NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ROBUST MACHINE LEARNING FOR COMPUTER VISION IN NAVAL APPLICATION

by

Gabriel Custodio Rangel

June 2023

Thesis Advisor:	Eric C. Eckstrand
Co-Advisor:	Johannes O. Royset
Second Reader:	Robert L. Bassett

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE ROBUST MACHINE LEARNING FOR COMPUTER VISION IN NAVAL APPLICATION	**5. FUNDING NUMBERS**
6. AUTHOR(S) Gabriel Custodio Rangel	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

This thesis proposes the development of a resilient machine learning algorithm that can classify naval images for surveillance, search, and detection operations in vast coastal areas. However, real-world datasets may be affected by label noise introduced either through random inaccuracies or deliberate adversarial attacks, both of which can negatively impact the accuracy of machine learning models. Our innovative approach employs Rockafellian Risk Minimization (RRM) to combat label noise contamination. Unlike existing methodologies reliant on extensively cleaned datasets, our two-step process involves adjusting neural network weights and manipulating data point nominal probabilities to isolate potential data corruption effectively. This technique reduces the dependency on meticulous data cleaning, thereby promoting more efficient and time-effective data processing. To validate the efficacy and reliability of the proposed model, we apply RRM in several parameter configurations to naval environment datasets and assess its classification accuracy against traditional methods. By leveraging the proposed model, we aim to bolster the robustness of ship detection models, paving the way for a novel, reliable tool that could improve automated maritime surveillance systems.

14. SUBJECT TERMS computer vision, neural networks, stochastic gradient descent, Rockafellian risk minimization	**15. NUMBER OF PAGES** 93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ROBUST MACHINE LEARNING FOR COMPUTER VISION IN NAVAL APPLICATION

Gabriel Custodio Rangel
Capitão de Corveta, Brazilian Navy
BNS, Brazilian Naval Academy, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2023

Approved by: Eric C. Eckstrand
Advisor

Johannes O. Royset
Co-Advisor

Robert L. Bassett
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis proposes the development of a resilient machine learning algorithm that can classify naval images for surveillance, search, and detection operations in vast coastal areas. However, real-world datasets may be affected by label noise introduced either through random inaccuracies or deliberate adversarial attacks, both of which can negatively impact the accuracy of machine learning models. Our innovative approach employs Rockafellian Risk Minimization (RRM) to combat label noise contamination. Unlike existing methodologies reliant on extensively cleaned datasets, our two-step process involves adjusting neural network weights and manipulating data point nominal probabilities to isolate potential data corruption effectively. This technique reduces the dependency on meticulous data cleaning, thereby promoting more efficient and time-effective data processing. To validate the efficacy and reliability of the proposed model, we apply RRM in several parameter configurations to naval environment datasets and assess its classification accuracy against traditional methods. By leveraging the proposed model, we aim to bolster the robustness of ship detection models, paving the way for a novel, reliable tool that could improve automated maritime surveillance systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

ADH	alternating direction heuristic
AIS	automatic identification system
ANN	artificial neural network
BN	Brazilian Navy
CNN	convolutional neural network
CV	computer vision
DAC	deep abstaining classifier
EEZ	exclusive economic zone
ERM	empirical risk minimization
GPU	graphic processing unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
L2R	learning-to-reject
MaCVi	Maritime Computer Vision workshop
MASATI	Maritime Satellite Imagery
ML	machine learning
MNIST	Modified National Institute of Standards and Technology
NN	neural network
OOD	out-of-distribution
OR	operations research
ReLu	rectified linear units
RGB	red, green, blue
RRM	Rockafellian risk minimization
SaR	search and rescue
SGD	stochastic gradient descent
SisGAAz	Blue Amazon Management System
ST&I	science, technology, and innovation
SVM	support vector machines

UAV	unmanned aerial vehicles
VGG	Visual Geometry Group
VTS	vessel traffic services
YOLO	You Only Look Once

EXECUTIVE SUMMARY

Machine learning (ML) is rapidly advancing, enabling machines to make decisions based on data analysis. A specialized sector of this field, computer vision (CV), uses advanced algorithms to interpret visual information, transforming industries such as automotive, medical, security, and military by creating innovative opportunities. In the military sector, these tools have proven beneficial in improving decision-making, situational awareness, surveillance capabilities, supporting operations, and facilitating the effective use of autonomous systems in complex environments.

Our research is primarily focused on applying CV principles to naval applications, specifically solving a binary classification problem that indicates the presence or absence of ships. This forms a vital part of a broader surveillance tool and employs a novel strategy called Rockafellian Risk Minimization (RRM) [1]. The RRM method is designed to combat the challenges associated with label corruption in datasets inherent to such complex and dynamic environments as maritime surveillance. Central to our approach is the Alternating Direction Heuristic (ADH), a two-pronged strategy that sequentially optimizes different sets of variables. This iterative two-step process adjusts the neural network weights and manipulates the data point probabilities, effectively isolating potential data corruption. The result is a more robust and accurate maritime surveillance and detection system that enhances decision-making and situational awareness in naval operations.

Our evaluation uses two diverse datasets, the Airbus Ship Detection (AIRBUS) [2] and Maritime Satellite Imagery (MASATI) [3]. To test our methodology's robustness, we progressively increased label corruption levels in these datasets and observed how this affects model performance.

Our research utilizes two strategies within the ADH process: *w*-optimization and *u*-optimization. In the *w*-optimization phase, we trial two distinct Neural Network (NN) optimizers, Adam [4] and Stochastic Gradient Descent (SGD) [5, Section 3G], to adjust neural network weights. The *u*-optimization phase involves implementing either ADH-

LP (linear programming) or ADH-SUB (subgradient) algorithms to modify the probabilities of each data point and effectively isolate potential data corruption.

ADH-LP leverages linear programming for computational optimization, providing globally optimal solutions but requiring more processing time. On the other hand, ADH-SUB employs a faster subgradient approach better suited for larger datasets or limited computational resources. The primary objective is not to enhance performance through architectural adjustments but to demonstrate how the RRM approach can offer benefits over the traditional ERM approach, particularly in handling data corruption and enhancing model performance.

Our study's RRM approach to train NNs consistently outperformed or matched the ERM approach, irrespective of the dataset used (MASATI or AIRBUS). The ADH-LP and ADH-SUB algorithms under RRM demonstrated remarkable resilience to data corruption while maintaining high-performance levels, with ADH-LP consistently emerging as the superior performer. Overall, our results establish RRM as a robust and resilient approach for handling some level of data corruption.

In conclusion, our innovative approach utilizing RRM offers a promising solution for reducing dependency on label-correct data, thereby enabling the development of more robust ship detection models. This research takes strides toward improving automatic ship detection and overall maritime security. By effectively handling data corruption and testing innovative methods, we improve the capabilities of maritime surveillance systems to monitor coastal and delimited sea areas efficiently.

References

[1] J. O. Royset, L. L. Chen, and E. Eckstrand, "Rockafellian Relaxation in Optimization under Uncertainty: Asymptotically Exact Formulations." *arXiv*, 2022 [Online]. Available: https://arxiv.org/abs/2204.04762

[2] "Airbus Ship Detection Challenge." https://kaggle.com/competitions/airbus-ship-detection (accessed May 30, 2023).

