rain and haze are very similar, other weather classification studies suffer from problems such as category imbalance and overfitting.

Therefore, EfficientNet [109] is incorporated into the weather classification model, which is one of the most advanced neural network frameworks developed recently, while adopting a model that can achieve excellent Top-1 level accuracy using only approximately 19 million parameters. Furthermore, an evolutionary algorithm [110] is used through iterative hyperparameters such as learning rate, batch size, and weight decay until the convergence of the best solution is reached to solve these problems.

4.2.3.3 Model Retraining and Updating

Maritime object detection systems often require retraining due to concept drift in ever-changing environments, such as varying weather patterns or ship traffic. Incremental learning is necessary to train the model with available data and continually retrain it as new examples appear. However, **catastrophic forgetting** [111] is a major limitation of incremental learning, where the model loses its previously learned knowledge when it is trained on new data.

Model retraining can better leverage the previously learned features and result in faster convergence to the global optima of model training than training the model only on new data from scratch [112]. A balance must be struck between retraining on the entire dataset and incremental learning with a limited set of newly captured examples to achieve the best trade-off between accuracy and computational efficiency. To address this issue, several techniques can be employed, such as weight regularization [113], rehearsal strategies [114], and knowledge distilling techniques [115] [116]. In the implementation, the base models were trained on the new data using transfer learning, freezing partial backbone layers of the base model, and fine-tuning others to avoid overfitting. It also added a sampling of old data from past appropriate periods and routes to the training data to mitigate the problem of limited training data, considering the fixation of sailing routes, and periodic changes in hemispheres, seasons, etc..

4.2.3.4 Data Collection

In this part, the data collection is implemented from on-board edge servers to on-shore cloud servers using SINETStream [117]. The SINETStream software library [118] is designed to facilitate the development and deployment of secure and efficient IoT applications, with features that include portability, security, and performance tuning support. It utilizes a topic-based publish-subscribe (Pub-Sub) messaging model, which can be used by maritime object detection systems to implement messaging between on-board edge servers and on-shore cloud servers using a variety of SINETStream-supported message brokers.

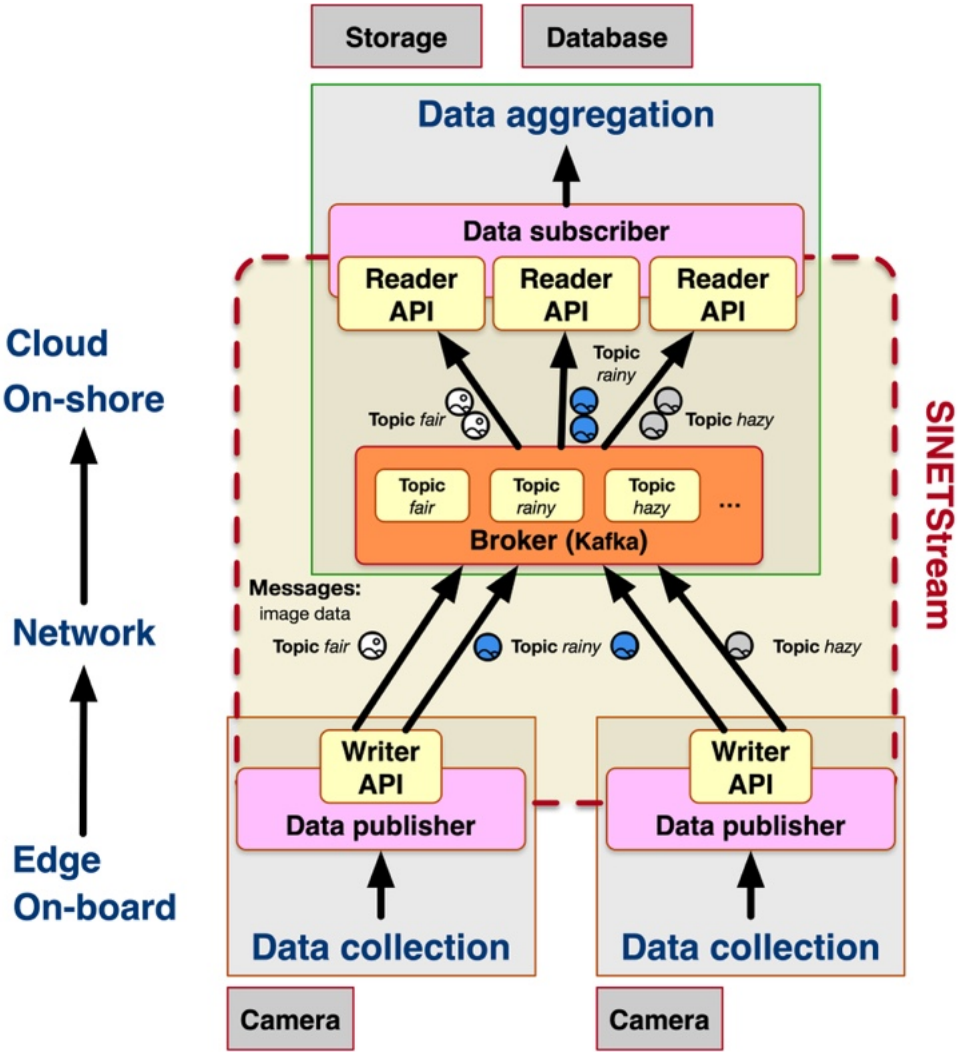


FIGURE 4.10: Data Collection with the SINETStream APIs.

The implementation includes a publisher program called *Writer* in on-board edge servers shown in Fig. 4.10, which sends the collected image data, and a subscriber program on

the cloud called *Reader* that processes the collected data, corresponding to the main functions of *data publisher* and *data subscriber* in 4.2.2. The *Writer* and *Reader* programs use the SINETStream Writer/Reader APIs [119] to send and receive sensor data streams through the broker located on the cloud server by setting the same topic. The message broker can handle a large number of messages, ensuring that this system can scale with the number of devices and volume of data. Different topics can be set to be divided into different weather conditions, routes, and ships, according to the training data requirements. To summarize this section, four technical challenges were tackled and the corresponding solutions were developed. A weather classification model and noise rendering CycleGAN models were constructed to enable adaptive model specialization. Additionally, topic-based messaging was implemented to facilitate data aggregation from remote ships, while also enabling model updates through retraining on collected incremental data under model lifecycle management.

4.3 Summary

In the first method, a multi-class weather classification model and several image noise removal algorithms are incorporated to design an adaptive image enhancement and noise removal tool for maritime object detection under different weather conditions. After that, image resolution resizing can reduce uploading image data sizes to satisfy more critical time requirements of maritime object detection systems when deploying to edge nodes. In the second method, a weather-aware object detection method Weather-OD for maritime surveillance systems is proposed and evaluated. Weather-OD automates object detection model specialization for ships and other maritime objects by incorporating automatic weather classification for weather data augmentation and adaptive model specialization for the three most common weather categories, namely fair, rainy, and hazy conditions. Additionally, Weather-OD can continuously capture data during inference, periodically update specialized inference models, and manage the lifecycle of the specialized models deployed on the edge.

These methods for enhancing object detection in maritime surveillance systems under various weather conditions are promising, but they still have limitations. In real maritime surveillance systems, the object detection model enhancement also needs to consider the constraints of the deployed platform and the requirements of accuracy.

Therefore, in the next chapter, the author will introduce the consistency of object detection results, and optimize the object detection model retraining execution to improve the performance of models under challenging budgeted conditions.

Chapter 5

Object Detection Model Optimization

In maritime object detection, ensuring high performance involves addressing not only weather-induced degradation but also inconsistencies in inference results for time-series images. These inconsistencies, caused by moving cameras and dynamic environments, pose a significant challenge to maintaining reliable object detection across consecutive frames. This chapter presents two complementary approaches to improve the performance and reliability of maritime object detection systems. First, a method described in Section 5.1 is proposed to evaluate the consistency of object detection results in time-series images by incorporating image similarity metrics tailored to moving camera scenarios. This method dynamically measures and enhances consistency, ensuring accurate object tracking and detection in time-varying maritime environments. Second, it introduces an optimization method described in Section 5.2 for object detection models, focusing on improving model update and retraining efficiency in maritime mobile systems with constrained computational resources. By designing an adaptive retraining mechanism and prioritizing optimization strategies, the method ensures effective model

performance while balancing resource usage and system constraints. Together, these approaches tackle key challenges in maritime object detection, contributing to more robust, efficient, and reliable systems for dynamic maritime applications.

5.1 Consistency problem

Before introducing the proposed method for evaluating the consistency of object detection results in time-series images, it is essential to understand the consistency problem posed by moving cameras and dynamic environments in maritime object detection.

5.1.1 Concept

Object detection researches utilizing machine learning play crucial roles in application domains using mobile devices such as environmental monitoring and autonomous driving. In object detection in mobile devices, there is a growing demand for improved inference accuracy directly correlated with safety and reliability. However, existing researches for evaluating consistency face the challenge of accurately measuring inference results of object detection of mobile devices due to variations in the content of temporal image sequences. Consistency in the study is defined as the difference in predictions obtained from object detection models between similar images.

In the authors' previous work on mobile object detection [120], retraining of weather-specific object detection models to reduce the effects of weather noise has improved inference accuracy. However, regardless of data expansion and the use of machine learning techniques to improve the accuracy of object detection models such as dropout [121], inconsistencies in inference results between similar images remain a challenge. Discrepancies in object detection results between similar images within continuously captured time series data are a major problem faced by conventional accuracy metrics. However, it is difficult to accurately measure the inference results of object detection for moving objects due to changes in image content, because existing consistency metrics [53] for inference results between successive images have been developed mainly for static surveillance cameras fixed at a certain location.

In this study, an object detection scenario is assumed for time-series images collected from cameras installed on moving objects. In order to evaluate the consistency of object detection results for similar images collected while moving, the author proposes a configurable threshold evaluation method that takes image similarity into account. By quantitatively evaluating the similarity between time-series images and dynamically setting a consistency threshold, it is possible to determine whether the object detection model is able to consistently detect object information. The consistency can be measured immediately on the inference device side, without the need for data annotation. In this study, the consistency of inference results was measured for YOLOv5, a specialized object detection model for maritime objects, and confirmed that object detection model consistency improved through continual retraining. This is expected to lead to the construction of more reliable object detection systems and to improvements in the safety and reliability of object detection for moving objects.

This section explicitly addresses the issue of consistency in object detection from time-series images. When images are similar to the human eye, object detection models should consistently detect the same objects. However, due to object changes caused by movement, camera vibrations, and environmental changes such as weather and light conditions, the conventional consistency measurement method [53]

- These methods have been developed mainly for static surveillance cameras, and it is difficult to accurately measure the inference results of object detection in moving objects due to changes in the image content. However, object detection in moving objects requires adaptation to constantly changing environments.
- In the environment of mobile objects, there is a continuity of the collected data, and in most situations, annotation is not possible. Common consistency measurement methods require reference to a pre-prepared dataset and cannot measure the consistency of consecutive images when inferring object detection models.

To overcome these issues, it is necessary to develop a method for measuring the consistency of time-series images that takes into account the characteristics of object detection in moving objects.

5.1.2 Scenario: Maritime Object Detection

As shown in the Fig. 5.1, a scenario is assumed for the purpose of maritime navigation assistance and environmental monitoring, which is an example of mobile object detection. Before a maritime surveillance application is launched, a maritime object detection model can be built using maritime datasets from a pre-trained model repository such as PyTorch Hub. Vessels can also collect time-series images from different sea areas with different shooting conditions, such as various weather conditions in each sea area, and utilize them to build a maritime dataset while being parsed into an inference model for object detection. During the voyage, the time-series image data must be similar and the inference results must be accurate and consistent. To improve the accuracy of object detection, it is essential to quickly detect inconsistent inference results obtained in the object detection model and to continuously improve the accuracy of the model.

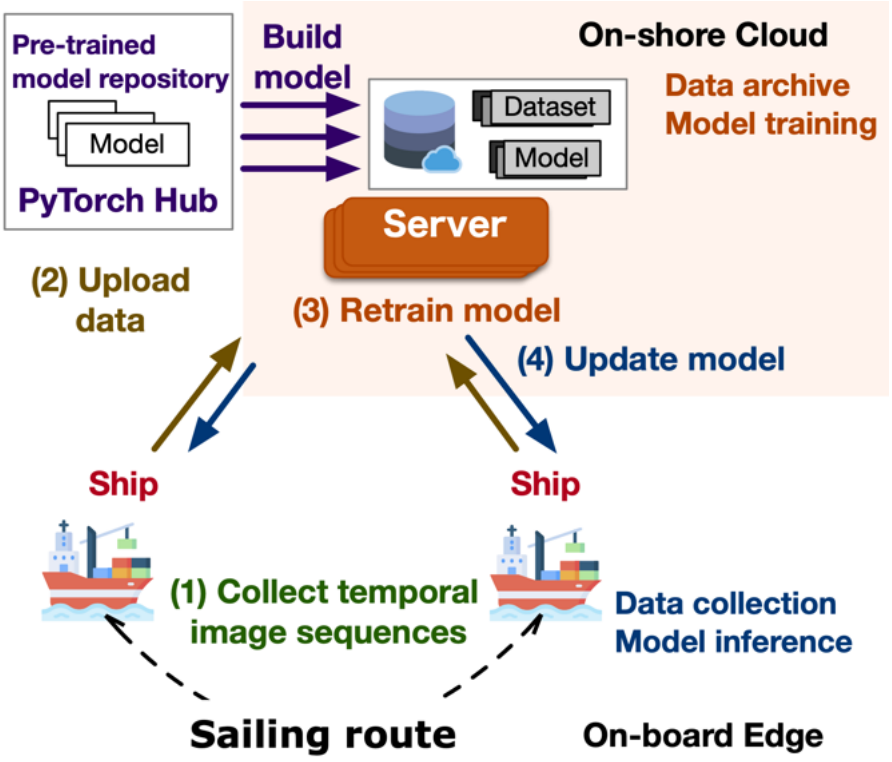


FIGURE 5.1: Scenario of Maritime Object Detection on Time-series Images.

To construct the object detection system described above, the prototype system [120] developed by the authors in a previous study is used. The system architecture includes an IoT device and an edge server, which are assumed to be installed on board, and a cloud server located on land. Specifically, data is collected from onboard cameras, object detection is performed on the onboard edge server, and the results are provided

for maritime surveillance. On the cloud, data collected from the ship's camera sensors are used to build a new training dataset. Continual retraining of the maritime object-specific object detection models is accomplished by leveraging the high computational performance of the cloud's GPUs. Retraining models can be downloaded to an edge server on the vessel to update the inference models in use, thereby improving the object detection performance for this particular voyage route.

5.1.3 Consistency Evaluation

In this part, the author addresses the issues described in Section 5.1.1 and describes a proposed method for evaluating the consistency of object detection by introducing a configurable threshold that takes into account image similarity in time-series images. Time-series images taken from moving objects vary in the degree to which the similarity between adjacent before and after images varies depending on the terminal attached to the camera and the movement of the subject. The conventional consistency measure [53] is designed for fixed cameras and is susceptible to natural variations in image content.

This method measures the image similarity of time-series images and introduces a configurable threshold based on the IoU index, which indicates the overlap of inference results between the previous and following images. In this dissertation, consistency is defined as the difference in predictions obtained from object detection models between similar images. This threshold value enables more accurate consistency evaluation by flexibly responding to variations in time-series images. In addition, this method utilizes the Non-Maximum Suppression algorithm [44] to measure inference results quickly and autonomously without using ground truth. This allows the consistency of inference results to be measured at inference terminals installed on mobile devices and is expected to improve the efficiency of measuring the consistency of inference results.

5.1.3.1 Measurement for Consistency Evaluation

In this part, the author illustrates the proposed consistency measure with an example of consistency between temporally consecutive images i and j. These consistency results $c_{i,j}$, where p_i and p_j represent the set of inference results for image i and image j, respectively, are obtained as in equation 5.1 $|p_i \cap p_j|$ denotes the number of instances

of the same class or attribute detected in both image i and image j. $|p_{i,j}|$ indicates the number of elements included in the inference result p_i for image i but not in the inference result p_j for image j. The reverse is also true for $|p_{j,i}|$. In particular, consistency is zero if the common part of p_i and p_j is empty, or if the common part of p_i and p_j is less than the sum of $|p_{i,j}|$ and $|p_{j,i}|$. Thus, the equation 5.1 quantifies the consistency of time-series images and serves as an indicator that allows for the appropriate handling of exceptions.

$$c_{i,j} = \frac{|p_i \cap p_j| - |p_{i,j}| - |p_{j,i}|}{|p_i \cap p_j|} \tag{5.1}$$

For a given time-series image data D with N images, measure the pairwise consistency between each pair of consecutive images in the time-series image and calculate the mean value of consistency by averaging the results over all $N-1$ pairs, as in equation 5.2:

$$C_D = \frac{1}{N-1} \sum_{i=i}^{N-1} c_{i,i+1}. \tag{5.2}$$

5.1.3.2 Evaluation of Image Similarity

This part describes the evaluation of image similarity in the consistency evaluation of object detection from time-series images. Here, the author uses one of the objective evaluation methods of image quality, Structural SIMilarity (SSIM) [122]. In time-series images from cameras installed on moving objects, the objects may move, and as a result, the image content changes. Conventional methods such as Peak Signal Noise Ratio (PSNR) cannot distinguish between the entire image and local differences because of pixel-by-pixel comparisons. Fig. 5.2 shows how SSIM measures the similarity between two images by comparing the luminance, contrast, and structure of the images. In the case of images collected from moving objects, SSIM can provide a better assessment of image similarity because differences across the entire image can reflect object movement rather than local differences. SSIM is also computationally less expensive than the method [123] that extracts low-level features of similar images and is easier to apply to large time-series images.

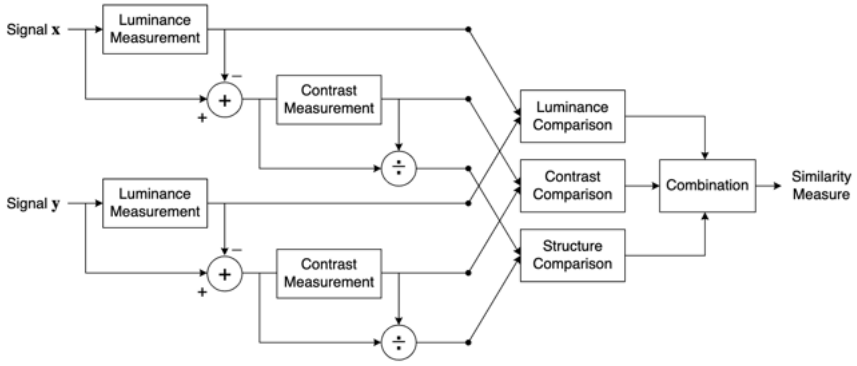


FIGURE 5.2: Diagram of the Structural SIMilarity (SSIM) Measurement.

5.1.3.3 Thresholding for Consistency Evaluation

This part describes the introduction of a threshold value that can be set for consistency evaluation metrics. In order to flexibly respond to changes in the target from a fixed camera to a moving camera, consistency is evaluated by tuning the image similarity and the tolerance of the IoU. To compute consistency, the maximum IoU is calculated through a non-maximum suppression algorithm for identical objects in the before and after images. The author also shows the consistency condition of the threshold of IoU leading to the image similarity $SSIM_{i,j}$ between image i and image j, as in equation 5.3. where τ is a pre-defined threshold value and the general threshold value $\tau = 0.5$ was used.

$$IoU > \tau * SSIM_{i,j} \tag{5.3}$$

In images from fixed cameras, the image content is relatively stable and the position and size of objects are often consistent. However, in images from a moving camera, the image content is not stable and the position and size of objects may fluctuate due to camera shaking or movement. Therefore, the introduction of a configurable IoU threshold enables appropriate consistency evaluation even for images from moving cameras. When image similarity is high, there is a high likelihood that appropriate object detection will occur even with a high IoU threshold. On the other hand, when image similarity is low, the IoU threshold should be set low to ensure consistency in object detection. Thus, by tuning the IoU threshold to the image similarity, the consistency of object detection can be properly evaluated.