[3] A.-J. Gallego, A. Pertusa, and P. Gil, "Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks," *Remote Sens.*, vol. 10, no. 4, p. 511, Mar. 2018, doi: 10.3390/rs10040511.

[4] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. Accessed: May 08, 2023 [Online]. Available: http://arxiv.org/abs/1412.6980

[5] J. O. Royset and R. J.-B. Wets, *An Optimization Primer*. in Springer Series in Operations Research and Financial Engineering. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-76275-9

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my family. To my wife, Larissa Gama, for her unwavering support during this period. Without her, none of this would have been possible. Additionally, I am thankful to my sons Henrique and Jose for the love they have shown me throughout my life.

I am thankful to my parents, Alecia and Clodomiro, for their incredibly supporting and encouraging me throughout my entire life, especially during my studies.

To my thesis advisor, Eric Eckstrand, who has shared invaluable knowledge and expertise with me on the intricacies of machine learning and Python coding. Thank you for your guidance during my thesis research.

To my co-advisor, Johannes Royset. Thank you for your unwavering assistance and patience. I truly appreciate the support and encouragement he provided during challenging times, and I will always remember it. I also want to express my gratitude for your guidance throughout my graduate studies.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The field of machine learning (ML) is expanding quickly, allowing machines to learn and make data-based decisions. Within this broad field, computer vision (CV) is a specialized area that uses advanced algorithms to help machines interpret visual information. Using advanced mathematical and computational techniques, CV enables machines to "see" and interpret visual information in previously impossible ways. From self-driving cars, medical imaging, security, and the military CV is transforming and creating new opportunities for innovation and discovery.

In the military, these tools have shown promise in improving decision-making, helping with situational awareness, enhancing surveillance capabilities, supporting military operations, and enabling autonomous systems to work effectively in complex and dynamic environments.

Brazil has an extensive coast comprising 3.6 million square kilometers, covering 200 nautical miles within the Brazilian Exclusive Economic Zone (EEZ) [1]. The Brazilian Navy (BN) has been designated to assure Brazilian sovereignty in this area.

One crucial aspect of maritime surveillance is automatic ship detection. However, existing approaches depend on extensive and meticulously cleaned datasets, and collecting and processing high-quality data can take much time and effort.

This study investigates neural network (NN) technologies to efficiently monitor coastal and delimited sea areas. We aim to develop an approach that reduces dependency on such data and promotes more robust ship detection models.

A. BACKGROUND

1. The Blue Amazon

Given its immense Atlantic coastline that spans almost 7,500 kilometers, Brazil is interested in any progress or alterations concerning the Atlantic Ocean. In 2004, paralleling the size and biodiversity of the Green Amazon rainforest in the northwest Brazilian region, the BN introduced the term "Blue Amazon," which refers to a vast area of the Atlantic

Ocean that belongs to Brazil and can be explored by them. It covers the ocean's surface, the waters above the ocean floor, the soil, and the marine subsoil from the coastline to the edge of Brazil's Continental Shelf. This significant and extensive region was officially registered as a trademark in 2010 [1]. Figure 1 shows the dimensions of both Amazons.

The Blue Amazon's strategic and economic importance is undeniable due to its abundant resources, including fishing, marine biodiversity, and resources such as minerals, oil, and natural gas reserves [2].

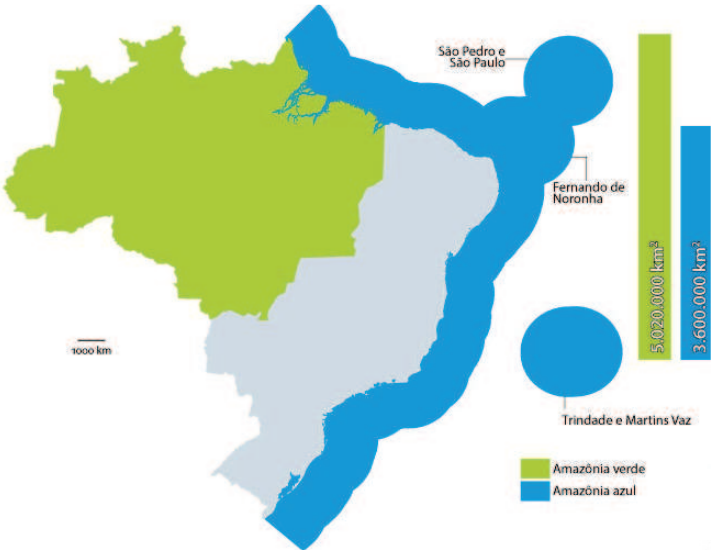

Figure 1. Amazons' dimensions. Adapted from [3].

Safeguarding this vast area called Blue Amazon is challenging. If protective measures are not sufficiently robust, criminal activities such as piracy, smuggling, and illegal disposal of pollutants will persist. The BN conducts naval patrols in Brazil's Jurisdictional Waters to prevent unlawful activities and prosecute offenders [4]. To that end, the BN has created strategic initiatives to protect this vast region. One such program is the Blue Amazon Management System (SisGAAz), which aims to provide adequate measures for its protection [5]. Figure 2 illustrates the SisGAAz system.

SisGAAz is regarded as one of the BN's boldest endeavors and is a prominent feature of its strategic programs, known for its heavy reliance on scientific and

technological advancements. The primary objective of this project is to employ satellite technology, multi-platform systems, radars, and sensors for continuous monitoring and management of the Brazilian jurisdictional waters and the Search and Rescue (SaR) region, thereby integrating data and decision-making networks to oversee the “Blue Amazon” area [6].

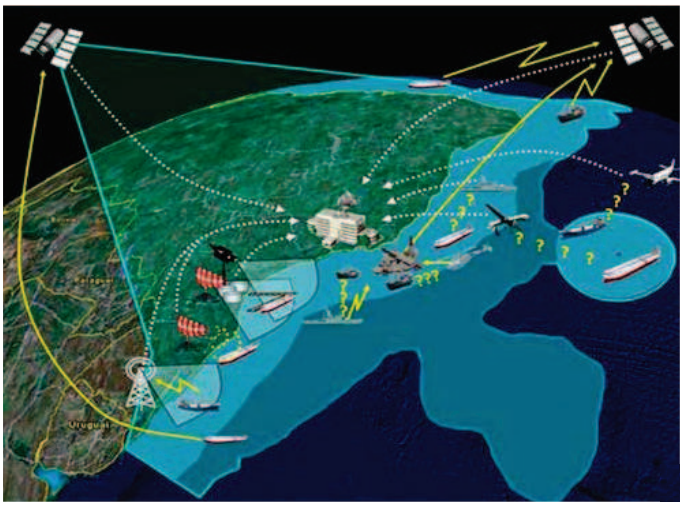

Figure 2. SisGAAz system. Adapted from [2].

The investigation conducted in this thesis is intended to support the surveillance objectives of SisGAAz.

2. Machine Learning Concepts

In the upcoming subsections, we provide an overview of some concepts at a high level, aiming to introduce standard notation, and key techniques employed in this study.

ML focuses on developing models by computer algorithms, which are mathematical representations or rules, that allow computers to learn from data inputs and make predictions or decisions based on this previous learning.

We can classify the three main types of ML algorithms: unsupervised, supervised, and reinforcement learning. Unsupervised learning involves the algorithm identifying patterns or structures in data without labeled information. In contrast, in supervised

learning, the algorithm learns from labeled data, where input-output pairs are provided, enabling it to understand the relationship between inputs and outputs. The third type, reinforcement learning, entails training an algorithm to make decisions in an environment based on feedback in the form of rewards or penalties [7]. Figure 3 shows a diagram of three ML main types.

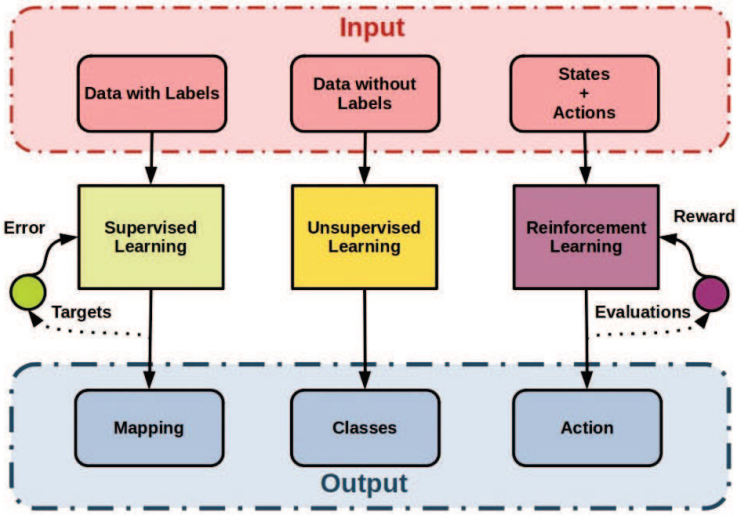

Figure 3. Machine learning types. Source: [8].

a. Regression and Classification

Regarding ML and statistical modeling, there are two essential tasks: Regression and Classification. Both tasks involve predicting outcomes or labels of data based on unique features. We can see these two tasks' differences by looking at the generated output or response variable type.

Regression is a technique to estimate a continuous target variable based on one or more input features. Essentially, it tries to find the best-fit line or curve that minimizes the difference between predicted and actual values of the target variable.

On the other hand, classification produces a categorical or label output. The goal of classification is to learn a boundary that differentiates classes in the data and separates them.

Understanding the differences between these two tasks is crucial so that the appropriate method can be chosen for the intended application [9]. The plots shown in Figure 4 illustrate both tasks.

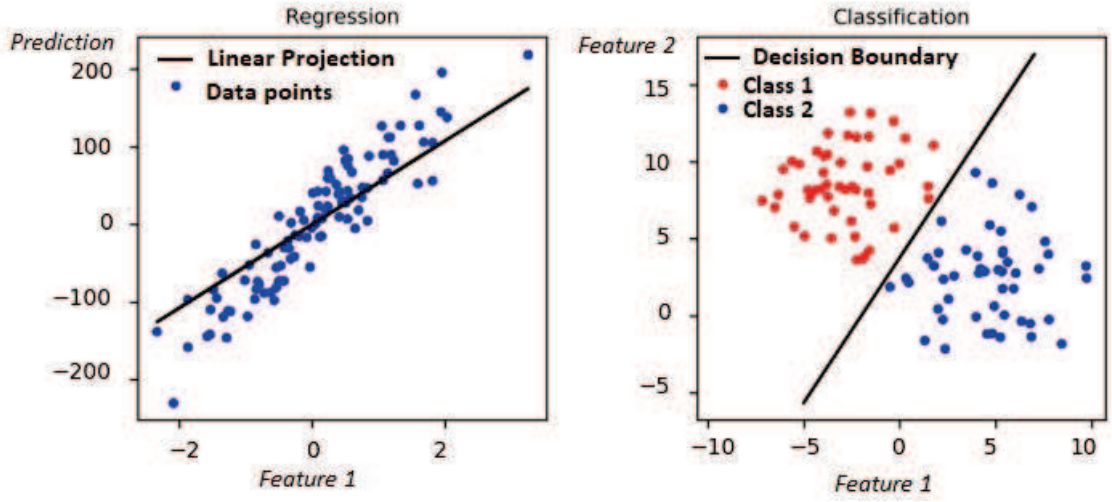

Figure 4. Regression and classification plots. Adapted from [10].

b. Types of Classification

We can separate classification tasks into two main types, distinguished by the number of classes or categories in the target variable.

Binary classification is an ML task that aims to classify data points into one of two possible categories or classes. The two classes are often denoted as class 1 and class 0, or $+1$ and -1 [7]. Solving this fundamental problem has practical applications in a wide range of settings, such as predicting whether a person is likely to purchase more items or not based on income and credit history, classifying an email as spam or not, or determining the presence or absence of some object in a digital image.

In binary classification, the input data points represent features or attributes. The goal is to learn a model to predict the correct class label for new, unseen data points. The model undergoes training using pre-labeled data, where each data point is assigned a known class label. It uses this data to learn the patterns or relationships between the input

features and the class labels and makes predictions on new data points based on these learned patterns.

In contrast, multiclass classification, also called multinomial or multivariate classification, entails the classification of data points into more than two categories. The target variable has more than two categories, and the model assigns data points to one of these categories [7]. We can observe in Figure 5 two plots of each type of classification described.

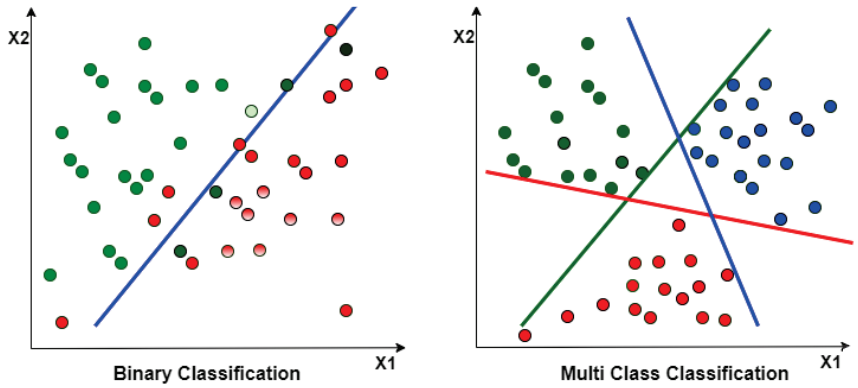

Figure 5. Classification plots. Source: [11].

c. Loss function

This function might go by several names, such as the objective or cost functions. However, it is commonly called the loss function in ML contexts and discussions around minimization problems. It measures the discrepancy between the generated output and the ground truth (or target value) for a particular input example in an ML model. In [12], it is highlighted that the loss function quantifies the error or loss incurred by the model's prediction, and the primary goal in training is to minimize this loss function value.

The loss function used for a given task varies depending on its nature. In the case of regression tasks, the mean squared error (MSE) loss function is generally used, whereas the cross-entropy loss function is typically used for classification tasks [13].

d. Learning and Optimization

Learning and optimization play crucial roles in the field of ML. Learning can improve optimization algorithms by leveraging accumulated knowledge and performance through experience. This type of algorithm usually acts indirectly, assuming that the data they are working with is a good representation of the real-world problem they are trying to solve.