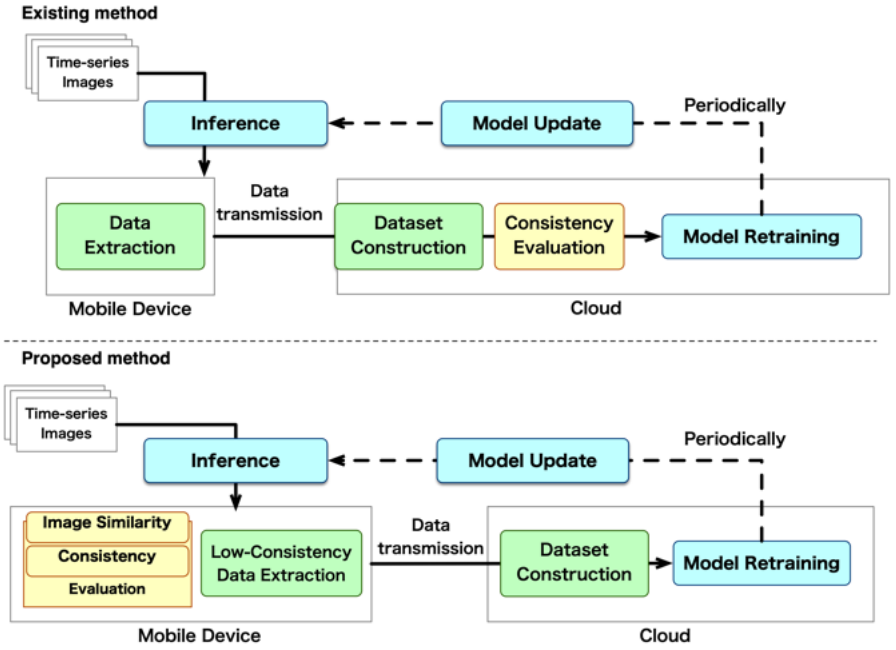


FIGURE 5.3: Consistency Evaluation Processing of Training/Retraining Models.

5.1.3.4 Evaluating Consistency of Training/Retraining Models

This part describes the implementation of measuring model consistency and training/retraining of object detection models based on the proposed consistency evaluation method. The method adds and adjusts the training layer of the model for training on newly collected data. The comparison of the consistency evaluation procedures of the proposed method and existing methods is presented in Fig. 5.3. The proposed method is unique in that the consistency measurement is performed immediately after the inference at the mobile device. On the other hand, the conventional method starts with inference at the mobile device, extracts data, transmits the data to the cloud, builds a dataset in the cloud, measures consistency, and retrains the model. In the proposed method, the consistency measurement combined with the image similarity measurement is performed first, following inference at the mobile device. This difference in procedure allows for early detection of consistency problems in the inference process and in data processing since the consistency measurement is performed first on the moving object. It also allows for more efficient data transmission and data set construction by extracting data of high value for use in new data sets for retraining the model.

In addition to the existing random extraction and low-confidence data extraction, the proposed low-consistency data extraction method is used to extract the collected data.

Existing random extraction constructs an unbiased dataset by randomly sampling from the collected data. Low-confidence data extraction, on the other hand, aims to improve the performance of inference models by extracting data in which object detection models detect objects with low confidence. The proposed low-consistency data extraction aims to improve the consistency of the model. The experiment in Section 6.4 will also verify the effectiveness of the consistency improvement by comparing the accuracy of the new model using the extraction of low-consistency training data with the new model retraining randomly extracted training data as a baseline.

5.2 Object Detection Model Optimization

In this section, the author introduces an optimization method for object detection models, focusing on improving model update and retraining efficiency in maritime mobile systems with constrained computational resources.

5.2.1 Scenario: Maritime Surveillance System for Dynamic Environments

In this part, a scenario is assumed for maritime surveillance during a voyage between two sea areas, which is an example of mobile object detection. Fig. 5.4 depicts a distributed object detection system for maritime surveillance, focusing on addressing challenges such as accuracy degradation caused by dynamic navigation plans, unpredictable weather conditions, and communication constraints. The bottom of the figure illustrates the dynamic maritime environment, where ships follow scheduled or flexible itineraries while navigating between ports under varying weather conditions. The offline nature of the onboard edge servers allows them to collect and process data autonomously during voyages. As ships arrive at ports or encounter other ships, they can reconnect to exchange data and models, integrating their localized training into ships. There are two typical voyage patterns that can be grouped by location and time:

1. **Scheduled Shuttle** route around *Port A* or *Port B* in each sea area, which involves a regular, predictable schedule around a specific sea area.

2. **Flexible Itinerary** across sea areas, allowing ships to adapt their routes based on situational needs and long-distance shipping.

The system relies on two key roles: onboard edge servers located on ships and on-shore cloud servers. Each component plays a distinct role in ensuring the robustness and efficiency of the system. Onboard edge servers, deployed on ships, are responsible for collecting data during voyages, performing local inference for object detection, and conducting local model training when necessary. These servers are designed to operate under offline conditions, allowing them to function even when disconnected from the central cloud infrastructure. Their primary responsibility is to process data in real-time, adapt models to local environmental conditions, and exchange information with other edge servers to enhance collaborative performance. This decentralized approach ensures that object detection remains accurate despite limited communication or varying environmental factors. The on-shore cloud servers, located at ports or centralized facilities, serve as the hub for data archiving, remote model training, and overall model management. These servers are equipped with significant computational resources, enabling them to retrain models with large datasets collected from multiple edge servers. They provide a centralized repository for updated models and datasets, ensuring consistency and high performance across all deployed edge devices. The cloud servers facilitate model updates by allowing edge servers to upload locally trained models and download optimized global models. The interaction between onboard edge servers and the cloud servers is dynamic and adaptive, depending on factors such as communication conditions, geographic location, and weather conditions. Ships equipped with edge servers join or leave model and information-sharing groups as their operational context changes. These ships form different groups named *Group A* or *Group B* in Fig. 5.4, based on their proximity, voyage schedules, and detected object distributions, enabling targeted data exchange and model optimization. This collaborative interaction enables ships to share insights about local conditions and adapt their models accordingly.

To better improve the object detection performance in maritime surveillance, object detection model retraining is executed both locally on edge servers and remotely on cloud servers. Edge servers perform lightweight local training to quickly adapt to environmental changes, such as varying lighting or weather conditions, without relying on consistent connectivity. In contrast, cloud servers handle comprehensive model retraining by aggregating data from multiple edge devices, thereby improving the global

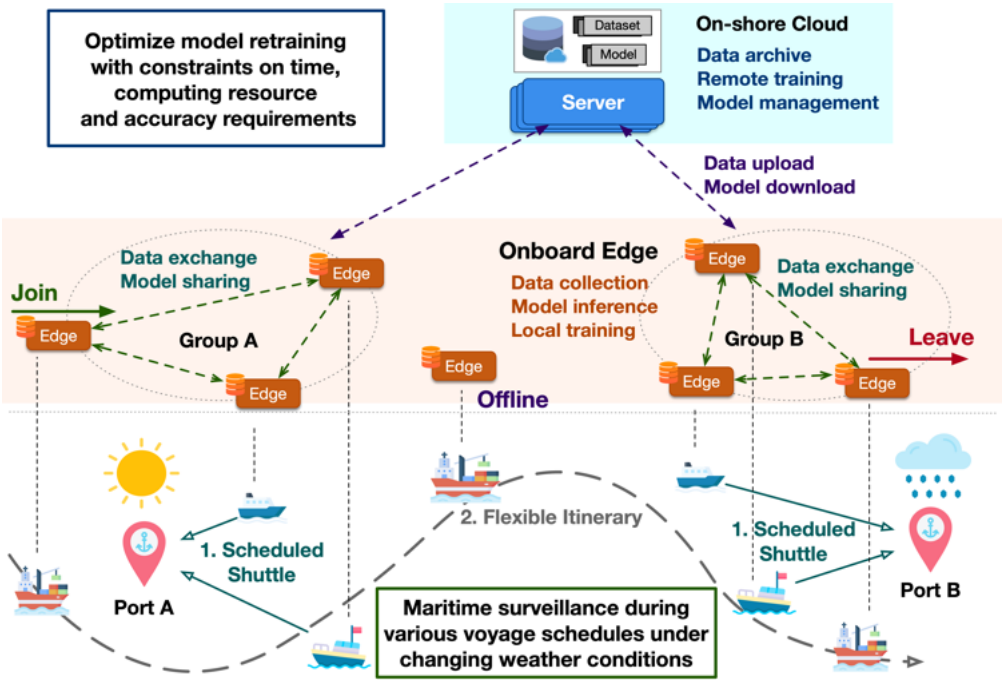


FIGURE 5.4: Scenario of Maritime Surveillance System for Dynamic Environments.

model's robustness and accuracy. The retraining approach ensures that models remain effective in both localized and generalized scenarios. The kind of system leverages a collaborative architecture between onboard edge servers and on-shore cloud servers, dynamically adapting to operational constraints and environmental changes. By combining decentralized local training with centralized global retraining, the system achieves robust object detection performance, ensuring reliable maritime surveillance across diverse and challenging conditions.

5.2.2 System Architecture

Fig. 5.5 presents the system architecture of a maritime mobile object detection system, focusing on optimizing model retraining and operational efficiency under constrained resources. The system comprises five core modules, each handling critical functions to ensure effective and adaptive object detection capabilities as follows:

Metadata Management and Trigger Configuration: These modules deal with setting up triggers for model retraining and managing metadata associated with the behaviors of devices and models.

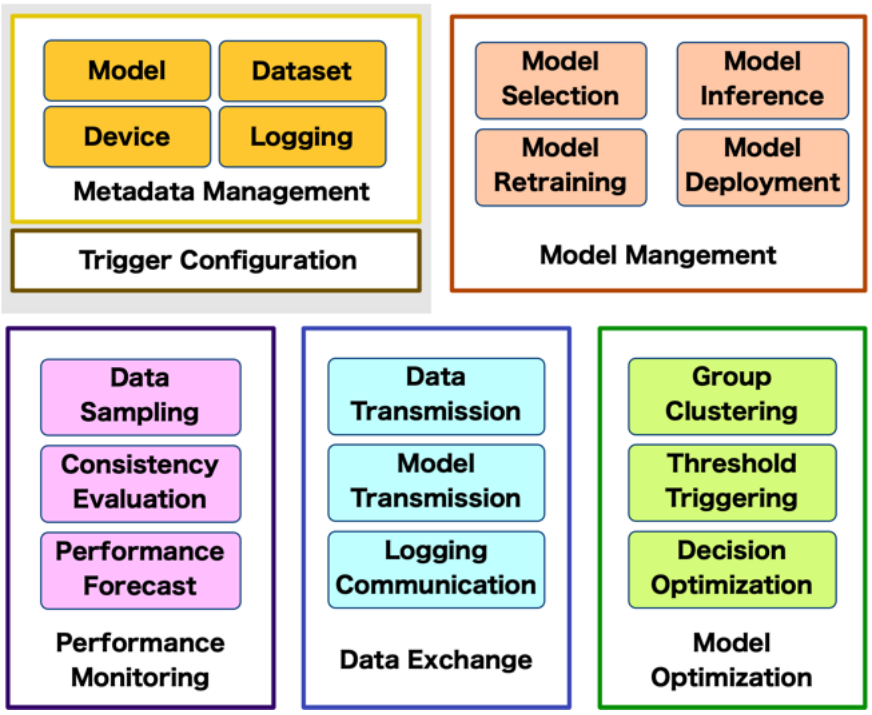


FIGURE 5.5: Architecture of Maritime Object Detection System.

- *Metadata Management* component stores and manages metadata related to the devices and models, such as training data used, model version, and performance metrics. It also manages the datasets used for training and evaluating object detection models including data labeling, and data augmentation. Besides, it handles the interaction with the hardware devices on the mobile platform and model selection and retraining logs relevant to monitoring and optimization.

- *Trigger Configuration* component pre-defines the thresholds and parameters of trigger conditions that initiate model retraining. These triggers could be based on factors like model performance degradation, the availability of new data, or changes in the operational environment.

Model Management: These modules focus on the core aspects of model lifecycle management.

- *Model Selection* component is responsible for choosing the most suitable object detection model from a pool of model candidates based on factors like performance, and computing constraints.

- *Model Retraining* component facilitates the retraining of models to adapt to performance degradation or operational conditions. It involves transfer learning or fine-tuning existing models based on model selection.
- *Model Inference* component handles the execution of the selected or deployed model for object detection on the maritime platform.
- *Model Deployment* component manages the deployment of selected or retrained models to the maritime systems, ensuring model update and continuing model inference

Performance Monitoring: These modules continuously monitor the performance of the deployed object detection model on different mobile devices.

- *Data Sampling* component handles the sampling of data from cameras on maritime mobile devices like ships. It determines the sampling rate and strategies for efficient data acquisition according to image contents such as detected objects, and similarity of images.
- *Consistency Evaluation* component assesses the consistency of the inference results, identifying inconsistencies of time-series images reflecting potential errors.
- *Performance Forecast* component predicts the future performance of the deployed object detection model based on current trends and data. It assists in making informed decisions about model retraining and optimization.

Data Exchange: These modules facilitate data communication in maritime mobile systems, enabling collaborative data analysis and model improvement.

- *Data Transmission* component handles the transmission of data among the maritime mobile devices, or between maritime mobile devices and the shore-side cloud servers. It supports data sampling exchange for performance evaluation, and dataset sharing for remote model retraining.
- *Model Transmission* component manages the transmission of updated models to the maritime mobile systems. It supports selected models and retrained model deployment according to optimization decisions.

- *Logging Communication* component handles communication related to logging data, and supporting information for optimization decision-making.

Decision Optimization: These modules, the core of the proposed optimization method, focus on optimizing decision-making based on the object detection results.

- *Group Clustering* component groups mobile devices into clusters based on their proximity of physical locations, voyage schedule, and detected object distributions. This can be helpful for narrowing the range of data exchange and model optimization.
- *Threshold Triggering* component executes model retraining triggering with predefined thresholds based on the object detection results. It triggers different levels of performance decay alerts to launch model selection or retraining processes.
- *Decision Optimization* component makes decision-making processes based on the object detection results and other relevant information. This may involve using techniques like rule-based systems or machine learning algorithms.

The system architecture aims to create a robust and adaptive system for object detection in challenging maritime environments. By continuously monitoring model performance, retraining models when necessary, and optimizing decision-making, the system enhances the performance and efficiency of object detection for maritime mobile systems.

The monitored performance metrics are processed in *Group Clustering*, where models are grouped based on their routes or common characteristics, allowing for targeted management of similar models, particularly useful in geographically distributed scenarios. *Threshold Triggering* then determines whether specific performance thresholds are met for each group. The processing flow of object detection model optimization follows structured components, as illustrated in Fig. 5.6. The process begins with object detection initiation, where the object detection model performs model inference on input data. During inference, performance monitoring modules track key metrics related to model performance, latency, and resource usage. Following this, the monitored performance metrics are processed in *Group Clustering*, where models are grouped based on their routes or common characteristics. This allows targeted management of similar models, especially useful in geographically distributed systems.

If a trigger is activated, indicating that thresholds are unmet, the flow initiates trigger activation, looping back to inference with adjustments to meet performance targets. If the trigger is not activated, trigger deactivation proceeds, moving to model retrieval to identify if an existing model variant meets the required thresholds. If an appropriate model is found, model selection is conducted to deploy it back into inference, completing a feedback loop for optimizing current detection requirements without retraining. In cases where no suitable model variant exists, the system advances to a retraining optimization decision, where an assessment of retraining costs and the potential performance improvement is made. If retraining is deemed beneficial, the model retraining process is initiated. Once retraining is complete, the newly optimized model is reintroduced into inference, thereby enhancing performance and updating models for following inference. This cyclical framework ensures the dynamic adaptation of object detection models in response to changing performance demands, enabling sustained performance while managing computational resources effectively.

5.2.3 Problem Statement

In this part, the author defines a couple of `Trigger` or `Keep` decisions and formalizes the optimization problem to be addressed. From the time $t = 0$ to $t = T$, the model performance is monitored at regular intervals. At each time t, the model performance is evaluated based on current and historical inference performance results. The *Threshold Triggering* makes a `Trigger` decision when arbitrary model retraining trigger conditions are satisfied, and a `Keep` decision otherwise. A `Trigger` decision initiates the model retraining process, while a `Keep` decision maintains the current inference model without retraining or selecting.

For a model optimization, three phases are assumed: model retraining triggering, model selection, and model retraining optimization. In the first phase, the model retraining triggering phase, the conditions are defined that trigger model retraining and assign urgency levels to prioritize these tasks based on the current performance of the object detection model in 5.2.3.1. In the second phase, the model selection phase, the model selection process is proposed to prioritize model reuse and minimize retraining costs when a threshold trigger is activated in 5.2.3.2. In the third phase, the model retraining

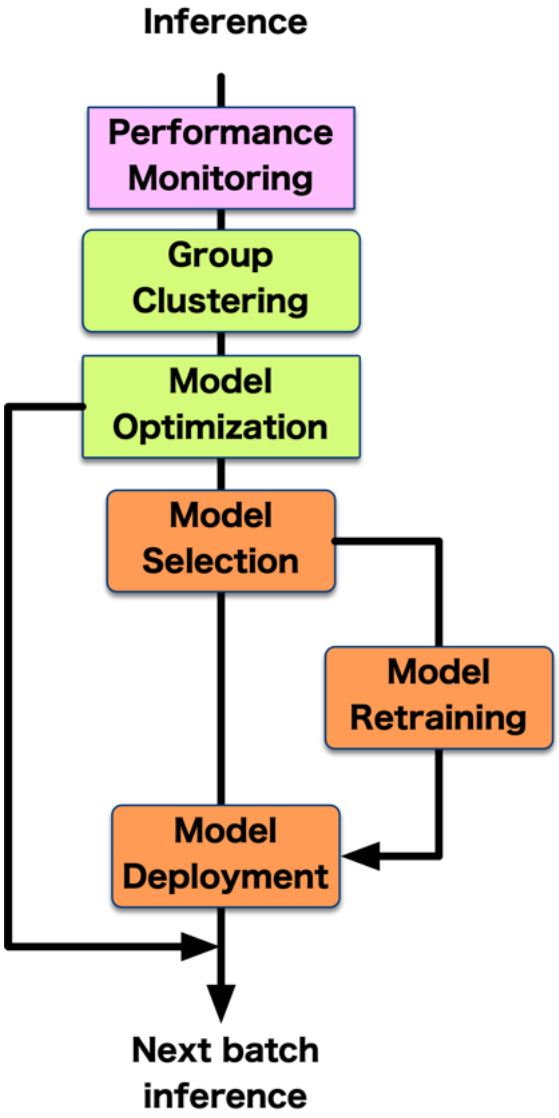


FIGURE 5.6: Processing of Object Detection Model Optimization.

optimization phase, three optimization plans are proposed to optimize the object detection model retraining process after model retraining triggers are activated in 5.2.3.3. For solving the object detection model optimization problem, an optimization problem is illustrated based on Mixed-integer linear programming (MILP). The MILP-based optimization problem is designed to optimize the object detection model retraining process by considering the trigger conditions, urgency levels, and optimization plans.

5.2.3.1 Model Retraining Trigger

In this part, the model retraining conditions are defined that trigger model retraining and assign urgency levels to prioritize these tasks based on the current performance of

the object detection model. The trigger conditions directly reflect the model's performance and the need for retraining. While the urgency levels of triggers represent the importance of retraining actions relative to the performance and corresponding performance thresholds. These conditions and their associated urgency levels are outlined as follows:

0. **No Trigger**: No retraining is necessary as the model's performance is within acceptable bounds and any following triggers are not activated.

1. **Precautionary Trigger**: This type of trigger is used to prevent performance degradation in advance before it falls below a higher expected threshold. Although recent performance is declining, it remains above a critical performance threshold when the trigger is activated. The precautionary trigger decision function can be defined as:

$$D(t, \tau_{extra}) = \begin{cases} \text{Trigger} & A^{est}_{d,m,t'} > \max t' \in [t+1, t+\tau_{extra}](\tau_{perf}(t')) \\ \text{Keep} & \text{otherwise} \end{cases} \tag{5.4}$$

   Where:

   - τ_{extra} is the precautionary trigger extra period.
   - t' is a time within the range from $t+1$ to $t+\tau_{extra}$.
   - $\tau_{perf}(t')$ represents the performance threshold function over time t'.
   - $A^{est}_{d,m,t'}$ represents the estimated performance based on a performance decay model incorporating a logistic function. This formulation captures the natural performance decline over time, assuming that the impact of retraining is most pronounced immediately after deployment and gradually diminishes. Here, the logistic decay function to fit model the gradual decline in model performance is given by:

$$A^{est}_{d,m,t'} = A_{d,m,t_{last\_retrain}} - \frac{A_{d,m,t_{last\_retrain}} - A_{d,m,\infty}}{1 + e^{-k(t-t_0)}}, \tag{5.5}$$

where $A_{d,m,t_{last\_retrain}}$ is the initial performance at the last retraining time, A_∞ is the empirically lower bound of performance, k is the decay rate, controlling how fast the performance declines,t_0 is the inflection point where the decline accelerates. These parameters are decided on recent model performance historical data.

The trigger condition is based on the pre-defined *precautionary trigger extra period* τ_{extra}, which defines the searching range of a potentially higher expected performance threshold. When the trigger is activated, retraining is initiated to prevent further degradation.

2. **Periodic Trigger**: This type of trigger is used to resist model staleness along with gradual performance degradation, as the lowest priority trigger. The periodic trigger decision function can be defined as follows:

$$D(t, \tau_{interval}) = \begin{cases} \text{Trigger} & t - t_{last\_retrain} > \tau_{interval} \\ \text{Keep} & \text{otherwise} \end{cases} \tag{5.6}$$

Where:

- t is the current time.
- $t_{last\_retrain}$ is the last model retraining time.
- $\tau_{interval}$ is the pre-defined retraining interval.

The trigger condition is based on the pre-defined *retraining interval*, which is determined based on the expected model performance degradation rate. Periodic retraining is scheduled at regular intervals, irrespective of the current performance metrics. Each trigger testing compares the latest model retraining time with the current time to determine if the *retraining interval* has been reached. When the trigger is activated, retraining is initiated to prevent model staleness.

3. **Cumulative Trigger**: There is a continuous decline in performance, with the model nearing a high threshold. This condition indicates a persistent issue requiring retraining to address long-term precision degradation. The trigger condition is based on the pre-defined *performance decline threshold*, which defines the cumulative performance decline threshold. When the trigger is activated, retraining

is initiated to address the long-term precision decline. The cumulative trigger decision function can be defined as:

$$D(t, \tau_{cumulative}) = \begin{cases} \text{Trigger} & A_{d,m,t_{last\_retrain}} - A_{d,m,t} > \tau_{cumulative} \\ \text{Keep} & \text{otherwise} \end{cases} \tag{5.7}$$

Where:

- $A_{d,m,t_{last\_retrain}}$ is the model's performance at the last retraining time.
- $A_{d,m,t}$ is the model's current performance.
- $\tau_{cumulative}$ is the cumulative performance decline threshold.

4. **Critical Trigger**: This type of trigger is used to address severe performance degradation. The trigger condition is based on the model's performance falling below a critical *performance threshold*, necessitating immediate retraining to restore acceptable performance levels. When the trigger as the top priority is activated, retraining is initiated to address the severe precision decline. The critical trigger decision function with performance threshold can be defined as:

$$D(t, \tau_{critical}) = \begin{cases} \text{Trigger} & A_{d,m,t} < \tau_{critical} \\ \text{Keep} & \text{otherwise} \end{cases} \tag{5.8}$$

Where:

- $A_{d,m,t}$ is the model's current performance.
- $\tau_{critical}$ is the critical performance threshold.

Each time the model performance is monitored, the model retraining conditions are matched with the above parameters pre-defined by the user. Higher urgency levels are assigned to triggers that are more critical and require immediate attention. Since several arbitrary trigger conditions on one same inference model may be activated simultaneously, the urgency levels are used to prioritize the retraining tasks, i.e. the model retraining task with the highest urgency level is selected for execution.

5.2.3.2 Model Selection

The model selection process in the optimization framework is designed to prioritize model reuse and minimize retraining costs when a threshold trigger is activated. The procedure involves the following steps:

1. **Threshold Trigger and Model Search:** Upon activation of a threshold trigger, the system first searches for alternative models within the same group before initiating model retraining. This step aims to identify any existing models that may satisfy current performance requirements.

2. **Model Evaluation Threshold:** During the search, available models are evaluated sequentially based on specific conditions to determine their suitability:

   - **Condition 1:** If an alternative model meets the current performance threshold, it is selected and deployed immediately in place of the current model, effectively switching to a higher-performing model without retraining.
   - **Condition 2:** If an alternative model does not fully meet the performance threshold, but demonstrates greater performance than the current model, the system initiates retraining using this higher-performing model as a base to further enhance performance.
   - **Condition 3:** If an alternative model shows lower performance than the current model, it is disregarded, and retraining is performed using the current model instead.

3. **Retraining Decision:** After evaluating available models, the system selects the best-suited model for inference. If no models meet the performance requirements or provide improvement over the current model, retraining is carried out on the current model to ensure the desired performance level is achieved.

Fig. 5.7 illustrates an example for the scope of model selection after group clustering (in Section 5.2.4.1) for mobile devices in a maritime environment, where devices store previously trained models locally for reuse. The devices are grouped into *Group A* and *Group B* based on their stored models and relevance to current tasks. In *Group A*, three devices store multiple versions of relevant models, enabling them to serve as potential

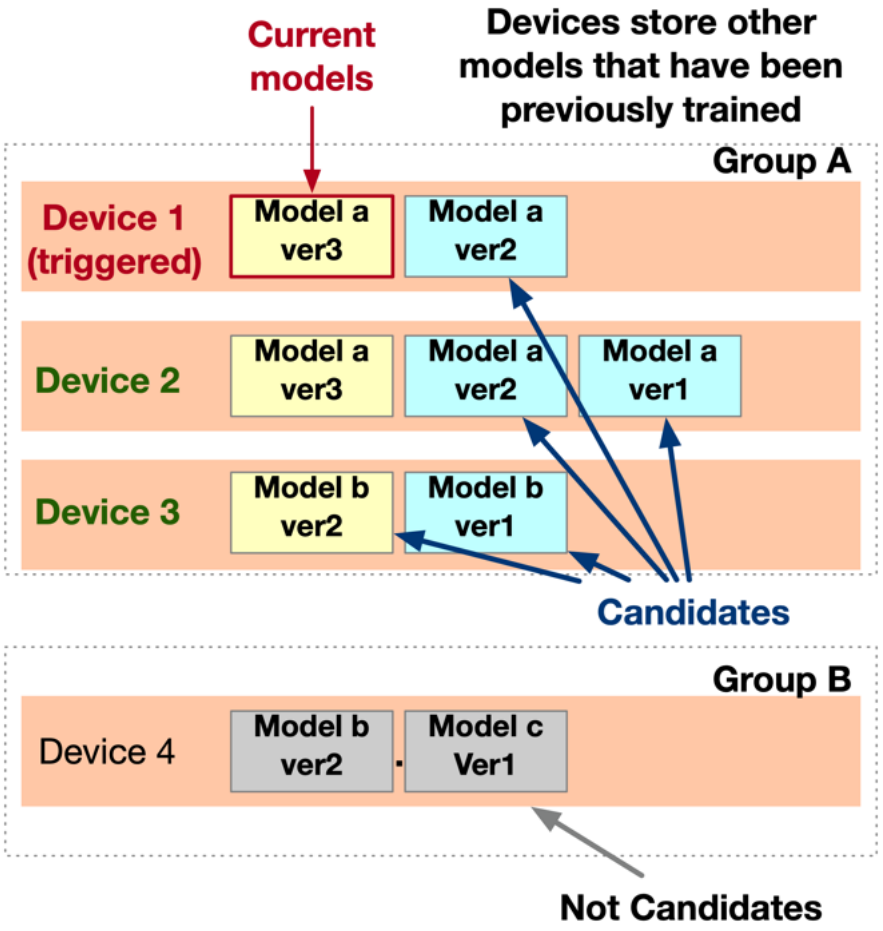


FIGURE 5.7: Scope Example of Model Selection.

candidates for selection during retraining or inference tasks. For example, *Device 1* triggered for model update currently uses *Model a ver3*, but can reference older versions of the same model *Model a ver2* and *Model a ver1* for efficient fine-tuning or rollback. Similarly, *Device 2* also has *Model a ver3* and earlier versions, making it a candidate for sharing model parameters. *Device 3*, storing models of a different type, provides potential resources for other devices requiring similar models. In contrast, *Group B* contains models irrelevant to the triggered task, such as *Model c ver1*, making them **non-candidates** for selection. This group clustering process ensures computational efficiency by narrowing model selection to relevant devices, minimizing unnecessary data transfer and resource usage. The system prioritizes model reuse to streamline updates and optimize resource-constrained environments. This decision-making process ensures that only models offering a clear performance benefit are deployed, thereby optimizing model performance and computational resource usage.

5.2.3.3 Optimization Plans

To support the subsequent optimization plans, the following terms and notation are introduced, defining core elements used within the optimization problem:

- Time: Represents a time duration with the same interval to periodically monitor the model performance and trigger retraining actions, and the time is defined mathematically as t.
- Model: Represents an object detection detector, characterized by its model parameters, training dataset, testing performance, and estimated time and GPU memory consumption when retraining. Each model is defined as m, with the current model state prior to retraining denoted as $m_{current}$, and the selected or optimized model post-retraining represented as m^*.
- Device: Denotes the computational environment (e.g., edge servers or mobile devices) on which object detection models are deployed, with defined computational capabilities and resource constraints.

In this part, three optimization plans are proposed to optimize the object detection model retraining process after model retraining triggers are activated. All optimization plans are designed to optimize the model retraining process by considering the trigger conditions, urgency levels, and the current performance of the object detection model. Three optimization plan alternatives are provided for satisfying three different requirements: time efficiency, resource effectiveness, and performance improvement. The three optimization plan alternatives include **Time Minimization Plan**, **Resource Minimization Plan**, and **Performance Improvement Plan**.

Time Minimization Plan is designed to minimize the model retraining time by selecting the fastest retraining model on the fastest retraining nodes. The plan aims to minimize the total retraining time by selecting the fastest retraining model on the fastest retraining nodes, ensuring rapid adaptation of the object detection model. This plan is particularly suitable for scenarios requiring immediate model updates, where maintaining operational continuity is crucial.

Resource Minimization Plan is designed to minimize resource consumption by selecting the most resource-effective retraining model on the most resource-effective retraining

nodes. Concretely, the resource of one-time retraining is defined as the product of the retraining time and the retraining consumption of GPU memory. The resource can be calculated as follows:

$$R = T_{\text{retrain}} \times \text{GPU Memory} \tag{5.9}$$

where R is the resource of one-time retraining, whose unit is GBhr, $T_{\text{Retraining}}$ is the retraining time, and GPU Memory is the retraining consumption of GPU memory.

The plan aims to minimize computational resource usage by selecting the most resource-effective retraining model on the most resource-effective retraining nodes, optimizing retraining to conserve device resources. This plan is designed for resource-constrained environments where computational or memory limitations restrict frequent or resource-intensive retraining.

Performance Improvement Plan is designed to minimize the expected model performance degradation by selecting the best retraining action. It predicts all potentially available retraining models and nodes, and estimates their model performance improvement. The plan aims to maximize the model performance improvement by selecting the best retraining action, while it empirically requires more time or resources than the other two plans.

5.2.3.4 Optimization Problems

This part presents the optimization problems of maritime mobile object detection. The system consists of a network of interconnected nodes, including mobile devices and cloud servers, that cooperate to detect and track maritime objects. The mobile devices, such as cameras and sensors mounted on ships, capture time-series image data. This data is either processed locally on the edge devices or transmitted to centralized cloud servers for further computation. D is the set of mobile devices, C is the set of cloud servers, and N is the set of all nodes including all devices and servers. G is defined as the set of groups, divided by *Group Clustering*, D_g and C_g mean the mobile device and cloud server sets divided into group g. The existing inference model set is denoted as M, especially, m_d means the current inference model of mobile device d. The detailed

notations and definitions are listed in Table 5.1. Furthermore, a binary decision variable $e(d, n, m, t)$ is defined to indicate the execution of model m on node n including any device and cloud server at time t. It can be defined as:

$$e(d, n, m, t) = \begin{cases} 1, & \text{if model } m \text{ is executed on node } n \text{ by device } d \text{ at time } t \\ 0, & \text{Otherwise} \end{cases} \tag{5.10}$$

These decisions can be integrated into $q(d, m, t)$, defined to indicate the decision of model selection and retraining on inference model m of mobile device d at time t. The selection decision $s(d, m, d', m', t)$ is defined to indicate whether device d selects another model m' from device d' at time t. Especially, device d can be the same as device d', which means the model is selected from the same device.

It can be defined as:

$$q(d, m, t) \in \{s(d, m, d', m', t), e(d, n, m, t)\} \tag{5.11}$$

The result generated by decision $q(d, m, t)$ are denoted as $R(q(d, m, t))$, which indicates the selected or retrained model from model m on device d at time t.

The accuracy of model m on device d at time t is denoted as $A_{d,m,t}$, which consists of confidence score and consistency, which can be calculated as:

$$A_{d,m,t} = \beta_{\text{conf}}\text{Confidence Score}_{d,m,t} + \beta_{\text{cons}}\text{Consistency}_{d,m,t} \tag{5.12}$$

where confidence score measures the prediction confidence from model m, and consistency measures the stability of the model's predictions from model m on device d at time t. β_{conf} and β_{cons} are the respective weights, set to 0.5 each in the experiment. $\text{Confidence Score}_{d,m,t}$ and $\text{Consistency}_{d,m,t}$ are normalized to the range from 0 to 1.

The proposed model retraining trigger algorithm is designed for optimizing model retraining in maritime mobile object detection systems. The algorithm presented in Algorithm 3, starts by clustering devices into groups based on certain characteristics using the Group Clustering function. For each group, the algorithm monitors the performance

TABLE 5.1: Notations and Definitions.

Notations	**Definitions**
d, D	mobile device index, set
c, C	cloud server index, set
n, N	node index, set
m, M	inference model index, set
g, G	group index, set
t, T	current time, final time
$q(d, m, t)$	decision of model selection and retraining
$e(d, n, m, t)$	whether retrain model m of device d by node n at time t
$s(d, m, d', m', t)$	whether select model m' of device d' for device d at time t
$R(q(d, m, t))$	results, selected or retrained model from model m on device d at time t
T_m^d	training time of model m on a device
T_m^c	training time of model m on a cloud server
$\tau_{n,t}^C$	available computing resource of node n at time t
$\tau_{d,t}^A$	minimum accuracy threshold of device d at time t
$C_{m,n,t}$	computing resource usage for model m on node n at time t
$p_{d,m,t}$	trigger priority value for model m on device d at time t
$A_{d,m,t}$	accuracy of model m on device d at time t
$\bar{A}_{d,m,t}$	average accuracy of model m on device d from start time to time t

of each device using the Performance Monitoring function. Whether or not a device triggers a retraining condition is determined by the Retraining Trigger function, the result of which is indicated by a retraining trigger $trigger_d$ with a non-negative integer. The triggered device gets a new optimal model m_d^* and a retraining trigger $trigger_{m_d^*}$ by the optimal model m_d^* is selected for that device through the Model Selection function. Once the models for the devices in a group are updated, the group undergoes retraining optimization using the Retraining Optimization function, and the results R_g are collected. The final output R includes the retraining optimization results for all device groups.

The optimization problem of **Time Minimization Plan** can be formulated as:

$$\min \quad \sum_{d \in D} \sum_{m \in M} T_m^n \cdot e(d, n, m, t) \tag{5.13}$$

Algorithm 3 Retraining optimization algorithm.

Input: Device Set D, Model Set M, Current Time t
Output: Retraining Optimization Decisions Q
$Q \leftarrow \emptyset$
$G \leftarrow$ Group Clustering(D)
for each group g in G **do**
    for each device d in group g **do**
        $m_d \leftarrow$ model of device d
        $A_{d,m,t} \leftarrow$ Performance Monitoring(m_d)
        $p_{d,m,t} \leftarrow$ Retraining Trigger($p_{d,m,t}$)
        if $p_{d,m,t} > 0$ **then**
            $m_d^*, p_{d,m^*,t} \leftarrow$ Model Selection(d)
            model of device $d \leftarrow m_d^*$
    $Q_g \leftarrow$ Retraining Optimization(g)
    Add Q_g to Q

s.t.

$$\forall d, m, e(d, n, m, t) \in \{0, 1\} \tag{5.14a}$$

$$\forall d, d', m, m', s(d, m, d', m', t) \in \{0, 1\} \tag{5.14b}$$

$$\forall d, m, p_{d,m,t} \in \{0, 1, 2, 3, 4\} \tag{5.14c}$$

$$\forall d, m, \bar{A}_{d,m,t} \geq \tau_{d,t}^{A} \tag{5.14d}$$

$$\forall d, m, q(d, m, t) \leq p_{d,m,t} \tag{5.14e}$$

$$\forall n, m, \sum_{d \in D} C_{m,d,t} \cdot e(d, m, n, t) \leq \tau_{n,t}^{C} \tag{5.14f}$$

$$\forall g, m, \sum_{d \in D} \sum_{n \in N} e(d, n, m, t) \leq 1 \tag{5.14g}$$

$$\forall d, m, \sum_{n \in N} e(d, n, m, t) + \sum_{d' \in D} \sum_{m' \in M} s(d, m, d', m', t) = 1 \tag{5.14h}$$

The objective is to minimize the total training time of all models. The term T_m^n represents the training time of model m on node n, and $e(d, n, m, t)$ is a decision variable that indicates whether retraining of model m is triggered on device d at time t. Thus, this objective function minimizes the overall retraining time by determining which retraining actions are most efficient.

Furthermore, these constraints ensure that the retraining decisions are made based on the available computing resources, the accuracy requirements, and the priority levels for retraining, which can be summarized as follows:

- The constraint (5.14a) specifies that a retraining decision $e(d, n, m, t)$ is a binary decision variable, meaning it can only take values of either 0 or 1. Here, 0 indicates no retraining of model m on device d at time t, while 1 indicates that retraining will occur.

- The constraint (5.14b) states that a model selection decision $s(d, m, d', m', t)$ is also a binary decision variable. It represents whether device d selects model m' from device d' at time t. A value of 1 means the selection is made, while 0 means it is not.

- The constraint (5.14c) specifies that the trigger priority value $p_{d,m,t}$ is an integer between 0 and 4. It indicates the priority level for retraining model m on device d at time t. A higher value represents a higher urgency for retraining, useful for scheduling or prioritizing retraining tasks based on various factors, such as data changes or model drift.

- The constraint (5.14d) ensures that the average accuracy ($\bar{A}_{d,m,t}$) of model m on device d at time t must be greater than or equal to a specified threshold (τ_d^A). The average accuracy is calculated from accuracy from the start time to the current time t, which can be defined as:

$$\bar{A}_{d,m,t} = \frac{\sum_{t'=0}^{t} A_{d,m,t'}}{t}.$$

  This condition ensures that the models used meet a minimum accuracy requirement, maintaining the quality of predictions or inferences.

- The constraint (5.14e) ensures that the decision variable $q(d, m, t)$ is less than or equal to the trigger priority value $p_{d,m,t}$. Here, $q(d, m, t)$ represents the decision of model selection $s(d, m, d', m', t)$ or retraining $e(d, n, m, t)$ on inference model m of mobile device d at time t. This constraint enforces that the retraining or selection decisions are made only when the trigger priority value indicates a need for retraining.

- This constraint (5.14f) limits the computing resources used for model m on node n at time t. The term $C_{m,d,t}$ represents computing resources required for model m on device d. The total resource consumption must be less than or equal to the

available capacity ($\tau^C_{n,t}$). It ensures that the retraining or inference tasks do not exceed the available computational resources.

- This constraint (5.14g) enforces that model m is retrained at most once across all nodes and devices in a given group g. This is useful for avoiding duplicate retraining tasks and ensuring that resources are used efficiently.

- This constraint (5.14h) requires that, model retraining $e(d, n, m, t)$ and model selection $s(d, m, d', m', t)$ are mutually exclusive for a given model m on device d at same time. The sum must equal 1, indicating that a device can either retrain or select a model at a given time, ensuring no conflicting decisions are made.