In contrast, optimization algorithms, which focus on finding the best solution based on the mathematical problem formulation, do not usually make assumptions about the data.

The primary objective of an ML algorithm is to minimize the expected generalization error, often referred to as risk. This error quantifies the disparity between a model's performance on the training data and unseen test data, indicating how well the model is expected to perform on new, previously unseen data [12].

A widely used technique is to minimize the expected loss, which refers to the average loss across all possible inputs based on the true data distribution. Nevertheless, there are situations where it may be challenging or impractical to directly handle the true distribution, especially when working with extensive datasets, due to computational constraints or other limitations. In such scenarios, one common approach is to approximate the proper distribution using the empirical distribution derived from the training set. The empirical distribution is a discrete probability distribution that assigns equal probabilities to the observed data points within the training set [7]. The approach of reducing the average risk during the training process is referred to as empirical risk minimization (ERM).

e. Neural Networks

Neural network (NN) is widely used as a type of ML model and has applications in diverse fields, such as image recognition, natural language processing, and pattern recognition. This thesis will mainly focus on a specific kind of NN called a convolutional neural network (CNN). However, before delving into CNNs, it is essential to understand the workings of regular NN.

NN, also called Artificial Neural Network (ANN), is an ML model that draws inspiration from the human brain's structure. They comprise three elements: the input, hidden, and output layers. Figure 6 shows the structure of NN with three layers.

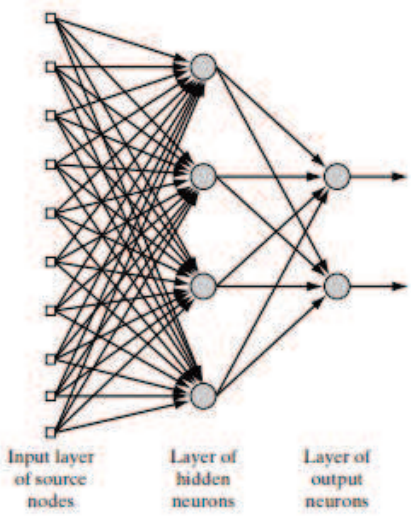

Figure 6. Basic ANN and its layers. Source:[14].

The layers, consisting of interconnected nodes or neurons organized in a network, are designed to process, and learn from data using the backpropagation, which calculates the error gradient, compares the prediction with the ground truth, and adjusts the network weights, employing an optimization technique. This process is performed iteratively until the NN converges to a state where the error is minimized, and the network can make accurate predictions [12].

A basic illustration of a perceptron neuron structure can be seen in Figure 7. It involves inputs labeled as "$x_1, x_2, \ldots, x_m$," which are multiplied by their corresponding weights "$w_1, w_2, \ldots, w_m$," summed with a bias (w_0), and then passed through an activation function (f). These operations result in the generation of the final output.

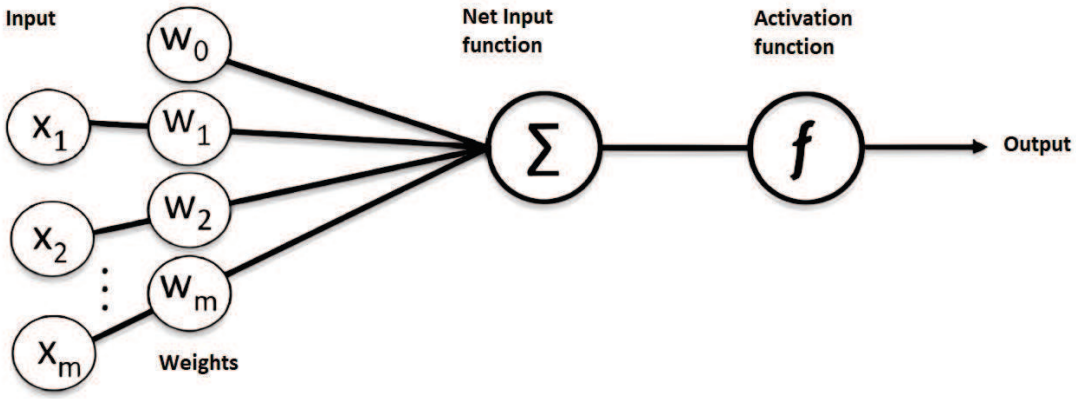

Figure 7. Perceptron model. Adapted from [15].

During training, activation functions play a crucial role in adjusting gradients. Using activation functions in NNs introduces nonlinearity into the system as they determine whether a perceptron should be activated. This nonlinear behavior of activation functions enables deep NNs to learn complex functions [16]. Two famous activation functions are the Rectified Linear Units (ReLU) and the Sigmoid.

The ReLU activation function is defined as

$$f(x) = \max(0, x). \tag{1.1}$$

The sigmoid activation function is defined as

$$f(x) = \frac{1}{(1+\exp(-x))}. \tag{1.2}$$

In Figure 8 we have the graph representations of those activation functions.

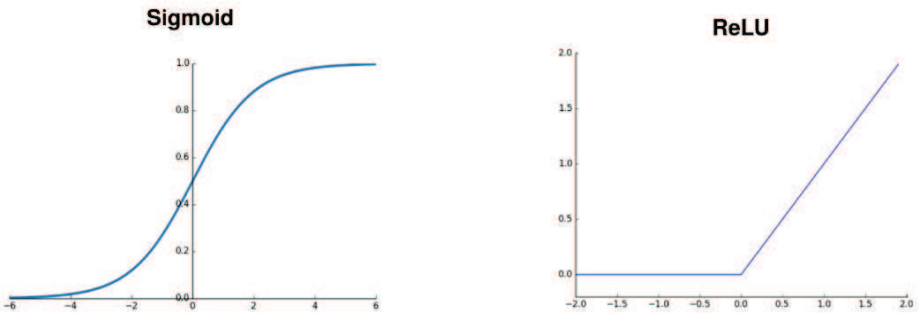

Figure 8. Activation functions. Adapted from [17].

Typically, in multilabel classification networks, the final layer utilizes another activation function called softmax. This function converts the real-valued activations into probabilities for different classes [18]; the softmax function is defined as

$$f(x) = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)}, \tag{1.3}$$

where x_i represents the i-th element of the input vector and $\sum_{j=1}^{K} \exp(x_j)$, denotes the sum of exponential values over all elements in x. Each probability in the result is in the range 0 to 1, and the sum of the probabilities is 1, representing a valid probability distribution over the *K* classes.

f. Computer Vision and Convolutional Neural Networks

Computer vision (CV) is a field that uses mathematical techniques to develop algorithms that can analyze and understand visual information in our environment. CV aims to create machines that can perceive and comprehend the visual world, like human perception [18]. Its focus is creating methods for extracting and interpreting information from images and videos, including identifying and categorizing objects.

In this research, our attention is directed towards images. As defined by Russ in [10], an image is a visual representation of an object or environment captured through a camera or digital means. In CV, an image is a collection of pixels arranged in two dimensions representing specific color or intensity values, typically comprising one or

more color channels. Color channels refer to the different color components that make up an image, such as red, green, and blue (RGB) channels in a color image or grayscale channels in a grayscale image. Each color channel contains information about the intensity of its respective color in the image. These values can be manipulated using various algorithms and methods to obtain insights and make decisions in CV tasks. Figure 9 shows an example of an image of 4x4 pixel RGB image.