The objective function of **Resource Minimization Plan** can be reformulated as:

$$\min \quad \sum_{d \in D} \sum_{m \in M} C(m, d, t) \cdot T^n_m \cdot e(d, d, m, t) \tag{5.15}$$

where $C(m, d, t)$ is the computing resource usage for model m on device d at time t, and T^n_m is the training time of model m on node n. Similarly, the constraints can be reformulated as the before mentioned constraints.

The objective function of **Performance Improvement Plan** aims to minimize the total number of times across all devices and models where the model accuracy drops below the specified threshold from time $t + 1$ to T. Firstly, a model m^* is defined as the updated version of model m after retraining decision $e(d, n, m, t)$ is executed for model m on node n by device d at time t. If $e(d, n, m, t) = 1$, the model m is retrained, and m^* represents the updated version of model m at future time steps. The next step is to determine the accuracy of the estimated model m^* at time steps after retraining is performed: The accuracy $A_{d,m^*,t'}$ of the retrained model m^* on device d at time $t' > t$. It is influenced by retraining execution at time t and time-dependent performance degradation. Typically, accuracy degrades over time if no further retraining is performed. The accuracy at each future time step is compared against a specified accuracy threshold $\tau^A_{d,t'}$ to determine if the model's performance is acceptable: If the model accuracy drops below the threshold at any time t'. A binary variable $r_{d,m^*,t'}$ can indicate whether the accuracy threshold is met or not, which can be defined such that:

$$r_{d,m^*,t'} = \begin{cases} 1, & \text{if } A_{d,m^*,t'} < \tau^A_{d,t'} \\ 0, & \text{Otherwise} \end{cases} \tag{5.16}$$

After determining the accuracy condition, the next step is to calculate the sum of all occurrences where the accuracy drops below the threshold from $t+1$ to the final time T: It needs to calculate the sum of $r_{d,m^*,t'}$ for all devices $d \in D$, for all future time steps t' from $t+1$ to T. This represents the total number of time steps across all devices where the model accuracy is below the acceptable threshold. The final step is to minimize the total number of situations where the accuracy falls below the threshold. It depends on the retraining decision $e(d, n, m, t)$ at the current time t by executing retraining ($e(d, n, m, t) = 1$), the model accuracy $A_{d,m^*,t'}$ is expected to improve, thereby reducing the number of times $r_{d,m^*,t'} = 1$ in future time steps. Conversely, if retraining is not executed ($e(d, n, m, t) = 0$), the model's accuracy might degrade faster, leading to more occurrences where $r_{d,m^*,t'} = 1$. Therefore, the objective function is formulated as follows:

$$\min \quad \sum_{t'=t+1}^{T} \sum_{d \in D} r_{d,m^*,t'} \tag{5.17}$$

To explicitly incorporate the effect of retraining decisions ($e(d, n, m, t)$), the objective function can be modified by introducing the time-dependent effect of retraining. $\alpha(t', t)$ can be defined as a time-dependent coefficient that represents the effectiveness of retraining over time. It is modeled to gradually decrease as time moves further from the retraining point, reflecting the diminishing improvement of retraining on model accuracy with a decay function with the logistic curve.

$$\alpha(t', t) = A_{\min} + \frac{A_{\max} - A_{\min}}{1 + e^{-k((t'-t)-x_0)}} \tag{5.18}$$

Where $A_{\max}$ is the initial and maximum model accuracy, usually set to the current accuracy, $A_{\min}$ is the minimum improvement of retraining, usually set to the minimum historical accuracy, x_0 is the inflection point, which is the decline time to the middle point between $A_{\max}$ and $A_{\min}$, k is the growth rate to control the decline speed near the inflection point. This formulation reflects that the retraining effect is most significant immediately after retraining and gradually diminishes over time.

Therefore, the updated objective function for the retraining optimization problem can be formulated as:

$$\min \quad \sum_{t'=t+1}^{T} \sum_{d \in D} \sum_{m \in M} \left(r_{d,m,t'} - \alpha(t', t) \cdot e(d, n, m, t) \right) \tag{5.19}$$

Additionally, the following constraint can be added to ensure that the term inside the summation is always greater than or equal to zero:

$$r_{d,m,t'} - \alpha(t', t) \cdot e(d, n, m, t) \geq 0, \quad \forall d \in D, m \in M, t' \in \{t+1, \ldots, T\} \tag{5.20}$$

The constraint 5.20 ensures that the item of object function inside the summation is always greater than or equal to zero. This prevents the optimization from overestimating the impact of retraining, ensuring realistic results.

5.2.4 Implementation

In this part, the author illustrates the implementation of object detection model optimization, including model retraining, group clustering, and dataset construction.

5.2.4.1 Group Clustering

In this part, the author illustrates the group clustering process for maritime mobile devices based on their proximity of physical locations, voyage schedule, and detected object distributions. The group clustering process is implemented before the model optimization process, and influences the ranges and results of model selection and retraining decisions. Group clustering is implemented to limit the search scope for reusable existing detection models, thereby optimizing the computational cost of model selection. Among various clustering techniques, the non-hierarchical K-means method was chosen for its computational efficiency and suitability for non-hierarchical data grouping.

The group clustering process, implemented prior to model optimization, serves as a foundational step that significantly influences the scope and outcomes of both model selection and retraining decisions. Clustering is used strategically to limit the search area for reusable detection models, thereby enhancing efficiency in model selection and

retraining. Among various clustering techniques, the non-hierarchical K-means method was selected for this framework due to its simplicity, scalability, and effectiveness in handling large datasets with minimal computational overhead. K-means is particularly suitable for mobile object clustering as it allows for the clear partitioning of data based on pre-defined centroids, which aligns well with dynamic object detection environments. By clustering objects based on specific characteristics, K-means aids in organizing and prioritizing models according to location, object distribution, and environmental conditions, thereby refining the retraining scope.

Mobile devices are clustered using K-means based on structured data points that represent environmental and object characteristics, including:

- **Sea Area and Data Distribution:** Clusters are defined by the geographical sea area through which mobile entities travel, alongside the distribution of detected object types within this region. This geographic grouping helps tailor models to specific maritime zones, where variations in object presence and density are anticipated.
- **Object Size and Distribution:** The average size of detected objects and the proportion of different object types, as identified through detection processes, are also included as clustering factors. This consideration ensures that models are tailored to the types and scales of objects commonly encountered within each cluster.
- **Object-Free Image Rate:** The percentage of images that contain no detected objects is also factored in, as it reflects sparsity levels in the data, allowing the clustering to prioritize areas with more dense or significant object occurrences.
- **Weather Conditions:** To address environmental variance, the percentage of images associated with different weather conditions (extracted through weather classification) is used to establish clusters. This helps optimize model selection for environmental contexts that frequently impact detection accuracy, such as fog or rain.

This clustering approach effectively reduces the operational search space by grouping mobile entities with similar characteristics, thus enhancing the efficiency of the overall optimization process.

5.2.4.2 Dataset Construction

In this part, the author illustrates the data processing pipeline for constructing a dataset to retrain maritime object detection models. Fig. 5.8 illustrates the data processing pipeline for constructing a dataset to retrain maritime object detection models. The pipeline adopts the pseudo annotation method MixMatch [124], leveraging semi-supervised learning to generate pseudo-labels for unlabeled data and reduce data annotation time and costs while enhancing the model's performance. The process unfolds across four interconnected stages: data augmentation, prediction distribution, consistency regularization, and confidence masking.

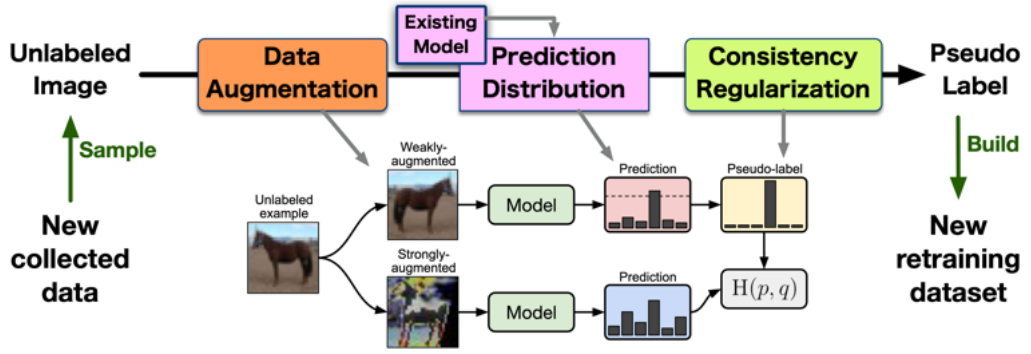


FIGURE 5.8: Processing of Dataset Construction with Pseudo-labeling.

The pipeline begins with **data augmentation**, a crucial step for enriching the diversity of the dataset. Various augmentation techniques are applied to unlabeled images, including transformations such as rotation, cropping, and flipping. These augmentations create multiple views of the same image, enhancing the dataset's variability while preserving the underlying semantic information. By generating diverse variations of the input data, this stage helps the model generalize across different environmental conditions, such as changing weather or lighting during maritime operations.

Next, the augmented data is passed through **prediction distribution**, where an existing pre-trained model is used to generate preliminary predictions for each augmented instance. The output from the model represents a probability distribution across different object classes. These predictions capture the model's current understanding of the data and provide a foundation for pseudo-label generation. This stage leverages the knowledge embedded in the pre-trained model to create an initial annotation framework, even in the absence of ground-truth labels.

The output distributions are then refined through **consistency regularization**, which ensures that the model's predictions are stable and coherent across multiple augmentations of the same image. The predictions from the augmented data are averaged to smooth out noise and stabilize the results. Subsequently, a sharpening operation is applied to enhance the confidence of the dominant class in the distribution, transforming the smoothed output into more decisive pseudo-labels. This process helps to align the predictions with the true underlying data patterns, reducing the uncertainty and improving the reliability of pseudo-labels. The remaining pseudo-labels are used to annotate the dataset, effectively converting unlabeled data into a semi-supervised training set.

5.3 Summary

This chapter presents a comprehensive framework for optimizing object detection models on maritime mobile systems, focusing on enhancing the efficiency of model retraining under resource constraints. The system architecture is designed to support adaptive object detection capabilities through continuous monitoring, model selection, and retraining optimization. This chapter outlines the evaluation method for consistency in time-series object detection results, and the optimization problem formulation for model retraining. Besides, the implementation of model retraining, group clustering, and dataset construction is detailed, providing a structured approach to enhancing object detection performance in maritime environments. This approach enables the system to adapt to dynamic maritime conditions, addressing challenges such as limited labeled data and fluctuating operational environments.

Chapter 6

Experiments and Evaluations

6.1 Overview

In this chapter, the experiments and evaluations of the proposed approaches are presented. The chapter is divided into two parts: the first part introduces weather-aware object detection, and the second part presents the object detection model optimization. Section 6.2 and Section 6.3 describe the experiments of the image quality improvement by the weather noise removal and the object detection performance of the weather-aware object detection method proposed in Chapter 4. Section 6.4 and Section 6.5 present the experiments of the consistency evaluation of time-series image results and the object detection model optimization method described in Chapter 5. The experiments are conducted to validate the effectiveness of the proposed approaches in addressing the challenges of object detection in maritime surveillance systems. The results of the experiments demonstrate the improvements in object detection performance and efficiency achieved by the proposed approaches.

6.2 Weather Noise Removal

In this section, experiments are conducted to demonstrate that the proposed tool adaptively removes noise and contributes to improving the performance of object detection.

6.2.1 Experimental Methodology

This experiment ran the proposed tool on the Amazon EC2 instance type of *g4dn.2xlarge* and input raw images with fixed size into the multi-class weather classification model. Fig. 6.1 shows the sample restored images by the five rain removal algorithms, and their corresponding quantitative evaluation results with synthetic rainy image datasets *Rain100H* [125]. In the above part, it shows an example image with the same original image and its corresponding rain noise removal results by five algorithms. The quantitative evaluation results and average processing time per image are shown in the table below the images, which the green ones in the table represent the best results in each evaluation metric, and the red ones represent the worst results. Concretely, it evaluated their **average processing time** and **result image quality** of the existing rain removal algorithms to find the most suitable rein noise removal algorithm as follows:

- **Progressive ResNet (PReNet):** It provides a better and simpler baseline rain removal network [28] by considering network architecture, input and output, and loss functions. Specifically, it proposed to take advantage of recursive computation by introducing repeatedly unfolding a shallow ResNet. A recurrent layer is further introduced to exploit the dependencies of deep features across stages, forming our progressive recurrent network. Furthermore, intra-stage recursive computation of ResNet can be adopted in PRN and PReNet to notably reduce network parameters with unsubstantial degradation in rain removal performance.
- **DerainNet (DDN):** It is a deep architecture that not only has benefits for high-level vision tasks but also can be used to solve low-level imaging problems [126]. It directly reduces the mapping range from input to output, which makes the learning process easier. To further improve the de-rained result, it also uses a priori image domain knowledge by focusing on high-frequency detail during training, which removes background interference and focuses the model on the structure of rain in images.

- **Gaussian Process-based Semi-Supervised Learning (GP-based SSL):** It is a comprehensive algorithm [127] that combines Gaussian Process, semi-supervised learning to improve the performance of the model on real-world images. It is more adept at rain and fog coexistence and less effective for rain maps such as large raindrops, which were rarely seen during previous training.

- **Lightweight Pyramid Network (LPNet):** It is a lightweight pyramid network [128] for single-image rain removal with domain-specific knowledge to simplify the learning process. It introduces the mature Gaussian-Laplacian image pyramid decomposition technology to the neural network. By decomposing the input image into multiple scales, the learning problem at each pyramid level is greatly simplified and can be handled by a relatively shallow network with few parameters

- **DerainNet:** It decomposes each image into a low-frequency base layer and a high-frequency detail layer [129]. Consequently, the detail layer is the input to the CNN for rain removal. To further improve visual quality, it also introduces an image enhancement step to sharpen the results of both layers since the effects of heavy rain naturally lead to a hazy effect.

6.2.2 Evaluation Metrics

In the quantitative evaluation of image quality, two full-reference image quality assessments were utilized: Peak Signal Noise Ratio (PSNR) and Structural Similarity (SSIM) [122] for synthetic rainy image datasets. The mathematical representation of the PSNR is as follows:

$$PSNR = 20 \cdot \log_{10} \left(\frac{MAX_f}{\sqrt{MSE}} \right). \tag{6.1}$$

where the MSE (Mean Squared Error) is:

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{6.2}$$

where f represents the matrix data of the original image, g represents the matrix data of the degraded image in question, m represents the number of rows of pixels of the images and i represents the index of that row, n represents the number of columns of pixels of the image and j represents the index of that column, MAX_f is the maximum signal value

that exists in the original image, known to be good. Besides, three commonly used non-reference image quality assessments: Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [130], Structural SIMilarity Evaluation with a Quality factor (SSEQ) [131], and Natural Image Quality Evaluator (NIQE) [132] were utilized for real-world rain image datasets.

6.2.3 Weather Noise Removal Evaluation Results

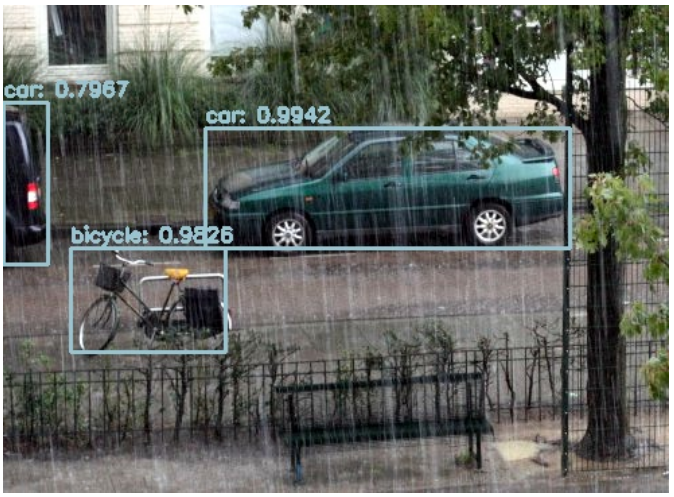
Ground truth	Testing image	PReNet	DDN	GP-based SSL	LPNet	DerainNet
PSNR		28.06	16.41	15.44	14.64	12.09
SSIM		0.88	0.52	0.39	0.46	0.27
BRISQUE		20.76	34.49	41.65	37.53	42.39
SSEQ		3.89	6.18	9.51	6.18	10.75
NIQE		85.50	89.72	73.65	85.34	83.01
Average process time (s)		0.1770	0.2660	0.2187	0.1137	0.4718

FIGURE 6.1: Comparison of Rain Noise Removal Algorithms.

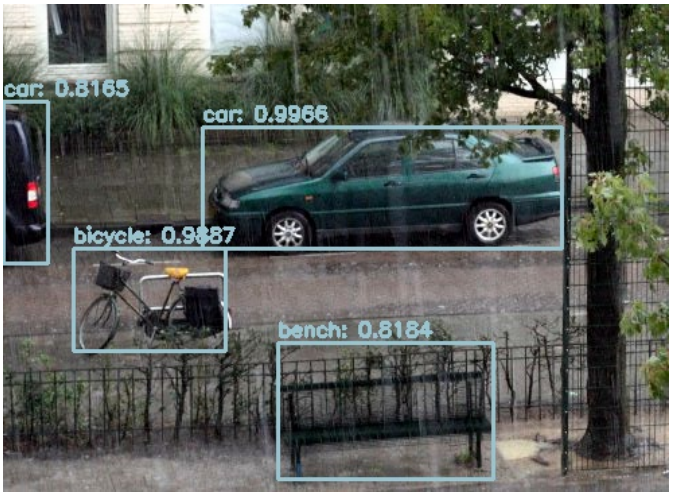
The author eventually selected an excellent algorithm: the Progressive ResNet (PReNet) algorithm [28] in the rain removal model according to the evaluation results in Section 6.2. PReNet is a recurrent neural network consisting of a simple combination of ResNet and multi-stage recursion, providing a favorable performance of image rain removal. It validated the tool with real-world rain image datasets [133] widely used in evaluations of rain removal algorithms. Finally, the experiment used YOLOv3 [134] to realize a simple object recognition application to offer further evidence of the beneficial effect of the tool for object detection applications. In Fig. 6.2, the tool predicted rainy weather successfully, and the generated fair image can be detected to a bench object with the YOLOv3 official model.

6.3 Weather-aware Object Detection Model Specialization

In this section, this experiment aims to validate the feasibility of the proposed approach through weather classification on weather-related noise rendering. Then, the performance of baseline models on maritime datasets and their performance degradation under adverse weather conditions is evaluated, followed by the comparison of specialized



(A)



(B)

FIGURE 6.2: Comparison of Object Detection Results under (A) Rainy and (B) Fair Weather Conditions.

models with baseline models. Finally, the model retraining experiment is conducted to validate the feasibility of model retraining to improve accuracy using a newly collected dataset in Weather-OD .

6.3.1 Experimental Methodology

The experimental environments, and maritime datasets for training object detection models can be summarized as follows. The experiment is carried out in a cloud server environment, which includes an NVIDIA A100 GPU providing high computing power suitable for large-scale data processing tasks. This environment is used for training and

retraining the object detection and weather classification models, as well as storing and processing large amounts of data. The performance of the object detection and weather classification models is evaluated under cloud settings commonly encountered in real-world maritime surveillance scenarios. Detailed specifications of the cloud environment are provided in Table 6.1.

Table 6.1: Experimental Environments.

Components	**Specifications**
CPU	Intel Xeon Platinum 8368 2.40GHz, RAM= 60GB
GPU	NVIDIA A100 GPU 40GB
OS	Ubuntu 20.04.6 LTS
Libraries	CUDA 11.5, Python 3.9, PyTorch 1.12.0, Torchvision 0.13.0

This experiment utilized two maritime object detection datasets for model specialization and two weather-related noise datasets for rain and haze noise rendering. To detect maritime objects, object detection tasks are based on the Singapore Maritime Dataset (SMD) [135] which includes 40 on-shore videos, 11 on-board videos, and 30 near-infrared on-shore videos with annotations for 10 object types including 6 types of ship and buoy. It is a commonly used comprehensive maritime benchmark dataset to provide better comparability for object detection in the maritime environment. For the experiment, 10% of the frames evenly was sampled to prevent overfitting, because there are only a few different ship shapes in the same video. Another dataset, DatasetShips (DS) [136] is also utilized for incremental learning with models trained on SMD. It is an open-source object detection dataset including about 5000 images with annotations for 10 types of vessels, more subcategorized than SMD.

Table 6.2: Comparison of mAP for Different Baseline Models on SMD, Synthetic SMD-rain, and SMD-haze Test Data.

Test data	**SMD**	**SMD-rain**	**SMD-haze**	**vs SMD-rain**	**vs SMD-haze**
YOLOv5	0.542	0.064	0.147	-88%	-73%
Faster R-CNN	0.553	0.051	0.121	-91%	-78%
DETR	0.313	0.021	0.096	-93%	-69%

Three deep learning-based object detection models can be obtained from pre-trained models on Microsoft COCO (Common Objects in Context) [137], which are publicly available on their respective official websites and on PyTorch Hub [138]. There are three types of models as follows:

- YOLOv5 [139], a single-stage detector that performs region proposal and object classification in a single pass of the network;
- Faster R-CNN [42], a two-stage detector that generates region proposals before performing classification and bounding box regression on the regions;
- End-to-End Object Detection with Transformers (DETR) [39], an object detection algorithm based on the Transformer architecture used in natural language processing tasks, which performs detection in an end-to-end manner without using anchor boxes or region proposals.

The experiments utilized three different variants of the YOLOv5 object detection model: YOLOv5s, YOLOv5m, and YOLOv5x, and took the average of their detection results. Additionally, one Faster R-CNN network with a ResNet-50-FPN backbone and one DETR network with a ResNet-50 backbone were adopted.

6.3.2 Weather Classification

This part aims to validate the performance of the **weather classification** model in automatically categorizing maritime data collected in maritime scenarios. Three commonly used evaluation metrics, **accuracy**, **recall**, and **F-1 score** (*higher* is *better*): are utilized to assess the object detection model accuracy. The accuracy metric measures the ratio of correctly classified objects to all objects in the dataset, while the recall metric measures the ratio of correctly identified positive objects to all positive objects in the dataset. The F-1 score is the harmonic mean of precision and recall, providing a single metric to evaluate the model's performance. Precision refers to the ratio of correctly identified positive objects to all objects classified as positive, while recall refers to the ratio of correctly identified positive objects to all positive objects in the dataset.

The evaluation results of the weather classification model are presented in Table 6.3, as demonstrated in the previous work [108]. The weather classification model achieved high precision and recall rates exceeding 95% on genuine images of the validation dataset. Additionally, the model accuracy is tested on synthetic datasets, including SMD-rain and SMD-haze based on SMD, using the weather classification model. The results of 100% precision and recall on synthetic data SMD-rain and over 99.5% precision and recall on SMD-haze proved the feasibility of automatic weather classification in collected data.

Overall, the results indicated that the weather classification model was highly accurate, with near-perfect performance for both datasets. By validating the accuracy of the weather classification model, it can be ensured that the collected data is correctly labeled and can be utilized to retrain specialized object detection models for better performance under specific weather conditions.

TABLE 6.3: Weather Classification Evaluation Results.

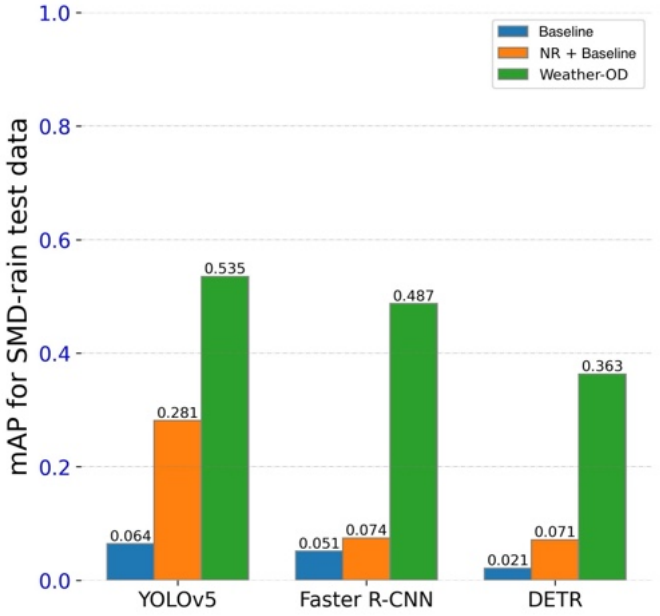
Type	Accuracy	Recall	F-1 Score
Fair	95.20%	96.00%	95.60%
Rainy	98.00%	95.50%	96.70%
Hazy	91.60%	94.40%	93.00%

6.3.3 Model Specialization Evaluation Results

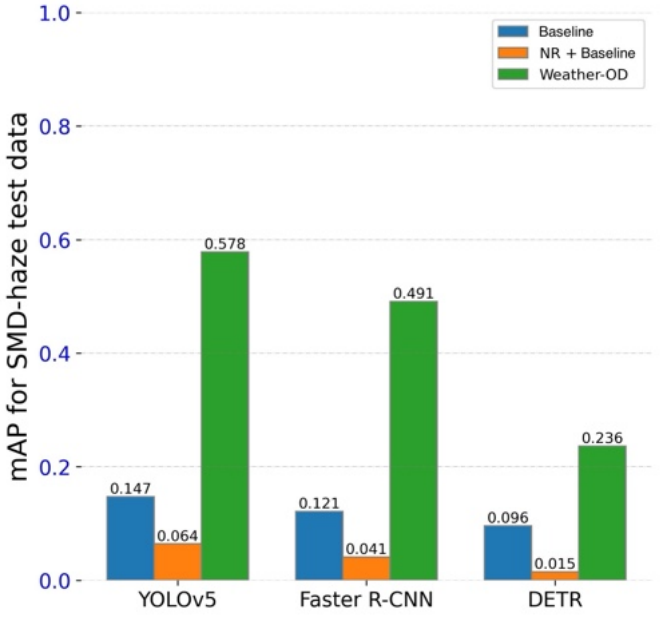
In this part, the results of the experiment are presented in a comparison table (Table 6.2), which shows the mAP scores for each model on each dataset, as well as the percentage difference in performance between the original SMD dataset and the synthetic rainy and hazy SMD datasets. Besides, the differences between the mAP on SMD and the synthetic adverse weather datasets are also included. The mAP scores and differences are reported as percentages, with negative percentages indicating a decrease in performance for the synthetic datasets compared to the standard SMD dataset. It can be found that both rain noise and haze noise lead to severe performance degradation in maritime object detection, with the mAP dropping from 88% to 93% and from 69% to 78%, respectively. These results highlight the importance of weather-aware object detection models in maritime surveillance systems.

The experiment was conducted to validate the effectiveness of baseline object detection models for improving object detection accuracy in maritime environments. Specifically, the performance degradation of these models under rainy and hazy weather conditions was focused on the evaluation. The **mean Average Precision (mAP)** (*higher* is *better*) metric was utilized to evaluate the performance of the models, which combines precision and recall across multiple Intersection over Union (IoU) thresholds to measure the accuracy of the models. Three different baseline models, namely YOLOv5 [139], Faster R-CNN [42], and DETR [39], were evaluated on three different datasets: the original SMD dataset, synthetic rainy SMD (SMD-rain), and synthetic hazy SMD (SMD-haze).

The objective of this experiment is to compare the performance of specialized object detection models in Weather-OD with existing baseline models, and those combined with weather-related noise removal pre-processing in the existing approach. The author tested and compared the following three methods under adverse weather conditions, and finally assessed statistical differences with ANOVA (Analysis of Variance) [140]:



(A)



(B)

FIGURE 6.3: Comparison of mAP on Synthetic (A) Rainy SMD-rain and (B) Hazy SMD-haze Test Data.

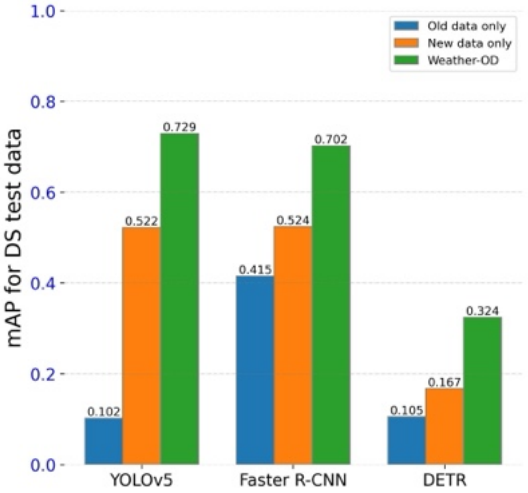
1. **Baseline**: Three types of baseline models as pre-trained models and trained the models using the SMD dataset. The author directly inputs these models with the SMD-rain and SMD-haze test data.

2. **Baseline models with noise removal (NR + Baseline)**: This is a commonly used method to cope with weather-related noise in existing studies. To implement this approach, the author used the rain noise removal model, Progressive ResNet (PReNet) [28], and the haze noise removal algorithm, Big FastNet [141]. The experiment used the rain and haze removal models to pre-process the SMD-rain and SMD-haze test data and obtain visually restored images for the baseline models.

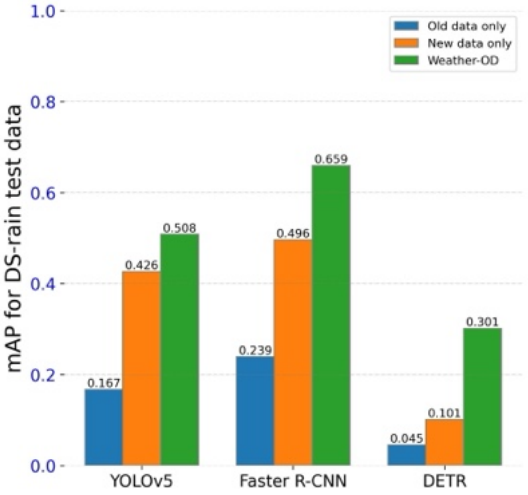
3. **Weather-OD :** The author trained a series of specialized models for rain and haze weather conditions on the SMD-rain and SMD-haze training data. Then these models were input into the corresponding special models with the SMD-rain and SMD-haze test data.

Fig. 6.3 compares the mAP on (A) synthetic rainy SMD test data in Fig. 6.3a and (B) synthetic hazy SMD test data in Fig. 6.3b with three methods above. When rain removal pre-processing with PReNet was applied to these baseline models, the mAP improved slightly. However, when haze removal pre-processing with Big FastNet was applied to these baseline models, the mAP was degraded compared with the baseline models. The reason is that haze removal tends to sharpen the object excessively, leading to unrealistic image outputs, which usually aggravates the task of feature extraction for the object detection model. There is thus no guarantee that weather-related noise removal pre-processing will help object detection, even if these methods achieve a certain level of effectiveness in visual restoration with non-reference image quality assessment [28] [141]. Additionally, Weather-OD with the specialized models achieved the highest mAP values on both rainy and hazy SMD test data, indicating their superior performance in these weather conditions.

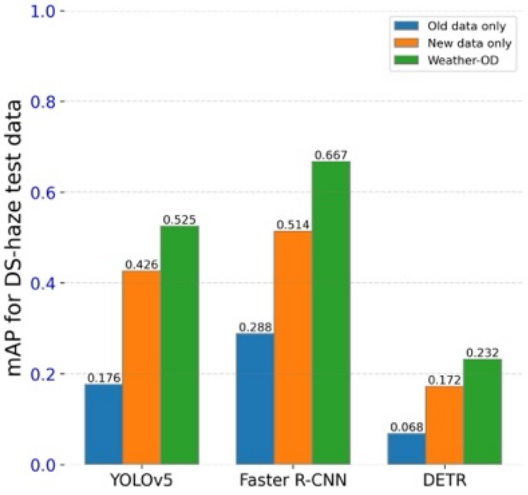
To assess whether there are any statistically significant differences in the above results, a statistical analysis using Analysis of Variance (ANOVA) [142] is also conducted. ANOVA is a statistical method that helps to interpret the experimental results by statistically evaluating the variance of data and differences between groups. In this experiment, the author analyzed the mAP values of the three method groups: baseline, NR + baseline, Weather-OD , and yielded two values: the F-statistic (F) and p-value (p). Each



(A)



(B)



(C)

FIGURE 6.4: Comparison of mAP on (A) DS, (B) DS-rain, and (C) DS-haze Test Data.

method encompasses outcomes derived from three model types (YOLOv5, Faster R-CNN, DETR) and two weather conditions (rain, fog). The F-statistic represents the ratio of the treatment's mean squares to the error's mean squares. The p-value, another outcome of this analysis, determines the statistical significance of differences between group means. If the computed p-value falls below the threshold of 0.05, it rejects the null hypothesis of the ANOVA and infers that substantial statistical distinctions exist between the means of the three groups.

The ANOVA analysis yielded results as follows: $F = 18.695, p < 0.0027$. They indicate that there are indeed substantial differences in the mAP values achieved by Weather-OD compared to the two existing methods. The results validate the purpose that the proposed approach has a notably different impact on object detection performance under various weather conditions. These findings suggest that specialized models for specific weather conditions can significantly improve the performance of object detection models under adverse weather conditions.

Finally, the average inference time of the three models is 232 ms, 769 ms, and 504 ms, respectively, on the NVIDIA Jetson TX2, and there is no difference from their baseline models. It highlights the specialized models executed on edge servers can provide performance improvements without increasing latency for maritime surveillance.

This part aims to validate the feasibility of model retraining to improve accuracy using a newly collected dataset in Weather-OD , and finally assess statistical differences using ANOVA. The author conducted an experiment on incremental learning using the DatasetShips (henceforth, DS) dataset with three models trained on the SMD dataset. The aim was to explore the effectiveness of incremental learning on maritime object detection tasks. The mean average precision results evaluated on DS test data and its synthetic DS-rain and DS-haze test data are shown in Fig. 6.4 from (A) to (C). These figures respectively showed the results for specialized models only on old training data, specialized models only on new training data, and specialized models on SMD with incremental learning on corresponding DS series training data. For each test dataset, the three figures show the mAP scores for three specialized models: YOLOv5, Faster R-CNN, and DETR. For each model, the mAP scores are shown for three training patterns:

1. **Old data only:** They were trained on SMD training data or the synthetic SMD-rain, and SMD-haze, from pre-trained models.

2. **New data only:** They were trained on DS training data, or the synthetic DS-rain, and DS-haze, from pre-trained models.

3. **Weather-OD :** It implemented incremental learning on DS series training data from their corresponding SMD series models.

The mAP results show that specialized models (old data only) perform poorly on the DS series test data, with mAP scores ranging from 0.045 to 0.415. However, the models with incremental learning on DatasetShips show significant improvements in mAP scores, with an increase of up to 10 folds in mAP scores on the DS series test data. Although the models (new data only) also improve mAP to a certain extent, it is still not nearly as much as the improvement from incremental learning.

Similar to the above-mentioned descriptions, the author conducted an ANOVA test to determine if there are statistically significant differences among the mAP values of three model training patterns: Weather-OD , old data only, and new data only. The obtained results for the mAP values across these training patterns are as follows: $F = 9.6692, p < 0.0009$. These experimental results validate the intended objectives, confirming the statistically significant differences between the proposed method Weather-OD and current methods.

These results indicate that incremental learning on maritime object detection tasks can be effective in improving the accuracy of the model on a new dataset while still maintaining the same level of performance on the original dataset. The models trained on the SMD series data, which was a more general maritime dataset, were not able to perform as well on the DS dataset, which had a more narrow focus on vessels. By continuously updating the model with new data, the system can improve its performance over time and adapt to concept shifts. The experiment highlights the importance of incremental learning in object detection tasks and emphasizes the need for domain-specific datasets to improve the accuracy of object detection models.

6.3.4 Validation on Real Maritime Data

To validate the proposed weather-aware object detection method, the author utilized actual weather data comprising 10 rainy and 10 hazy weather videos collected from on-board recordings on YouTube. The validation compared the performance of the baseline object detection model, trained with the SMD dataset, against the specialized object detection model developed using the proposed method. The key metric for evaluation was the average confidence score of object detection, which reflects the reliability of detected objects under adverse weather conditions.

A confidence score shows the probability of the image being detected correctly by the algorithm and is given as a percentage. The confidence score S is a numerical value (usually between 0 and 1) that represents the model' s certainty that a detected object actually belongs to a particular class. It is computed by the neural network as part of the detection pipeline, which can be expressed as follows:

$$S = P(\text{Object}) \times IoU_{\text{pred}}^{\text{truth}}$$

where $P(\text{Object})$ is the probability that an object exists in the bounding box, and $IoU_{\text{pred}}^{\text{truth}}$ is measured by the magnitude of overlap between two bounding boxes of ground truth and prediction. The formula for IoU is given below:

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}.$$

In this experiment, the author evaluated the average confidence score of each video in object detection is defined as the mean confidence score of all detected objects across all frames in the video. If a video consists of F frames, and each frame f contains N_f detected objects with confidence scores $C_{f,i}$, then the average confidence score for the video can be calculated as follows:

$$\bar{S} = \frac{1}{\sum_{f=1}^{F} N_f} \sum_{f=1}^{F} \sum_{i=1}^{N_f} S_{f,i}$$

where F is total number of frames in the video, N_f is number of detected objects in frame f, and $S_{f,i}$ is confidence score of the i-th object detected in frame f.

The BRISQUE [130] metric is a model that uses only the image pixels to calculate features. It calculates image quality by first loading the image and computing Mean Subtracted Contrast Normalized (MSCN) coefficients along with pairwise products to capture structural distortions. These coefficients follow different statistical distributions in distorted and natural images. Next, the MSCN coefficients are fitted to Generalized Gaussian Distributions (GGD) and Asymmetric Generalized Gaussian Distributions (AGGD) to extract shape and variance parameters. The image is then resized to extract additional features, forming the BRISQUE feature vector. These features are scaled to the range from -1 to 1 using the same parameters as the original model training, ensuring consistency. Finally, the scaled features are fed into a Support Vector Regressor (SVR) model trained on quality scores to predict the final image quality assessment, trained on an image database with corresponding differential mean opinion score (DMOS) values. The database contains images with known distortion such as compression artifacts, blurring, and noise, and it contains pristine versions of the distorted images. The BRISQUE score is a measure of image quality that ranges from 0 to 100, with lower scores indicating better image quality.

Here, Fig. 6.5 shows sample images of real maritime data in rainy and hazy conditions [146] [147]. Then Fig. 6.6 and Fig. 6.7 show the comparison of sample results of the baseline model and the specialized model on real rainy and hazy maritime data, respectively. From these results, the proposed method significantly detected more objects with higher confidence scores than the baseline model, which outperformed the baseline model.

After the validation, the author also conducted a quantitative evaluation to compare the performance of the proposed method with the baseline model. The two figures in Fig. 6.8 show the comparison of mean confidence scores and mean BRISQUE scores of the baseline model and the specialized model on real hazy maritime data, whose mean scores were calculated for each video. More confidence scores indicate better object detection performance, while lower BRISQUE scores indicate better image quality of video data.

The proposed method demonstrated an approximately 1.3-fold improvement in confidence scores compared to the baseline, highlighting its ability to mitigate the negative

impact of weather noise on object detection. This improvement underscores the efficacy of weather-specific model specialization in enhancing detection accuracy in adverse weather environments. While the proposed method shows clear advantages in moderate bad weather, its performance is yet limited under extreme weather conditions, such as heavy storms or dense haze. In these scenarios, the quality of restored images from existing weather removal techniques further deteriorates, reducing the effectiveness of both the baseline and the proposed method. Overall, the results validate the hypothesis that a weather-aware approach can significantly improve object detection performance in real-world maritime scenarios with adverse weather. However, they also suggest the need for further refinement to address detection limitations in extreme conditions.