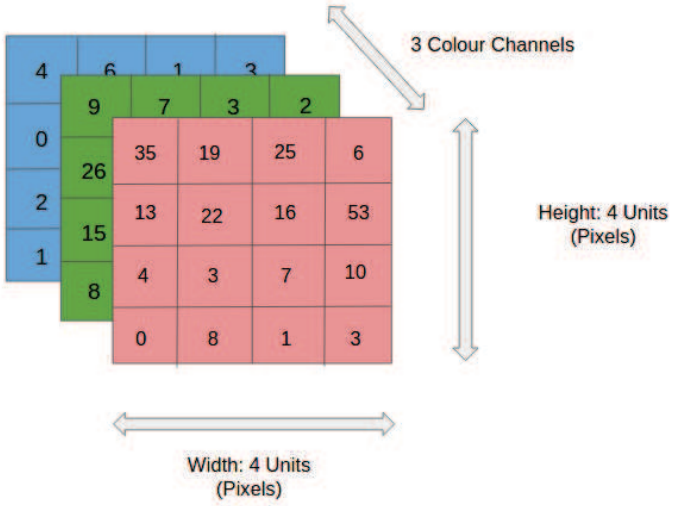

Figure 9. 3D tensor of a RGB image with dimension of 4x4x3. Source: [19].

Choosing the appropriate tool to extract and learn key characteristics from image data using CV is essential. It has been demonstrated that for this task, CNNs are the most effective option [20].

CNNs are a particular type of NN that share similarities with traditional NNs. Like NNs, they use weights, biases, and nonlinear functions to produce outputs. However, they differ using convolution operation instead of matrix multiplication in at least one layer [12].

In a convolutional layer of a CNN, a set of filters or convolutional kernels is used to perform convolution with the input to generate an output feature map. Each filter operates as a numerical grid, with the weights of the filters being learned during the training

of the CNN. The convolution operation involves multiplying the kernel with a specific input section and summing all values to yield a unique result in the output. The kernel navigates over the width and height of the initial feature map, maintaining this process until it cannot proceed further [20]. We can see an example of a 2D convolution operation in Figure 10.

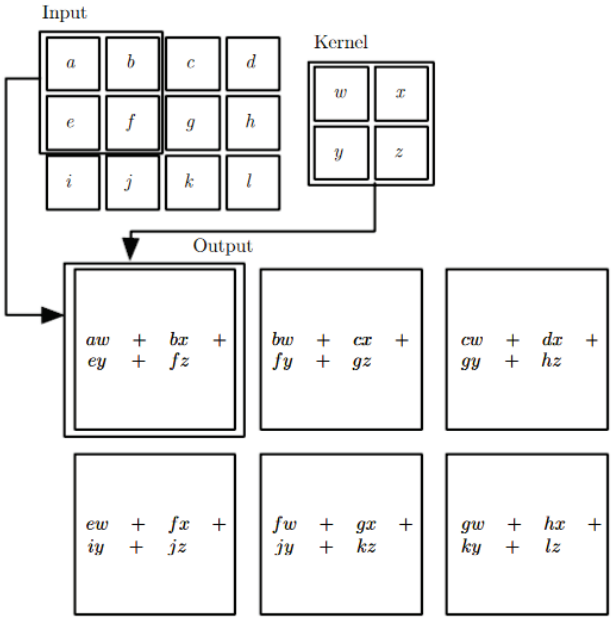

Figure 10. Example of convolutional operation with a 2x2 kernel. Source: [12].

Kernels have two important parameters: size and stride. The size can be any rectangle dimension, while the stride determines the step size used to slide the kernel over the input image during convolution [20]. Essentially, it sets the distance the kernel will shift through the pixels at each iteration. When a stride of 1 is used, the kernel moves one pixel per step, while a stride of 2 moves the kernel to two pixels per step, effectively downsampling the image by a factor of 2. Figure 11 provides another example of the output generated by the convolutional operation.

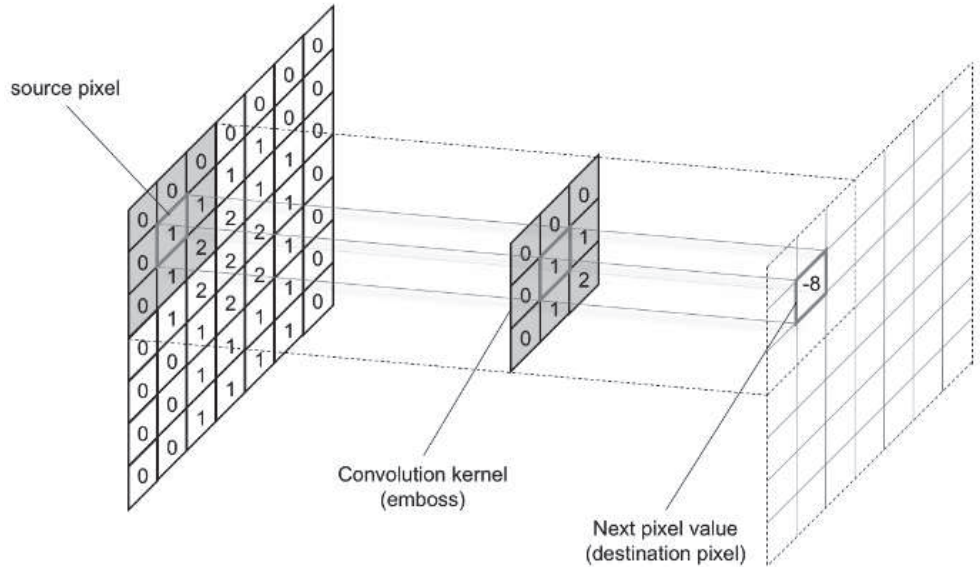

Figure 11. Convolution operation example. Source: [16].

Another essential operation in CNNs is pooling, which is used to downsample the input feature maps using a sampling technique. It is typically placed between convolution layers and uses a sampling technique to reduce the input. The process involves selecting the maximum or average value using a window, depending on the chosen pooling method. This operation applies to all feature channels, and different strides can be used [16]. Figure 12 provides max and average pooling examples, and Figure 13 the mathematical calculations behind the convolutional and max pooling operations.

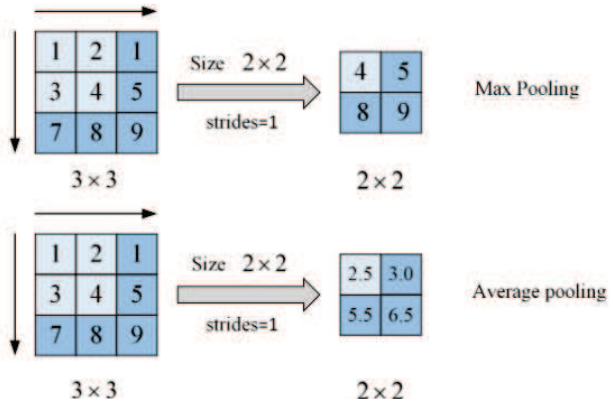

Figure 12. Max Pooling and average pooling. Source: [21].