(A)

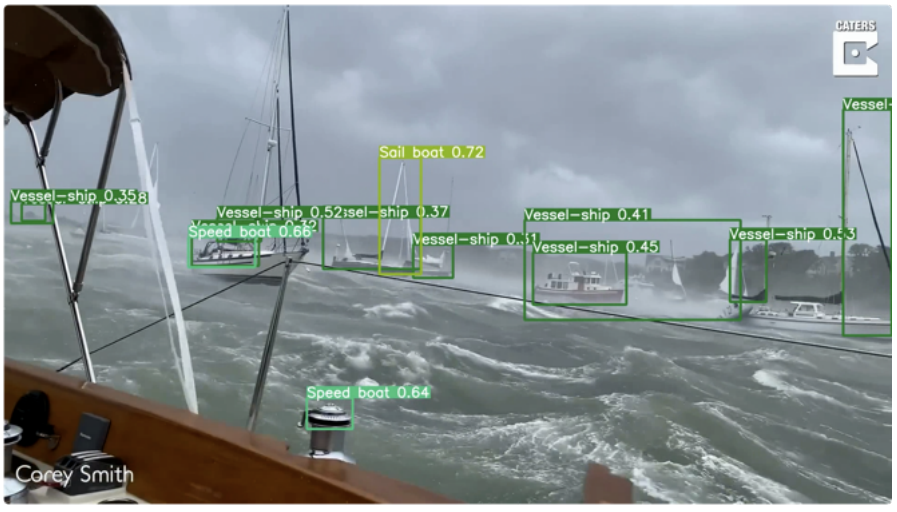
(B)

FIGURE 6.5: Sample Images of Real Maritime Data under (A) Rainy and (B) Hazy Weather Environments.

6.3.5 Discussion

In this part, it acknowledges that while Weather-OD presents promising results, there are limitations to its implementation in real-world maritime scenarios. One such limitation stems from the nature of the weather classification model, which can only provide a single-type output as described in weather classification results.This constraint becomes particularly relevant in practical maritime scenarios where multiple weather conditions, such as rain and fog, can occur simultaneously. Since existing approaches considered introducing multiple labels to weather classification [143] to predict multiple weather conditions for one image, which are still hard to simply apply to model specialization.

(A)

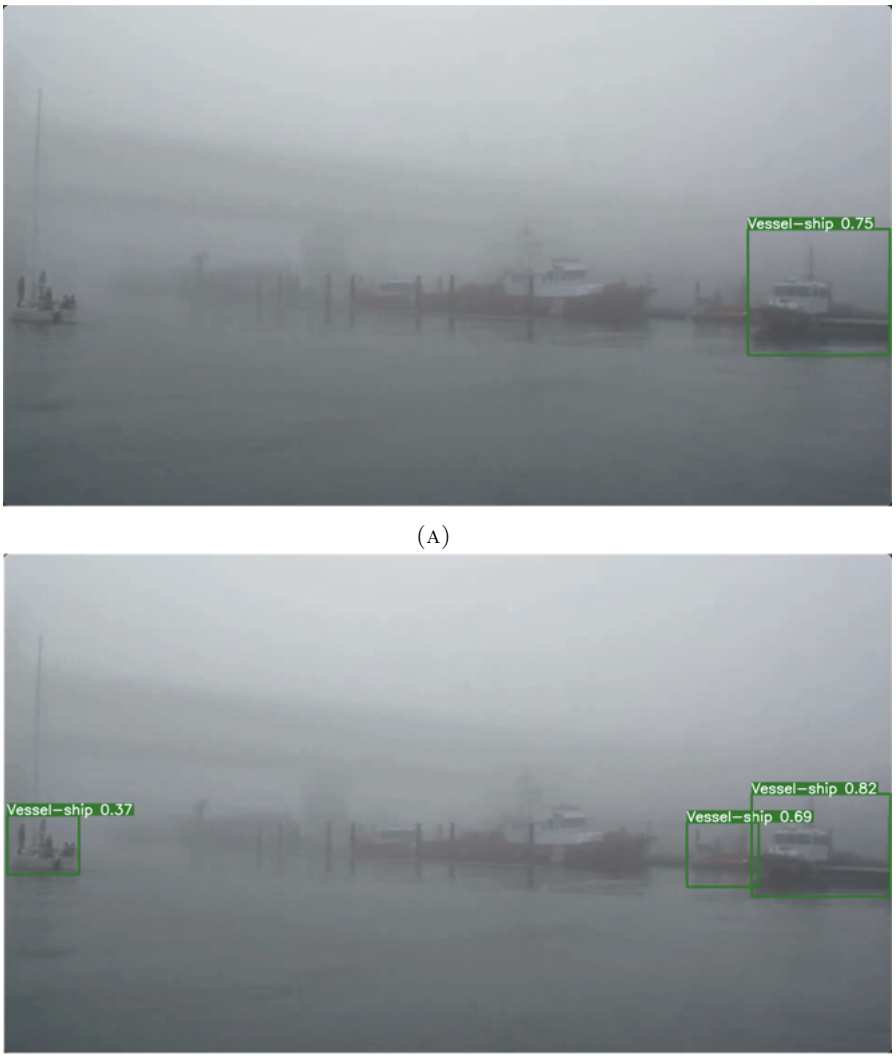


(B)

FIGURE 6.6: Comparison of Sample Results of (A) Baseline Model and (B) Specialized Model on Real Rainy Maritime Data.

Considering these limitations, future endeavors will be focused on enhancing the integration of weather classification and object detection models, aiming to enable these systems to simultaneously provide results for various weather types.

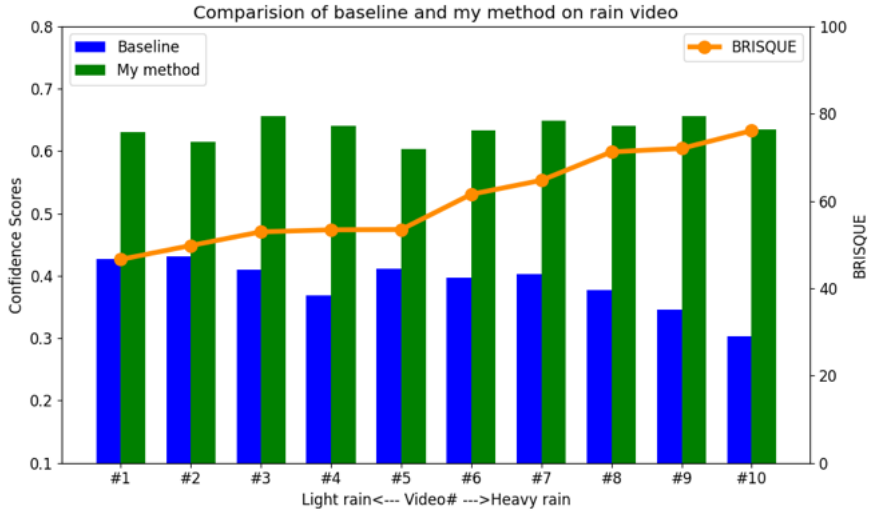
Furthermore, the model's performance in nighttime or low-light conditions also faces challenges from the reliability of images captured by electro-optical cameras in 4.2.3.1. While existing approaches [98] incorporate sensors like night cameras or radar to aid in object detection, they still encounter challenges stemming from inadequate datasets or significant interference caused by nighttime conditions.



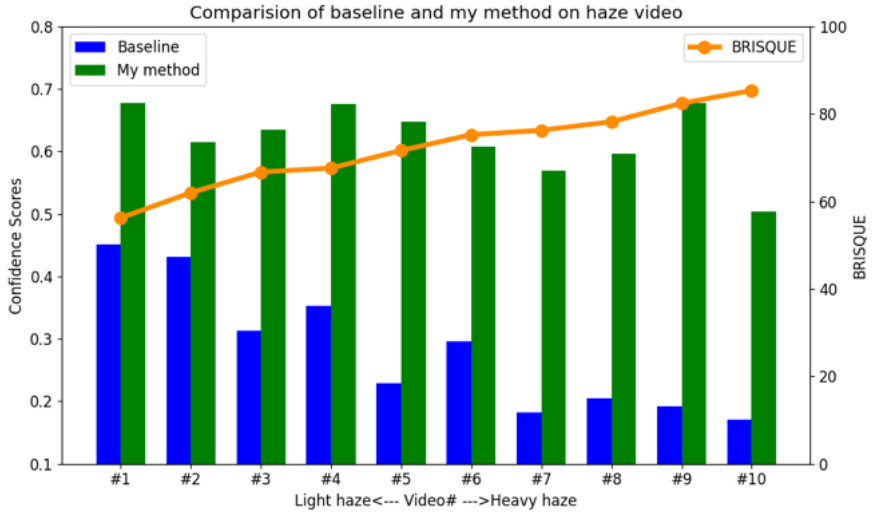
(A)

(B)

FIGURE 6.7: Comparison of Sample Results of (A) Baseline Model and (B) Specialized Model on Real Hazy Maritime Data.



(A)



(B)

FIGURE 6.8: Comparison of Sample Results of (A) Baseline Model and (B) Specialized Model on Real Hazy Maritime Data.

6.4 Validation of Consistency Evaluation

This section presents the results of evaluation experiments on the proposed consistency evaluation and consistency improvement method.

6.4.1 Experimental Methodology

The experiments were conducted in an experimental environment equipped with an NVIDIA A100 GPU to evaluate the consistency of model inference results and to measure the improvement of consistency by extracting low-consistency training data. Detailed

specifications of the experimental environment are shown in Table 6.4. The object detection method used in this study is an object detection model based on YOLOv5 [139], which has recently gained a great reputation among composite scale object detection models. Common image datasets such as ImageNet [144] and Microsoft COCO [137] are filled with visually dissimilar images, instead of time-series data. This makes them suitable for measuring average accuracy, but not for measuring consistency. Instead, the author adopted the SMD dataset [98], which consists of time-series images.

Table 6.4: Experimental Environments.

Components	**Specifications**
CPU	Intel Xeon Platinum 8368 2.40GHz, RAM= 60GB
GPU	NVIDIA A100 GPU 40GB
OS	Ubuntu 20.04.6 LTS
Libraries	CUDA 11.5, Python 3.9, PyTorch 1.12.0, Torchvision 0.13.0

Table 6.5: Comparison of Average Consistency Improvements with Retraining.

Retraining Pattern	**Initial Model**	**Retrained Model**	**Improvement**
All data	0.825945	0.956555	15.81%
Random	0.767963	0.922988	20.19%
Low confidence	0.767963	0.912219	18.78%
Low consistency	0.767963	0.959183	24.90%

To initially verify the proposed consistency evaluation method and the possibility of improving consistency through data extraction, retraining is performed with the following four methods of extracting training data:

- **All data**: The initial model is trained on all initial training data, and retrained on all retraining data.
- **Random**: Retraining with 10% of the retraining data randomly against the initial model trained with 10% of the data randomly extracted as a baseline.
- **Low confidence**: The baseline initial model was retrained by extracting the retraining data from the lower 10% of confidence in the inference results of the initial model, as in the existing method by Kong et al. [145].
- **Low consistency**: For the baseline initial model, retraining was performed by extracting the retraining data from the inference results of the initial model with the lower 10% of the consistency.

6.4.2 Evaluation Results

TABLE 6.6: Comparison of mAP Results with Retraining.

Retraining Pattern	Initial Model	Retrained Model	Improvement
All data	0.482822	0.939627	94.61%
Random	0.465673	0.827826	77.77%
Low confidence	0.465673	0.775002	66.43%
Low consistency	0.465673	0.942777	102.45%

The effect of the four retraining methods on consistency improvement is shown in Table 6.5. The results of Table 6.5 indicate that retraining with low-consistency training data is the most effective at improving consistency. Retraining with low-consistency training data is the most effective, with a higher rate of improvement in consistency than other retraining methods. It was also able to improve the results to the same level of consistency by retraining all data with 10 times the training data. The mAP improvement effect of the four retraining methods is shown in Table 6.6. The results in Table 6.6 show that retraining with low-consistency training data is the most effective, improving consistency by 102% and increasing mAP at a higher rate than the other retraining methods. Furthermore, the mAP results outperformed retraining on all data with 10 times the amount of data. Retraining with low-consistency training data is the most effective in improving consistency and mAP, and is verified to improve model performance.

TABLE 6.7: Comparison of Average Consistency Results with Continual Retraining.

Retraining	All Data	Random	Low Confidence	Low Consistency
Initial	0.825945	0.767963	0.767963	0.767963
1st	**0.800732**	0.773174	0.749319	<u>0.793161</u>
2nd	**0.81479**	0.773656	<u>0.812359</u>	0.79521
3rd	**0.869623**	0.799986	0.785523	<u>0.861749</u>
4th	<u>0.875427</u>	0.803903	0.860819	**0.888386**
5th	<u>0.956555</u>	0.922988	0.912219	**0.959183**

The SMD dataset was divided into approximately 60 pairs of time-series image data, and the previous 60 pairs of time-series image data were used as the training data for the initial model. Using the initial model described above, five-times retraining runs were conducted for each of the 10 pairs of time-series image data using the four retraining methods. The average consistency of the initial model and the inference results of the five successive retraining runs are shown in Table 6.7. Especially, the highest consistency value is shown in bold, and the second highest value is underlined. Sustained

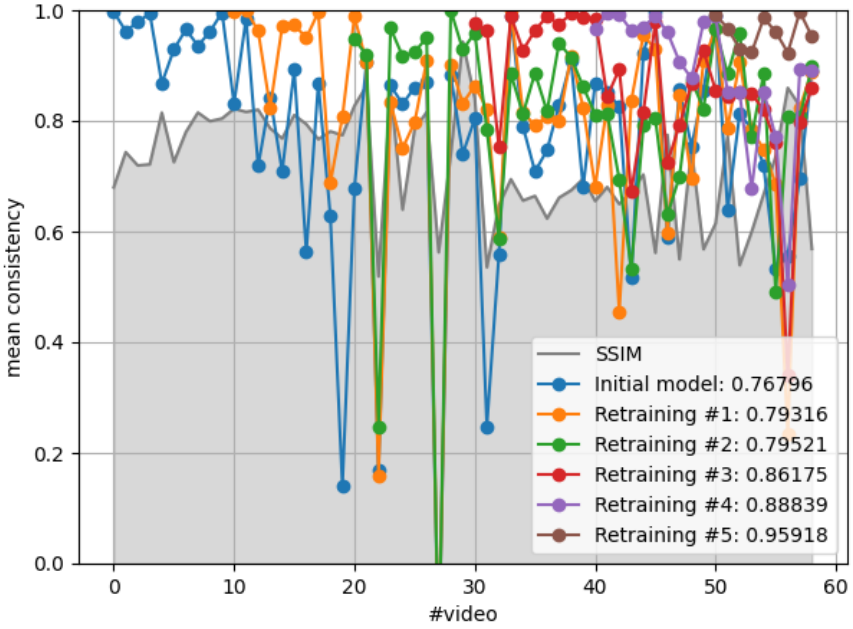


FIGURE 6.9: SSIM of Time-series Image and Consistency.

improvement in consistency was observed from the first to the fifth retraining. In this context, the existing method of extracting low-confidence data showed less improvement in consistency than retraining a random selection as a baseline. It is suggested that retraining the model by extracting low-confidence data from the proposed method is effective in improving consistency.

The relationship between SSIM and average consistency of time series images is shown in Fig. 6.9 as an example of the results of average consistency by retraining the proposed method. The average SSIM is represented by a gray fill, and the average consistency results for each number of retraining are represented by different color folds. Although there is some variation in consistency across the SSIMs of the time-series image data, overall, the consistency tends to improve as the number of retraining cycles increases. When the SSIM of the time-series images decreased, the consistency also tended to decline.

6.5 Object Detection Model Optimization

This section presents the results of the object detection model optimization experiment, which aims to evaluate different optimization plans for model retraining by simulating performance degradation of maritime data.

6.5.1 Experimental Methodology

The purpose of this experiment is to evaluate different optimization plans for model retraining by simulating performance degradation using simulated data. The experiment aims to examine how model performance changes over time and in response to retraining triggered by specific conditions. Experimental setup and configuration can be summarized including mobile device configuration, model performance changes, and retraining implementation.

TABLE 6.8: Experimental Environments.

Role	**Type**	**Amount**	**GPU Memory**	**Training Capacity**
Cloud		1	40 GB	Both models
Edge	Large	2	16 GB	Both models
	Small	2	10 GB	Only YOLOv5s

TABLE 6.9: Experimental Model Configurations.

Role	**Item**	**YOLOv5x**	**YOLOv5s**
Cloud	GPU memory usage	33.2 GB	11.4 GB
	Unit training time	0.5 hr	0.4 hr
Edge	GPU memory usage	14.7 GB	7.1 GB
	Unit training time	5 hr	4 hr

Table 6.8 shows the experimental environments, which consist of cloud and edge servers on mobile devices with different computational resources. Table 6.9 shows the configurations of the experimental models, including GPU memory usage and training time per unit training data, according to experiment results. The experiment platform setups in the experiment are configured to represent a group of mobile devices traveling between different sea areas, and their configurations are as follows:

- **Mobile Devices (Edge):** Group of mobile devices are configured as four mobile devices, all traveling one-way between two distinct sea areas. These entities belong to the same group, enabling them to share models and potentially retrain collaboratively based on the group's conditions.
- **Initial Models**: Two types of initial models are used in the experiment, differing in the number of parameters and initial performance. Specifically, YOLOv5s and YOLOv5x represent more lightweight and high-performance models requiring greater computational resources, respectively.

- **Computational Resources:** The mobile devices are divided into two groups based on their computational resources. The first group consists of devices with limited computational resources, while the second group includes devices with larger computational resources.
- **Inference Capability:** All devices are capable of running inference using both YOLOv5s and YOLOv5x models.
- **Training Capability:** Only the devices with larger computational resources can perform local training with the more demanding YOLOv5x model. This distinction allows to observe the impact of resource limitations on retraining decisions.
- **Training Data:** The models are evaluated based on their training time, number of epochs, and GPU memory usage. These factors are pre-configured using data from previous experiments with YOLOv5s and YOLOv5x models.

The data used in the experiment includes Singapore Maritime Dataset (SMD) [98] collected from Singapore, and KOLOMVERSE dataset [148] collected from Busan, South Korea. It simulates the performance degradation during ships traveling between these two areas. As time progresses during the experiment, the performance of the models is subject to increasing randomness, simulating the natural degradation that occurs as the environment changes. The length of the experiment is set to 100 units of time cycle, with each unit representing one hour. Throughout this period, the simulated data is designed to reflect a decline in model performance. In this experiment, the performance of the models is evaluated based on the confidence score as a proxy metric for performance, which is used as the key performance indicator. The threshold values for acceptable performance vary depending on the mobile device's location including two phrases:

- **Entering and Exiting Ports:** During the first and last 10 time cycles, the threshold for performance is set to 0.8.
- **Sailing at Sea:** During the middle 80 time cycles, the performance threshold is set to 0.6 to reflect the lower precision required during this phase of travel.

Besides, more implementation details of model retraining triggers are as follows:

- **Retraining Interval:** Retraining occurs at regular intervals specified by the period parameter $\tau_{interval}$, set to 50 time cycles.
- **Cumulative Performance Decline Threshold:** The cumulative performance decline threshold $\tau_{cumulative}$ is set to 0.2, indicating the maximum acceptable performance degradation accumulated before retraining is triggered.
- **Precautionary Trigger Extra Period:** The extra period for precautionary retraining τ_{extra} is set to 5 time cycles, defines the additional buffer time to ensure retraining is triggered before performance falls below the threshold in future.

The experiment includes a simulation of retraining events. When retraining is triggered based on the chosen optimization plan, new simulated data is generated to assess whether the model's performance improves after retraining. This simulation allows to test whether retraining under different conditions can effectively counteract the performance degradation observed over time.

By comparing the performance before and after retraining, it can assess the efficiency and effectiveness of various retraining strategies in maintaining model performance across different time cycles and sea areas. This experiment aims to validate the effectiveness of the proposed method for model selection and retraining in the maritime environment. To ensure a robust comparison, several methods address the retraining conditions and strategies in a unique manner, and their performances are measured under the same experimental settings as follows:

- **No Retraining** The first comparison is a simple baseline that does not involve any retraining as a baseline. In this method, the model uses the initial simulated data generated at the start of the experiment, and no retraining occurs regardless of performance degradation or other factors. This baseline serves as a control to observe how model performance declines over time when no interventions are made to maintain or improve performance.
- **Periodic Retraining:** The method involves periodic retraining, where the model undergoes retraining at regular intervals, independent of its current performance. The retraining period to half of the total experiment time (period = 50) was set in the experiment. This approach presents the impact of retraining frequency on

model performance and resource consumption without considering performance changes.