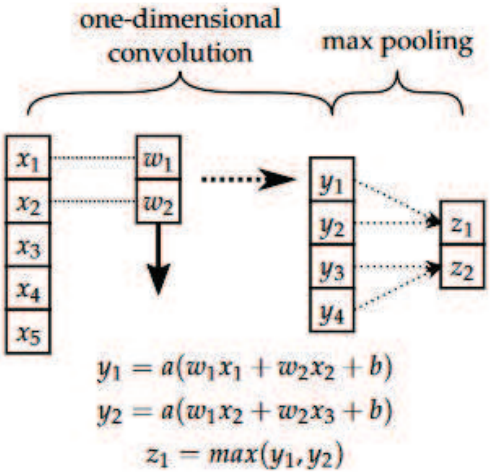

Figure 13. CNN with one convolutional layer and one max pooling layer. Source: [22].

Once the convolutional and pooling operations are done, we arrive at CNN's flattened layer, which takes the multi-dimensional output from the last convolutional layer and flatten it into a one-dimensional array that can be passed to the fully connected layers or dense layers. These layers are accountable for processing the flattened feature maps generated by the previous convolutional and pooling layers and making predictions. These layers function similarly to the hidden layers in a standard NN. Additionally, they are referred to as fully connected since each neuron in the layer is connected to every neuron in the preceding layer [23]. Figure 14 illustrates the passage from the flatten to FC layer.

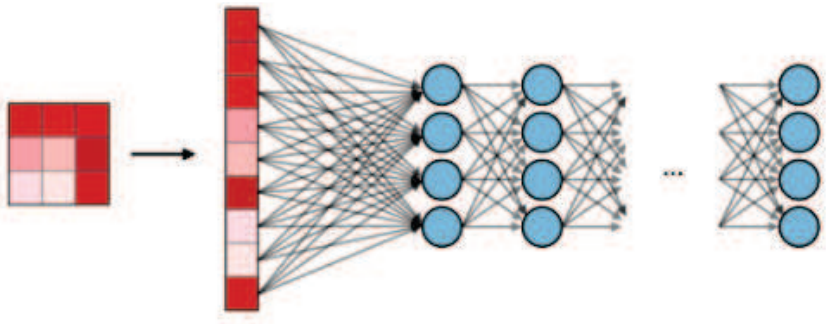

Figure 14. Fully connected layers after the flatten layer. Source: [24].

After processing the fully connected layer, the output is passed through a softmax activation function (1.3). This function converts the output values into probabilities for each class. The input is then classified based on the class with the highest probability.

B. STUDY OBJECTIVE

This study explores an alternative to traditional ERM for NNs to improve the ship detection system, which is utilized to safeguard territorial waters. This ML model will equip the authorities with supplementary capabilities to efficiently surveil the maritime environment.

In the absence of an efficient tool to address this issue, we rely on human resources to execute the task. This requires additional personnel to manually inspect an extensive collection of images in large datasets, resulting in a substantial increase in the time and effort spent on this endeavor.

The central concept of this thesis is to create a resilient ML algorithm using maritime CV to assist with surveillance in an area of concern.

C. MATHEMATICAL MODELING APPROACH

This thesis presents a new approach to image detection in the Naval environment. The goal is to improve the robustness of binary classification algorithms. The primary strategy used is an extension of the work done by Royset et al. in [25], which involves Rockafellian relaxation. This technique combines classical CNNs and stochastic gradient descent (SGD) training to resolve subproblems that fine-tune the nominal probabilities assigned to each image.

The Rockafellian methodology develops robust CNN models, acknowledging the inherent challenges of label corruption in ML datasets susceptible to deliberate attacks and unpredictable disruptions. Chapter III will further explain those methodologies.

D. THESIS ORGANIZATION

The organization of this thesis consists of five distinct chapters: an introduction, a literature review, a math formulations and methodologies section, models results and analysis, and a conclusion.

In Chapter II, a literature review is provided to underpin the efforts of this research, offering a comprehensive summary of pertinent ideas, circumstances, and procedures interconnected with the research aim.

Chapter III elaborates on the mathematical formulation and methodology used in this study. This section delves into the ML algorithms' mathematical underpinnings, emphasizing the models' distinctions.

In Chapter IV, the outcomes of our three unique methodologies are exhibited through model results, highlighting the comparison with a focus on model testing precision, AUC curve, and computational runtime. The concluding section of the chapter briefly analyzes the obtained results.

In conclusion, Chapter V summarizes the findings and briefly suggests potential future research.

II. LITERATURE REVIEW

To determine our objective in this research, we explore past works, the difficulties in achieving objectives, and the latest state-of-art techniques. Our discussion centers on CV and ML, their potential implications, and future directions. Therefore, we aim to contribute to the ongoing conversation about how these technologies can improve human capabilities.

This chapter has a literature review that covers the necessary knowledge to understand this study approach. Our overview begins with the ML approaches to tackle the classification problem, followed by a discussion on NNs. Finally, we explore the latest CV applications in the naval environment.

A. CLASSIFICATION PROBLEM

In ML, the task of classification is of paramount importance. This task requires predicting the target class for unseen data, an essential supervised learning component. In order to accomplish this, models are trained on labeled datasets, with each data point carefully sorted and assigned to a specific category.

The origins of key research in this field trace back to Maron in 1961 [26], who introduced the Naïve Bayes classifiers. This model established practical solutions for classification problems by implementing Bayes' theorem with the assumption of solid independence between features.

In 1967, Cover et al. [27] developed the k-nearest neighbors (k-NN) algorithm as a result of further research. This algorithm assigns new instances to the most common class among its k nearest neighbors through a majority vote.

As the field matured in 1986, Quinlan [28] introduced the Iterative Dichotomiser 3 (ID3) algorithm. This transformative algorithm constructs a decision tree from a given dataset, employing a recursive mechanism that consistently divides based on the attribute offering the maximum information gain. Thus, ID3 introduced decision tree learning to the

arsenal of methodologies in machine learning, significantly enriching the complexity and diversity of algorithms applicable to classification tasks.

In continuing to trace the evolution of classification algorithms, the rise of Support Vector Machines (SVMs) marked a significant milestone. Introduced by Cortes and Vapnik in 1995 [29], SVMs offered a powerful method for binary and multiclass classification problems. By constructing hyperplanes in a multidimensional space, SVMs efficiently distinguished between different classes while maximizing the margin among data points of different categories. Their versatility and robustness under various data conditions cemented their position as a popular choice for classification tasks.

Meanwhile, ensemble methods surfaced, offering improved classification performance by harnessing the collective power of multiple base models. These techniques aggregate predictions from several models to enhance accuracy and robustness, outperforming individual models.

One example of this strategy is the Boosting algorithm introduced by Freund and Schapire in 1997 [30]. Boosting is a strategy that trains a series of weak learners, models that perform marginally superior to arbitrary guessing, like small decision trees. This training is done on altered versions of the data. The training data is modified during each boosting iteration by assigning weights to each sample. These weights are adjusted in each step to highlight the instances that the previous model inaccurately classified. The model predictions are merged using a weighted sum or majority decision to generate the ultimate estimate.