- **Performance Threshold-Based Retraining:** The third comparison method introduces a more dynamic strategy by utilizing a performance threshold to trigger retraining. In this case, retraining is only performed when the model's performance falls below a predefined threshold. This method ensures that retraining only occurs when the model's performance declines, potentially reducing unnecessary retraining events while maintaining acceptable performance.

- **Cost-aware retraining algorithm (Cara):** The fourth method employs the *Cara* retraining strategy, an existing approach [89] that combines both performance threshold and cumulative performance degradation as retraining triggers. By using these two thresholds, *Cara* aims to balance retraining frequency with the need to maintain high performance, offering a more nuanced approach to retraining compared to methods that rely on a single condition.

- **Proposed Method:** The proposed method introduces an optimization strategy that combines model selection and retraining decisions based on both current model performance and available computational resources. This approach dynamically selects the most suitable model and retraining schedule to minimize resource consumption while ensuring the model meets the required performance threshold. Through this method, it is aimed to demonstrate superior performance in terms of performance, computational efficiency, and retraining frequency compared to the existing baselines.

Each method will be evaluated on key metrics such as confidence scores, computational resource usage, and retraining frequency to provide a comprehensive assessment of their effectiveness by comparing the results of these methods started from same initial models. The evaluation of optimization performance in this experiment is based on several key metrics that reflect the effectiveness of different retraining strategies:

- **Average Confidence Score** (*Higher* is *better*): The confidence score (CS) is used as a proxy for the model's predictive performance, with higher confidence scores indicating better performance.

- **Threshold Satisfaction Ratio** (*Higher* is *better*): The Threshold Satisfaction Ratio (TSR) evaluates the model's compliance with the predefined performance threshold. By measuring the percentage of time steps in which the model meets the performance threshold, it can be assessed how well each method maintains the required performance level throughout the experiment.

- **Performance-Threshold Ratio** (*Higher* is *better*): The Performance-Threshold Ratio (PTR) is a metric dependent on an arbitrary performance metric and its corresponding performance threshold, where

$$PTR_n = \frac{Performance_n}{Threshold_n}.$$

  This ratio represents the extent to which the model meets the required performance threshold during each evaluation. It indicates how well the model's performance aligns with the predefined performance threshold at each time step, providing a measure of compliance with the set performance standards.

- **Objective Function Value**(*Lower* is *better*) This metric assesses the overall optimization objective per each evaluation. For the plan of minimizing time and computing resources, the Objective Function Value (OFV) represents the total retraining time (unit: hr) or computational resources, i.e. GPU memory consumption (unit: GBhr) consumed by each optimization method. For the plan of performance improvement, the OFV represents the total expected TSR (unit: time) per each evaluation, according to model update execution by each optimization method.

By using these metrics, the evaluation experiment comprehensively evaluates the efficiency and effectiveness of various retraining strategies, with a particular focus on minimizing retraining costs while maintaining acceptable levels of performance.

6.5.2 Pseudo-labeling Retraining Evaluation Results

To validate the effectiveness of the proposed pseudo-labeling implementation, experiments were conducted using the KOLOMVERSE dataset [148], which contains approximately 5,700 sampling images. The dataset was divided into five subsets, and a

Table 6.10: Comparison of Retraining Results on Pseudo Label and Human Label.

Metrics	Instance sizes	Pseudo label	Human label	Ratio
Precision	small	0.083	0.281	0.295
	medium	0.194	0.354	0.548
	large	0.470	0.519	0.906
Recall	small	0.083	0.300	0.277
	medium	0.312	0.453	0.689
	large	0.592	0.636	0.931

cross-validation-like methodology was adopted to ensure the robustness of the evaluation and to minimize biases arising from the initial training data.

The experiment was conducted in the following manner: One subset of the dataset was selected as the initial training set, and a base model was pre-trained on this subset. This model served as the starting point for pseudo-labeling. The remaining four subsets were used for inference using the pre-trained model. Predictions with a confidence score of 95% or higher were treated as pseudo-labels, forming new training datasets for each of the four remaining subsets. Starting with the pre-trained SMD model as the base model, the pseudo-labeled datasets from each of the four remaining subsets were individually used for retraining. This iterative fine-tuning allowed the model to adapt to additional data derived from pseudo-labeling while incorporating the biases and dynamics of the maritime dataset. To eliminate the potential influence of the initial training subset's data distribution, a five-fold cross-validation-like experiment was performed. In each fold, a different subset was chosen as the initial training set, while the remaining four subsets were used for pseudo-labeling and fine-tuning. This ensured that each subset was used as the initial training set and pseudo-labeled data. After fine-tuning, the model performance was evaluated on a test dataset, independent of the training subsets. The accuracy of the pseudo-labeling approach was compared against the manually annotated labels to quantify its effectiveness.

The experiment compared the performance of datasets with pseudo labeling and human labeling in training object detection models, based on the mean Average Precision (mAP) at 50% IoU. Over five training iterations, the pseudo-labeled data achieved an average mAP of **0.78**, while the human-labeled data achieved a higher mAP of **0.87**. This indicates that pseudo-labeling can achieve a reasonable level of accuracy but falls short of the performance attained with human-annotated data, with approximately a 10% performance gap.

Further analysis revealed that the size of the detected objects significantly influenced the performance of pseudo-labeling, whose results are summarized in Table 6.10. The results analyzed by Python library *pycocotools* [149], show the precision and recall both at varying IoU thresholds, ranging from 0.50 to 0.95 in steps of 0.05. The instance sizes were categorized into three groups: small, medium, and large, based on the area of the detected objects according to the COCO evaluation metric. It decided that objects with an area between 0 and 32^2 pixels will be categorized as small, objects with an area between 32^2 and 96^2 pixels would be categorized as medium, and objects with an area larger than 96^2 pixels would be categorized as large. For large objects, the mAP achieved with pseudo labels closely approached that of human labels, suggesting that the pseudo-labeling method is effective for objects with distinct and easily detectable features. However, for small objects, the accuracy of pseudo labels dropped substantially, highlighting a notable limitation. Small objects often have lower visual resolution and are more susceptible to misclassification or omission during automated labeling, resulting in a greater disparity than human labels. The results indicate that pseudo-labeling can be a viable alternative to human labeling for objects with clear and distinguishable features.

6.5.3 Optimization Evaluation Results

The results of the experiment are summarized in Table 6.11, Table 6.12, and Table 6.13. The bold values in each table indicate the best performance, while the underlined values represent the second-best performance.

The experiment results for the time minimization plan are summarized in Table 6.11, whose each method's performance is evaluated across above four metrics. The proposed method demonstrates the lowest retraining time as average OFV with approximate 2/3 reduction, significantly outperforming other methods. It achieves the same level of performance (average CS, TSR, average PTR) as other methods: performance threshold, and method *Cara*. Overall, the proposed method balances time-efficiency and performance, achieving notable resource savings while maintaining reliable detection accuracy.

The results of the resource minimization plan experiment are shown in Table 6.12. This plan evaluates each method's efficiency in conserving computational resources while maintaining detection performance. Similarly, the proposed method achieves the best

Table 6.11: Results of Time Minimization Plan.

Metric	**No Training**	**Periodic**	**Performance Threshold**	**Cara**	**Proposed Method**
Average CS	0.62	0.689	0.763	**0.780**	0.777
TSR	0.447	0.690	0.915	**0.948**	0.938
Average PTR	0.606	0.648	**1.064**	1.015	0.979
Average OFV			9.2	9	**3.6**
Comparison			x2.56 times	x2.50 times	1

Table 6.12: Results of Resource Minimization Plan.

Metric	**No Training**	**Periodic**	**Performance Threshold**	**Cara**	**Proposed Method**
Average CS	0.620	0.689	0.759	**0.787**	0.785
TSR	0.447	0.690	0.918	0.950	**0.960**
Average PTR	0.606	0.648	1.017	**1.116**	0.968
Average OFV			0.251	0.689	**0.091**
Comparison			x2.76 times	x7.57 times	1

Table 6.13: Results of Performance Improvement Plan.

Metric	**No Training**	**Periodic**	**Performance Threshold**	**Cara**	**Proposed Method**
Average CS	0.620	0.689	0.785	0.798	**0.801**
TSR	0.447	0.690	0.925	0.947	**0.980**
Average PTR	0.606	0.648	1.111	1.088	**1.150**
Average OFV			0.197	0.199	**0.172**

resource efficiency with the lowest OFV, indicating substantial resource savings compared to other methods. It also obtains performance (average CS, TSR, average PTR) same with method *Cara*. Overall, the proposed method achieves optimal resource minimization, making it well-suited for constrained maritime environments.

The performance improvement plan results are displayed in Table 6.13. It evaluated each method's effectiveness in enhancing model accuracy and maintaining high performance over time. The proposed method achieves the highest detection quality, surpassing all other methods, indicating superior consistency in meeting the performance threshold. Overall, the performance improvement plan, particularly through the proposed method, offers better model performance than the time minimization and resource minimization plans.

The purpose of these experiments was to evaluate the effectiveness of different optimization plans for model retraining under performance degradation simulations. By using simulated data to mimic the challenges faced in maritime environments, the experiment

aimed to assess how various retraining strategies impact model performance, stability, and resource efficiency over time. The experiment compared three primary optimization plans, time minimization, resource minimization, and performance improvement, each designed to address specific aspects of model degradation and resource constraints. Results demonstrated that the proposed method consistently outperformed existing approaches across all plans, showing a clear advantage in maintaining high detection accuracy and minimizing retraining costs. Specifically, the performance improvement plan yielded the best results in maintaining high confidence scores and threshold satisfaction ratio, confirming that a targeted retraining approach can significantly enhance model performance in maritime environments.

6.6 Summary

In this chapter, the author conducted a series of experiments to validate the proposed methods for consistency evaluation, object detection model optimization, and model retraining. The results of these experiments demonstrate the effectiveness of the proposed methods in addressing key challenges in maritime object detection, including performance degradation, resource constraints, and weather noise. The consistency evaluation experiment confirmed that the proposed method can significantly improve model consistency by extracting low-consistency training data, leading to an improvement in consistency compared to the baseline. The object detection model optimization experiment demonstrated that the proposed method consistently outperformed traditional approaches across all optimization plans, showing a clear advantage in maintaining high detection accuracy and minimizing retraining time/computing resources. These experiment results validate the proposed methods and confirm their potential to address critical issues in maritime object detection, including weather noise, performance degradation, consistency problems, and model retraining optimization with resource constraints. These findings indicate that the proposed approach is feasible and valuable for real-world maritime applications, where balancing detection accuracy and resource constraints is critical.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This dissertation addressed several key challenges associated with object detection in maritime surveillance scenarios, proposing targeted methods to improve the accuracy, resilience, and efficiency of object detection systems deployed on maritime mobile platforms. First, to mitigate the impact of weather-related noise in maritime environments, the author introduced a weather noise removal technique and integrated it into a low-latency object detection system. By implementing this approach, it validated its feasibility through notable improvements in image quality after noise restoration, demonstrating its potential to enhance object detection accuracy in adverse weather conditions. Second, to address the limitations of fine-tuning an all-in-one model for complex maritime environments, the author developed a weather-aware object detection method specifically tailored for maritime scenarios. Using a maritime benchmark dataset (SMD) and a series of adverse weather dataset, the author demonstrated that specialized weather-aware models could reduce the impact of weather noise. This resulted in a significant increase in mean average precision over generic object detection systems, particularly in rainy and hazy conditions, when coupled with noise removal pre-processing. Third,

to maintain consistency in detection results from time-series images collected by mobile maritime devices, the author proposed a consistency measurement method. This method dynamically adjusts a consistency threshold based on image similarity, ensuring accurate performance even with time-series data captured from moving cameras. Experimental results showed that retraining on low-consistency training data improved both the mAP and consistency of object detection in time-series images, validating the effectiveness of this method for dynamic maritime environments. Finally, to address resource-constrained model deployment, the author developed an adaptive optimization method focused on resilience to device mobility, efficient use of onboard resources, and effective model management across distributed maritime platforms. Through experiments, the evaluation results demonstrated that this method improves resource utilization and retraining time, achieving comparable accuracy with significantly lower computational overhead than existing methods.

Future research will aim to expand the applicability and robustness of the proposed methods. First, the weather-aware object detection approach will be deployed in real-world maritime surveillance scenarios to assess its effectiveness in operational conditions. This real-world validation is essential to further enhance model robustness under diverse and unpredictable maritime environments. Additionally, the optimization framework will be extended to incorporate more advanced strategies. For example, minimizing operational costs by reducing price-related factors and minimizing CO_2 emissions by optimizing energy consumption could significantly improve the sustainability and cost-efficiency of maritime object detection systems. These additions will support broader environmental and economic goals while maintaining high-performance standards, creating a more holistic approach to maritime surveillance optimization.

7.2 Perspectives to Maritime Autonomous Surface Ships

The proposed methods are designed to enhance object detection in maritime surveillance scenarios, particularly facilitating the realization of ship automation and autonomy. This section discusses how the proposed methods can be adapted to support the development and deployment of MASS, highlighting the potential benefits and challenges associated with this transformation. MASS is transforming the maritime industry by introducing varying degrees of automation to vessel operations. To facilitate the process of the

Regulatory Scoping Exercise (RSE) defined by the International Maritime Organization (IMO) in May 2021 and published as IMO Circular MSC.1/Cir. 1640 [150]. The degrees of autonomy were organized into four levels as follows:

- **Degree One: Ship with Automated Processes and Decision Support.** Seafarers are on board to operate and control shipboard systems and functions. Some operations may be automated and at times be unsupervised but with seafarers on board ready to take control.
- **Degree Two: Remotely Controlled Ship with Crew Onboard.** The ship is controlled and operated from another location. Seafarers are available on board to take control and to operate the shipboard systems and functions.
- **Degree Three: Remotely Controlled Ship without Crew Onboard.** The ship is controlled and operated from another location. There are no seafarers on board.
- **Degree Four: Fully Autonomous Ship.** The operating system of the ship is able to make decisions and determine actions by itself.

The maritime object detection method proposed in this dissertation addresses key technological and operational challenges faced by MASS, particularly in object detection, environmental awareness, data transmission, and decision-making under adverse and dynamic maritime conditions. The proposed maritime object detection method addresses critical challenges in the implementation of MASS, supporting partial implementation with degree one and two MASS and laying the foundation for higher degrees of autonomy. Specifically, it provides support for the following aspects as follows, with regard to potential gaps or themes that require addressing in [150]:

- **Role and Placement of Master and Crew:** The proposed method enhances situational awareness, enabling the master and crew to make informed decisions in real time. By delivering accurate and consistent object detection results under varying environmental conditions, the system reduces the cognitive burden on onboard personnel and ensures safer navigation. For Crewed Ships (Degrees One and Two), the method acts as an advanced decision-support tool, complementing

human judgment with reliable detection outputs described in Section 4.2, even in adverse weather conditions.

- **Connectivity:** The proposed method is designed to operate efficiently in maritime environments where connectivity is often unstable or intermittent. By performing critical object detection tasks on edge devices onboard, the system minimizes reliance on real-time connectivity to centralized servers. The method supports low-latency decision-making by processing data locally, ensuring uninterrupted performance even in remote sea areas with limited network access.

- **Cybersecurity:** The proposed method utilized SINETStream [118] for secure data transmission and management including data protection, secure access permissions,and enhancing the cybersecurity posture of MASS. The method enhances the cybersecurity posture of MASS by integrating decentralized architecture and secure data management mechanisms.

- **Fundamental Issue Regarding Reduction of Risks Owing to the Absence of Persons Onboard:** The absence of onboard personnel in higher degrees of MASS (Degrees Three and Four) introduces significant risks, such as challenging weather conditions. The weather-aware object detection system described in Section 4.2 ensures accurate detection of objects under adverse weather conditions, reducing the likelihood of accidents and collisions. The retraining optimization framework described in Section 5.2 ensures that the system adapts to changing environmental conditions, maintaining its effectiveness over time.

The proposed maritime object detection method effectively supports key aspects of MASS implementation, including enhancing the role and placement of the master and crew, ensuring reliable operation under constrained connectivity, bolstering cybersecurity, and addressing fundamental challenges associated with unmanned ships. These contributions position the method as a critical enabler for advancing autonomous maritime technologies while maintaining safety and operational efficiency.

Bibliography

[1] The Insight Partners. Collision avoidance & object detection maritime market overview, growth, trends, analysis, research report (2021-2031). `https://www.theinsightpartners.com/reports/collision-avoidance-and-object-detection-maritime-market`, 2024.

[2] Xinyu Zhang, Chengbo Wang, Lingling Jiang, Lanxuan An, and Rui Yang. Collision-avoidance navigation systems for maritime autonomous surface ships: A state of the art survey. *Ocean Engineering*, 235:109380, 2021.

[3] John N Briggs. *Target detection by marine radar*, volume 16. Iet, 2004.

[4] Fabrizio Berizzi, Marco Martorella, and Elisa Giusti. *Radar imaging for maritime observation*. CRC Press, 2018.

[5] Suraya Zainuddin, Idnin Pasya, Nadiy Zaiaami, Asiah Maryam, Raja Syamsul Azmir Raja Abdullah, Megat Syahirul Amin Megat Ali, et al. Maritime radar: a review on techniques for small vessels detection. *Journal of Electrical and Electronic Systems Research (JEESR)*, 14:30–45, 2019.

[6] Ningyuan Su, Xiaolong Chen, Jian Guan, and Yong Huang. Maritime target detection based on radar graph data and graph convolutional network. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5, 2021.

[7] Jerome Williams, Luke Rosenberg, Victor Stamatescu, and Tri-Tan Cao. Maritime radar target detection using convolutional neural networks. In *2022 IEEE Radar Conference (RadarConf22)*, pages 1–6. IEEE, 2022.

[8] Jiaying Lin, Phillip Diekmann, Christian-Eike Framing, Rene Zweigel, and Dirk Abel. Maritime environment perception based on deep learning. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15487–15497, 2022.

[9] Nikola Lopac, Irena Jurdana, Adrian Brnelić, and Tomislav Krljan. Application of laser systems for detection and ranging in the modern road transportation and maritime sector. *Sensors*, 22(16):5946–5973, 2022.

[10] Ashiq Anjum, Tariq Abdullah, M Fahim Tariq, Yusuf Baltaci, and Nick Antonopoulos. Video stream analysis in clouds: An object detection and classification framework for high performance video analytics. *IEEE Transactions on Cloud Computing*, 7(4):1152–1167, 2016.

[11] Muhammad Usman Yaseen, Ashiq Anjum, Omer Rana, and Richard Hill. Cloud-based scalable object detection and classification in video streams. *Future Generation Computer Systems*, 80:286–298, 2018.

[12] Jin-Hwan Kim, Chul Lee, Jae-Young Sim, and Chang-Su Kim. Single-image deraining using an adaptive nonlocal means filter. In *2013 IEEE international conference on image processing*, pages 914–917. IEEE, 2013.

[13] Jérémie Bossu, Nicolas Hautiere, and Jean-Philippe Tarel. Rain or snow detection in image sequences through use of a histogram of orientation of streaks. *International journal of computer vision*, 93:348–367, 2011.

[14] Li-Wei Kang, Chia-Wen Lin, Che-Tsung Lin, and Yu-Chen Lin. Self-learning-based rain streak removal for image/video. In *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1871–1874. IEEE, 2012.

[15] C Ramya and S Subha Rani. Rain removal in image sequence using sparse coding. In *International conference on intelligent robotics, automation, and manufacturing*, pages 361–370. Springer, 2012.

[16] Duan-Yu Chen, Chien-Cheng Chen, and Li-Wei Kang. Visual depth guided color image rain streaks removal using sparse coding. *IEEE transactions on circuits and systems for video technology*, 24(8):1430–1455, 2014.

[17] Xianhui Zheng, Yinghao Liao, Wei Guo, Xueyang Fu, and Xinghao Ding. Single-image-based rain and snow removal using multi-guided filter. In *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20*, pages 258–265. Springer, 2013.

[18] Jing Xu, Wei Zhao, Peng Liu, and Xianglong Tang. An improved guidance image based method to remove rain and snow in a single image. *Computer and Information Science*, 5(3):49, 2012.