Later, Breiman, in 2001 [31], developed Random Forests. This effective ensemble technique creates multiple decision trees during training, identifying the individual trees' most common class as the output. This method counteracts the overfitting problem of decision trees and provides higher accuracy due to the collective power of the 'forest' of decision trees.

As ML continued to evolve, the deep learning NN models dramatically transformed the field. The foundational work by Hinton et al. in 2006 [32] on deep belief networks ignited the revolution of deep learning, which has proven to be highly effective in handling

complex, high-dimensional data. These models excel in classification tasks because they can learn hierarchical representations, capturing intricate patterns within data that more traditional models might miss.

This progress in NN has significantly enriched the array of advanced algorithms. A thorough exploration of these substantial advances will be provided in the following section.

B. NEURAL NETWORK STRUCTURES

The idea of using a computational version of human neurons was first introduced in 1943 by McCulloch et al. [33], who presented a model inspired by the brain's activity. This pioneering model refers to a system with either one or two inputs that are in binary form and a Boolean function that is activated only when the appropriate inputs and weights are present. Although neural networks draw inspiration from the workings of the human brain, they do not possess the capacity to learn in the same manner as humans.

Later, in 1958, Frank Rosenblatt extending McCulloch's work, publishing the Perceptron Model [34]. Rosenblatt's model can learn the weights of input values and use them to calculate the output. The neuron calculates the weighted sum of values it receives and compares it to a predetermined threshold. When the sum of the values exceeds that threshold, it will output a 1. Conversely, if the total falls below that threshold, it will output a 0. The perceptron is a basic structure, and when several of them are combined, we can form an ANN layer and an entire ANN.

The first ANN utilized to solve a real-world problem came from a model developed by Widrow et al. [35], which used an adaptive filter to eradicate reverberations on telephone lines.

NNs experienced a gap between 1960 and 1980 primarily due to limited computing power, data availability, and funding for research. However, the work done during this period laid the foundation for developing more powerful NN models later.

In 1989, bolstered by the increasing computational power, the backpropagation technique was introduced by Rumelhart et al. [36]. It calculates the loss function gradients

with respect to the NN weights. Therefore, it aims to minimize the loss function and improve the network's predictions. The backpropagation method has significantly impacted the popularity of NNs, enabling the efficient calculation of weight gradients and facilitating the development of advanced models capable of efficiently tackling complex and challenging tasks.

One complex task researchers tried to solve with the NNs' performance improvement is image recognition. This is challenging because it requires the NN to learn and identify patterns and features from complex, high-dimensional visual input. Traditional NN architectures were not designed for image recognition and were limited in their capabilities to learn these patterns from the input effectively; in that scenario, the CNNs appeared.

CNNs were developed as a solution to this problem by incorporating convolutional layers, which apply filters to specific areas of the input to extract features, and spatial pooling layers, which decrease the feature maps dimensions while preserving the vital data. Typically, after these layers, one or more fully connected layers are added to carry out the final classification.

One of the earliest CNNs was the LeNet architecture, which was introduced in 1998 by Yann LeCun et al. [37]. LeNet was designed to recognize handwritten digits, and the U.S. Postal Service utilized it to automate reading zip codes on envelopes. The structure included several convolutional layers and fully connected layers, achieving top-notch performance on a famously referenced dataset called Modified National Institute of Standards and Technology (MNIST) of handwritten digits [37] (Figure 15).

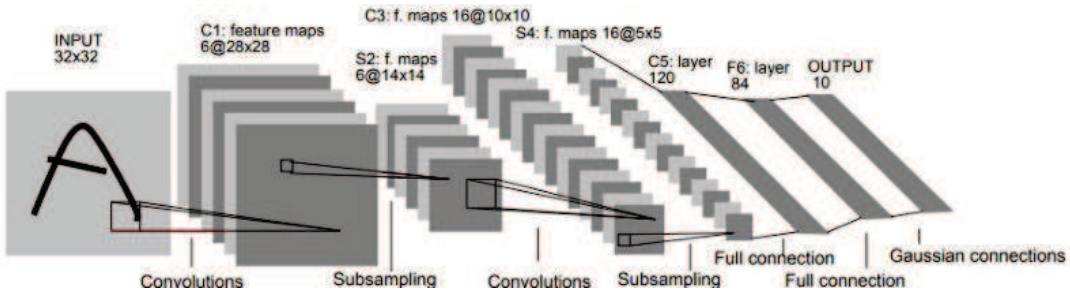

Figure 15. LeNet CNN architecture. Source: [37].

The development of CNNs has been a collaborative effort by researchers from academia and industry. Researchers have experimented with various architectures, including deeper and wider models, residual connections, and attention mechanisms, to improve the performance of CNNs on a variety of tasks. Furthermore, the progress in computing hardware, particularly GPUs, and the availability of massive datasets like ImageNet have facilitated the development of CNNs, making it feasible to train sizable models within a practical timeframe.

The introduction of backpropagation and the popularity of deep NNs led to advancements but also presented new obstacles. A significant issue that emerged is the vanishing gradient problem. In [38], Glorot et al. explain how that problem arises when training deep networks. The gradients that update the network weights can become exceedingly small, halting the network's learning ability. This issue becomes more pronounced with increasing network depth, making earlier layers difficult to train, thereby presenting a considerable challenge for the practical training of deep learning models.

Researchers have continued to refine and innovate upon CNN architectures, incorporating new techniques such as residual connections, attention mechanisms, and transfer learning. We can point out good examples of CNN that gain recognition after the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). In the 2012 and 2014 editions, two important CNN architectures were the winners for object classification, respectively: AlexNet [39] and GoogLeNet (Inception v1) [40].

Krizhevsky et al. made a remarkable discovery during the creation of AlexNet. They found that adding more layers to the network can boost performance if overfitting is prevented. Similarly, Szegedy et al., in GoogLeNet development, demonstrated that increasing the number of channels in each layer can also improve performance, resulting in the creation of more advanced architectures. These architectures have significantly advanced deep learning, particularly in image recognition. Figure 16 illustrates the AlexNet architecture.

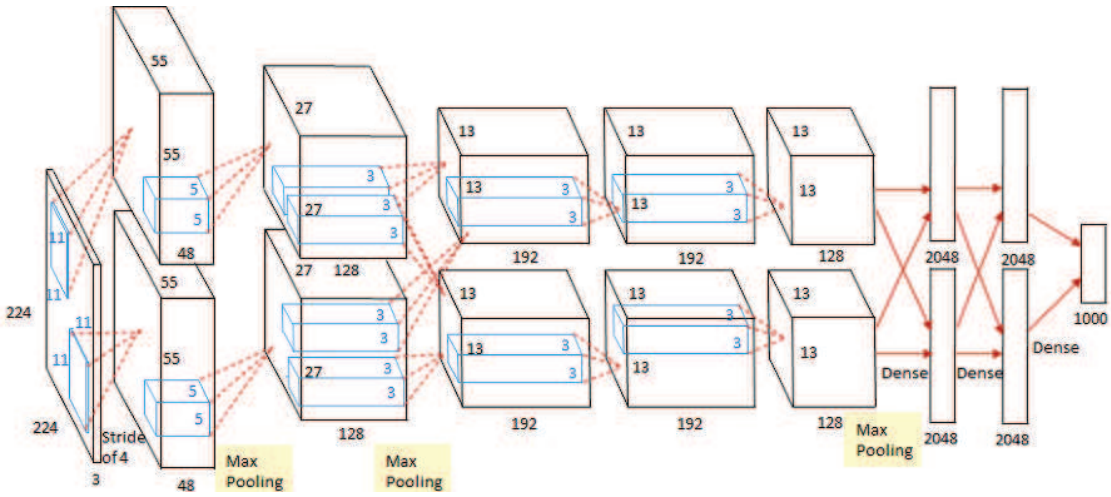

Figure 16. AlexNet CNN architecture. Adapted from [39].