[19] Daniel Zoran and Yair Weiss. Natural images, gaussian mixtures and dead leaves. *Advances in Neural Information Processing Systems*, 25:1736–1744, 2012.

[20] YiChang Shih, Dilip Krishnan, Fredo Durand, and William T Freeman. Reflection removal using ghosting cues. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3193–3201, 2015.

[21] Yu Li, Robby T Tan, Xiaojie Guo, Jiangbo Lu, and Michael S Brown. Rain streak removal using layer priors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2736–2744, 2016.

[22] Dean Huang, Liwei Kang, Minchun Yang, Chiawen Lin, and Yuchiang Frank Wang. Context-aware single image rain removal. In *2012 IEEE International Conference on Multimedia and Expo*, pages 164–169. IEEE, 2012.

[23] Xin Fan, Yi Wang, Xianxuan Tang, Renjie Gao, and Zhongxuan Luo. Two-layer gaussian process regression with example selection for image dehazing. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2505–2517, 2016.

[24] Yun-Fu Liu, Da-Wei Jaw, Shih-Chia Huang, and Jenq-Neng Hwang. Desnownet: Context-aware deep network for snow removal. *IEEE Transactions on Image Processing*, 27(6):3064–3073, 2018.

[25] Mingkang Chen, Jingtao Sun, Kazushige Saga, Tomota Tanjo, and Kento Aida. An adaptive noise removal tool for iot image processing under influence of weather conditions. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 655–656, 2020.

[26] Ruoteng Li, Robby T Tan, and Loong-Fah Cheong. All in one bad weather removal using architectural search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3175–3185, 2020.

[27] Jeya Maria Jose Valanarasu, Rajeev Yasarla, and Vishal M Patel. Transweather: Transformer-based restoration of images degraded by adverse weather conditions.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2353–2363, 2022.

[28] Dongwei Ren, Wangmeng Zuo, Qinghua Hu, Pengfei Zhu, and Deyu Meng. Progressive image deraining networks: A better and simpler baseline. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3937–3946, 2019.

[29] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep stacked hierarchical multi-patch network for image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5978–5986, 2019.

[30] He Zhang, Vishwanath Sindagi, and Vishal M Patel. Image de-raining using a conditional generative adversarial network. *IEEE transactions on circuits and systems for video technology*, 30(11):3943–3956, 2019.

[31] Ruoteng Li, Loong-Fah Cheong, and Robby T Tan. Heavy rain image restoration: Integrating physics model and conditional adversarial learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1633–1642, 2019.

[32] Yinglong Wang, Haokui Zhang, Yu Liu, Qinfeng Shi, and Bing Zeng. Gradient information guided deraining with a novel network and adversarial training. *arXiv preprint arXiv:1910.03839*, 2019.

[33] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[34] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing*, volume 1, pages 900–903. IEEE, 2002.

[35] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[36] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[37] Nils Tijtgat, Wiebe Van Ranst, Toon Goedeme, Bruno Volckaert, and Filip De Turck. Embedded real-time object detection for a uav warning system. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 2110–2118, 2017.

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[39] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[40] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[41] R Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[42] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

[45] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[46] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, pages 1–13. IEEE, 2017.

[47] Xiao Zeng, Biyi Fang, Haichen Shen, and Mi Zhang. Distream: scaling live video analytics with workload-adaptive distributed edge intelligence. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 409–421, 2020.

[48] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *Computer*, 50(10):58–67, 2017.

[49] Christopher Canel, Thomas Kim, Giulio Zhou, Conglong Li, Hyeontaek Lim, David G Andersen, Michael Kaminsky, and Subramanya Dulloor. Scaling video analytics on constrained edge nodes. *Proceedings of Machine Learning and Systems*, 1:406–417, 2019.

[50] Lixiang Ao, Liz Izhikevich, Geoffrey M. Voelker, and George Porter. Sprocket: A serverless video processing framework. In *Proceedings of the ACM Symposium on Cloud Computing*, page 263–274. Association for Computing Machinery, 2018.

[51] Miao Zhang, Yifei Zhu, Cong Zhang, and Jiangchuan Liu. Video processing with serverless computing: A measurement study. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, page 61–66. Association for Computing Machinery, 2019.

[52] Lijing Wang, Dipanjan Ghosh, Maria Gonzalez Diaz, Ahmed Farahat, Mahbubul Alam, Chetan Gupta, Jiangzhuo Chen, and Madhav Marathe. Wisdom of the ensemble: Improving consistency of deep learning models. *Advances in Neural Information Processing Systems*, 33:19750–19761, 2020.

[53] Caleb Tung, Abhinav Goel, Fischer Bordwell, Nick Eliopoulos, Xiao Hu, Yung-Hsiang Lu, and George K Thiruvathukal. Why accuracy is not enough: The need for consistency in object detection. *IEEE MultiMedia*, 29(3):8–16, 2022.

[54] Ling Liu, Wenqi Wei, Ka-Ho Chow, Margaret Loper, Emre Gursoy, Stacey Truex, and Yanzhao Wu. Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness. In *2019 IEEE 16th international conference on mobile ad hoc and sensor systems (MASS)*, pages 274–282. IEEE, 2019.

[55] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30:1195–1204, 2017.

[56] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.

[57] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[58] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

[59] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan O Arik, Larry S Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling budget. In *Computer Vision–ECCV 2020. Lecture Notes in Computer Science, vol 12355.*, pages 510–526, 2020.

[60] Jisoo Jeong, Seungeui Lee, Jeesoo Kim, and Nojun Kwak. Consistency-based semi-supervised learning for object detection. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 10759–10768, 2019.

[61] Younkwan Lee, Jihyo Jeon, Yeongmin Ko, Byunggwan Jeon, and Moongu Jeon. Task-driven deep image enhancement network for autonomous driving in bad weather. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13746–13753. IEEE, 2021.

[62] M Jehanzeb Mirza, Marc Masana, Horst Possegger, and Horst Bischof. An efficient domain-incremental learning approach to drive in all weather conditions.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3001–3011, 2022.

[63] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14765–14775, 2022.

[64] Huayi Zhou, Fei Jiang, and Hongtao Lu. Ssda-yolo: Semi-supervised domain adaptive yolo for cross-domain object detection. *Computer Vision and Image Understanding*, 229:103649, 2023.

[65] Ryan Wen Liu, Weiqiao Yuan, Xinqiang Chen, and Yuxu Lu. An enhanced cnn-enabled learning method for promoting ship detection in maritime surveillance system. *Ocean Engineering*, 235:109435, 2021.

[66] Haichen Shen, Seungyeop Han, Matthai Philipose, and Arvind Krishnamurthy. Fast video classification via adaptive cascading of deep models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3646–3654, 2017.

[67] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[68] Daniel Rivas, Francesc Guim, Jordà Polo, Pubudu M Silva, Josep Ll Berral, and David Carrera. Towards automatic model specialization for edge video analytics. *Future Generation Computer Systems*, 134:399–413, 2022.

[69] Peizhen Guo, Bo Hu, and Wenjun Hu. Mistify: Automating DNN model porting for On-Device inference at the edge. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 705–719. USENIX Association, April 2021. ISBN 978-1-939133-21-2.

[70] Tiffany Yu-Han Chen, Lenin Ravindranath, Shuo Deng, Paramvir Bahl, and Hari Balakrishnan. Glimpse: Continuous, real-time object recognition on mobile devices. In *Proceedings of the 13th ACM conference on embedded networked sensor systems*, pages 155–168, 2015.

[71] Yuanqi Li, Arthi Padmanabhan, Pengzhan Zhao, Yufei Wang, Guoqing Harry Xu, and Ravi Netravali. Reducto: On-camera filtering for resource-efficient real-time video analytics. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 359–376, 2020.

[72] Mengwei Xu, Mengze Zhu, Yunxin Liu, Felix Xiaozhu Lin, and Xuanzhe Liu. Deepcache: Principled cache for mobile deep vision. In *Proceedings of the 24th annual international conference on mobile computing and networking*, pages 129–144, 2018.

[73] Thomas Verelst and Tinne Tuytelaars. Blockcopy: High-resolution video processing with block-sparse feature propagation and online policies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5158–5167, 2021.

[74] Thomas Verelst and Tinne Tuytelaars. Segblocks: Block-based dynamic resolution networks for real-time segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2400–2411, 2022.

[75] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodik, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 115–131. IEEE, 2018.

[76] Haoyu Zhang, Ganesh Ananthanarayanan, Peter Bodik, Matthai Philipose, Paramvir Bahl, and Michael J Freedman. Live video analytics at scale with approximation and {Delay-Tolerance}. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392, 2017.

[77] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 conference of the ACM special interest group on data communication*, pages 253–266, 2018.

[78] Ben Zhang, Xin Jin, Sylvia Ratnasamy, John Wawrzynek, and Edward A Lee. Awstream: Adaptive wide-area streaming analytics. In *Proceedings of the 2018*

Conference of the ACM Special Interest Group on Data Communication, pages 236–252, 2018.

[79] Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Ylä-Jääski. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4150–4161, 2018.

[80] Ali Pesaranghader, Herna L Viktor, and Eric Paquet. A framework for classification in data streams using multi-strategy learning. In *Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings 19*, pages 341–355. Springer, 2016.

[81] Vinicius Souza, Tiago Pinho, and Gustavo Batista. Evaluating stream classifiers with delayed labels information. In *2018 7th Brazilian conference on intelligent systems (BRACIS)*, pages 408–413. IEEE, 2018.

[82] Tsuwang Hsieh, Behnaz Arzani, and Ankur Mallick. Data drift mitigation in machine learning for large-scale systems, November 17 2022. US Patent App. 17/322,184.

[83] James Bergstra, Brent Komer, Chris Eliasmith, Daniel L. K. Yamins, and David D. Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8:014008, 2015.

[84] Daniel S Soper. Greed is good: Rapid hyperparameter optimization and model selection using greedy k-fold cross validation. *Electronics*, 10(16):1973, 2021.

[85] Mohammed Qaraad, Souad Amjad, Ibrahim I. M. Manhrawy, Hanaa Fathi, Bayoumi Ali Hassan, and Passent El Kafrawy. A hybrid feature selection optimization model for high dimension data classification. *IEEE Access*, 9:42884–42895, 2021.

[86] Jonathan Lee, G. Tucker, Ofir Nachum, and Bo Dai. Model selection in batch policy optimization. *ArXiv*, abs/2112.12320, 2021.

[87] Ben Taylor, Vicent Sanz Marco, Willy Wolff, Yehia Elkhatib, and Zheng Wang. Adaptive deep learning model selection on embedded systems. *ACM Sigplan Notices*, 53(6):31–43, 2018.

[88] Fabian Hinder, Valerie Vaquet, Johannes Brinkrolf, and Barbara Hammer. On the hardness and necessity of supervised concept drift detection. In *ICPRAM*, pages 164–175, 2023.

[89] Ananth Mahadevan and Michael Mathioudakis. Cost-aware retraining for machine learning. *Knowledge-Based Systems*, 293(111610):1–16, 2024.

[90] Xian Li, Suzhi Bi, and Hui Wang. Optimizing resource allocation for joint ai model training and task inference in edge intelligence systems. *IEEE Wireless Communications Letters*, 10(3):532–536, 2020.

[91] Wenyu Zhang, Sherali Zeadally, Wei Li, Haijun Zhang, Jingyi Hou, and Victor C. M. Leung. Edge ai as a service: Configurable model deployment and delay-energy optimization with result quality constraints. *IEEE Transactions on Cloud Computing*, 11:1954–1969, 2023.

[92] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13693–13696, 2020.

[93] Jaykumar Kasundra, Claudia Schulz, Melicaalsadat Mirsafian, and Stavroula Skylaki. A framework for monitoring and retraining language models in real-world applications. *arXiv preprint arXiv:2311.09930*, 2023.

[94] Xin Wang, Xin Zhang, Hangcheng Zhu, Qiong Wang, and Chen Ning. An effective algorithm for single image fog removal. *Mobile Networks and Applications*, pages 1–9, 2019.

[95] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[96] Vijay Gupta. Keras multiclass image classification. `https://github.com/vijayg15/Keras-MultiClass-Image-Classification`, 2020.

[97] Vijay Gupta. Multiclass weather dataset. `https://www.kaggle.com/vijaygiitk/multiclass-weather-dataset`, 2020.

[98] Dilip K Prasad, Deepu Rajan, Lily Rachmawati, Eshan Rajabally, and Chai Quek. Video processing from electro-optical sensors for object detection and tracking in a

maritime environment: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 18(8):1993–2016, 2017.

[99] Ricardo Ribeiro, Gonçalo Cruz, Jorge Matos, and Alexandre Bernardino. A data set for airborne maritime surveillance environments. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(9):2720–2732, 2017.

[100] Donato Cafarelli, Luca Ciampi, Lucia Vadicamo, Claudio Gennaro, Andrea Berton, Marco Paterni, Chiara Benvenuti, Mirko Passera, and Fabrizio Falchi. Mobdrone: A drone video dataset for man overboard rescue. In *Image Analysis and Processing–ICIAP 2022: 21st International Conference, Lecce, Italy, May 23–27, 2022, Proceedings, Part II*, pages 633–644. Springer, 2022.

[101] Leon Amadeus Varga, Benjamin Kiefer, Martin Messmer, and Andreas Zell. Seadronessee: A maritime benchmark for detecting humans in open water. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2260–2270, 2022.

[102] Hyeon-Cheol Shin, Kwang-Il Lee, and Chang-Eun Lee. Data augmentation method of object detection for deep learning in maritime image. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 463–466. IEEE, 2020.

[103] Eric Luc Adrien Gastineau. *Domain adaptation via data augmentation.* PhD thesis, Georgia Institute of Technology, 2020.

[104] Maxime Tremblay, Shirsendu Sukanta Halder, Raoul De Charette, and Jean-François Lalonde. Rain rendering for evaluating and improving robustness to bad weather. *International Journal of Computer Vision*, 129:341–360, 2021.

[105] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[106] Kaihao Zhang, Dongxu Li, Wenhan Luo, and Wenqi Ren. Dual attention-in-attention model for joint rain streak and raindrop removal. In *IEEE Transactions on Image Processing*, pages 7608–7619, 2021.

[107] Codruta O. Ancuti, Cosmin Ancuti, Radu Timofte, and Christophe De Vleeschouwer. O-haze: a dehazing benchmark with real hazy and haze-free outdoor images. In *IEEE Conference on Computer Vision and Pattern Recognition, NTIRE Workshop*, NTIRE CVPR'18, pages 754–762, 2018.

[108] Yulong Zhang, Jingtao Sun, Mingkang Chen, Qiang Wang, Yuan Yuan, and Rongzhe Ma. Multi-weather classification using evolutionary algorithm on efficientnet. In *Proceedings of the 5th International Workshop on Mobile and Pervasive Internet of Things (PerIoT 2021)]*, page 546–551, 2021.

[109] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[110] Pu Wang, Thomas Weise, and Raymond Chiong. Novel evolutionary algorithms for supervised classification problems: an experimental study. *Evolutionary Intelligence*, 4(1):3–16, 2011.

[111] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[112] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31:6389–6399, 2018.

[113] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[114] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30:2990–2999, 2017.

[115] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[116] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental learning of object detectors without catastrophic forgetting. In *Proceedings of the IEEE international conference on computer vision*, pages 3400–3409, 2017.

[117] Atsuko Takefusa, Jingtao Sun, Ikki Fujiwara, Hiroshi Yoshida, Kento Aida, and Calton Pu. Sinetstream: Enabling research iot applications with portability, security and performance requirements. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 482–492. IEEE, 2021.

[118] SINETStream. `https://www.sinetstream.net/`, 2025.

[119] SINETStream GitHub page. `https://github.com/nii-gakunin-cloud/sinetstream`, 2025.

[120] Mingkang Chen, Jingtao Sun, Kento Aida, and Atsuko Takefusa. Weather-aware object detection method for maritime surveillance systems. *Future Generation Computer Systems*, 151:111–123, 2024.

[121] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[122] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[123] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. Fsim: A feature similarity index for image quality assessment. *IEEE transactions on Image Processing*, 20 (8):2378–2386, 2011.

[124] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.

[125] Wenhan Yang, Robby T Tan, Jiashi Feng, Jiaying Liu, Zongming Guo, and Shuicheng Yan. Deep joint rain detection and removal from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1357–1366, 2017.

[126] Xueyang Fu, Jiabin Huang, Delu Zeng, Yue Huang, Xinghao Ding, and John Paisley. Removing rain from single images via a deep detail network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3855–3863, 2017.

[127] Rajeev Yasarla, Vishwanath A Sindagi, and Vishal M Patel. Syn2real transfer learning for image deraining using gaussian processes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2726–2736, 2020.

[128] Xueyang Fu, Borong Liang, Yue Huang, Xinghao Ding, and John Paisley. Lightweight pyramid networks for image deraining. *IEEE transactions on neural networks and learning systems*, 31(6):1794–1807, 2019.

[129] Xueyang Fu, Jiabin Huang, Xinghao Ding, Yinghao Liao, and John Paisley. Clearing the skies: A deep network architecture for single-image rain removal. *IEEE Transactions on Image Processing*, 26(6):2944–2956, 2017.

[130] Anish Mittal, Anush K Moorthy, and Alan C Bovik. Blind/referenceless image spatial quality evaluator. In *2011 conference record of the forty fifth asilomar conference on signals, systems and computers (ASILOMAR)*, pages 723–727. IEEE, 2011.

[131] Lixiong Liu, Bao Liu, Hua Huang, and Alan Conrad Bovik. No-reference image quality assessment based on spatial and spectral entropies. *Signal Processing: Image Communication*, 29(8):856–863, 2014.

[132] Anish Mittal, Rajiv Soundararajan, and Alan C Bovik. Making a completely blind image quality analyzer. *IEEE Signal processing letters*, 20(3):209–212, 2012.

[133] Rain image datasets. `https://1drv.ms/f/s!AqLfQqtZ6GwGgep-hgjLxkov2SSZ3g`, 2021.

[134] Ali Farhadi and Joseph Redmon. Yolov3: An incremental improvement. In *Computer vision and pattern recognition*, volume 1804, pages 1–6. Springer Berlin/Heidelberg, Germany, 2018.

[135] Singapore Maritime Dataset Homepage. `https://sites.google.com/site/dilipprasad/home/singapore-maritime-dataset`, 2023.

[136] VisoComputacional. Datasetships dataset. `https://universe.roboflow.com/visocomputacional/datasetships`, 2023.

[137] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[138] Pytorch hub. `https://pytorch.org/hub/`, 2025.

[139] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v7.0 - yolov5 sota realtime instance segmentation. `https://github.com/ultralytics/yolov5`, November 2022.

[140] Martin G Larson. Analysis of variance. *Circulation*, 117(1):115–121, 2008.

[141] Peter Morales, Tzofi Klinghoffer, and Seung Jae Lee. Feature forwarding for efficient single image dehazing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 2078–2085, 2019.

[142] Lars St, Svante Wold, et al. Analysis of variance (anova). *Chemometrics and intelligent laboratory systems*, 6(4):259–272, 1989.

[143] Bin Zhao, Xuelong Li, Xiaoqiang Lu, and Zhigang Wang. A cnn–rnn architecture for multi-label weather recognition. *Neurocomputing*, 322:47–57, 2018.

[144] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[145] Yuxin Kong, Peng Yang, and Yan Cheng. Edge-assisted on-device model update for video analytics in adverse environments. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9051–9060, 2023.

[146] Corey Smith. Boat technician films storm at cape cod. https://youtu.be/w2tdO1LJR3Y?si=fNqzA-5LMzEhr8EG, 2019.

[147] Life in videos. Sailing in dense fog. https://youtu.be/Pq37Ot8qDhc?si=4vdRaU67HgdV6m6W, 2018.

[148] Abhilasha Nanda, Sung Won Cho, Hyeopwoo Lee, and Jin Hyoung Park. Kolomverse: Korea open large-scale image dataset for object detection in the maritime universe. *IEEE Transactions on Intelligent Transportation Systems*, 25(12):20832–20840, 2024.

[149] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Coco api. https://github.com/cocodataset/cocoapi, 2022.

[150] International Maritime Organization. Outcome of the regulatory scoping exercise for the use of maritime autonomous surface ships (mass). https://www.imo.org, 2021.