A notable similarity between the two architectures, AlexNet and GoogLeNet, is their choice of activation functions – the ReLU (1.1). This function was introduced by Krizhevsky et al. [31]. and has demonstrated faster training of deep NNs than traditional sigmoid functions. Using ReLU, the deep learning community has mitigated the vanishing gradient problem, a critical challenge when working with deep architectures.

Meanwhile, the major innovation differentiating GoogLeNet from AlexNet is the inception module, a set of convolutional layers capturing features at multiple scales using different filter sizes. The inception modules also incorporate 1x1 convolutions, reducing the number of channels and making the network more computationally efficient. Figure 17 shows the structure of the inception module.

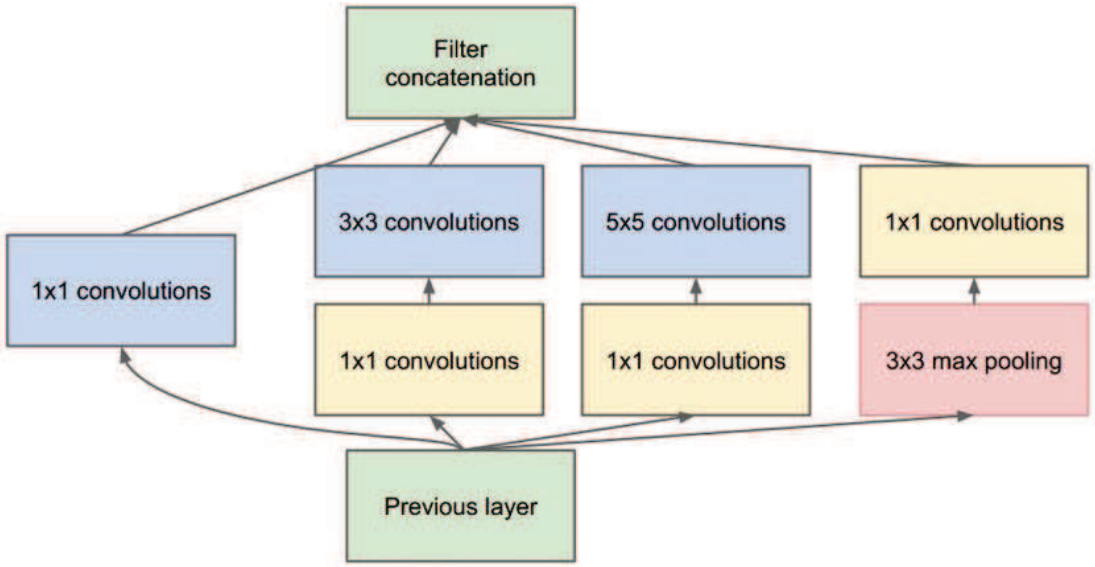

Figure 17. Inception Module with dimension reductions. Source: [40].

Another notable CNN architecture that, while not winning the ILSVRC, gained significant recognition for its outstanding performance in the 2014 edition is the VGG (Visual Geometry Group). This deep CNN architecture was designed at the University of Oxford by Simonyan et al. [41]. Although it took second place in the classification task, losing to GoogLeNet (Inception v1), the performance of VGG was remarkably impressive. Its straightforward and efficient architecture has significantly impacted the deep learning community. VGG comes in two types: VGG-16 and VGG-19, which refer to the number of layers in the network. Figure 18 and Figure 19 illustrate the structure of VGG-16 and VGG-19, respectively.

Figure 18. VGG-16 structure. Adapted from [42].

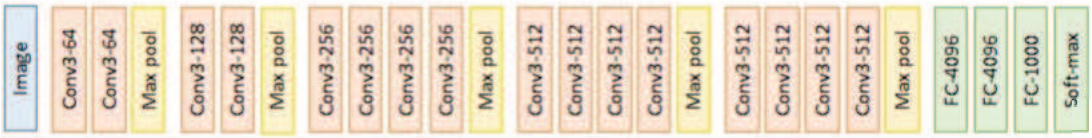

Figure 19. VGG-19 structure. Adapted from [42].

In 2015, The Residual Network (ResNet) was created by He et al. [43] using a deep learning architecture. ResNet's central innovation is the implementation of residual learning using shortcut connections, also known as skip connections. By allowing the gradient to bypass layers during backpropagation, these connections make it possible to train NNs with a previously infeasible number of layers. This innovative approach quickly gained traction due to its remarkable capability to construct more complex NNs. With its ability to learn complex features through its extensive depth and the introduction of residual blocks, ResNet significantly improved its performance and won first place on the ILSVRC 2015 classification task. Figure 20 shows the ResNet shortcuts.

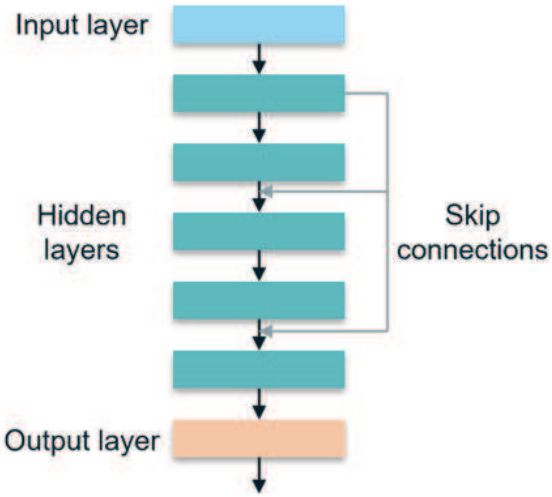

Figure 20. ResNet skip connections structure. Adapted from [44].

A key similarity between VGG and ResNet lies in their depth. Both architectures significantly increased the depth of the models compared to their predecessors, with VGG

going up to 19 layers and ResNet extending this concept even further, with architectures having up to 152 layers.

However, there are several fundamental differences between VGG and ResNet, mainly related to their approach to resolving the vanishing gradient problem associated with deeper networks [45]. VGG, while deeper than its predecessors, suffers from this issue due to its uniform architecture. On the other hand, ResNet introduced the concept of residual learning through shortcut connections, or skip connections, that enable the gradient to be directly backpropagated to previous layers [46].

Another difference is the complexity of the models. VGG, despite its depth, maintains a remarkably straightforward and uniform architecture, consisting solely of 3x3 convolutions and 2x2 pooling throughout the network. In contrast, ResNet's design is more complex due to the incorporation of its residual blocks.

Overall, researchers have reached different kinds of CNN architecture through a combination of theoretical insights, empirical experimentation, and a willingness to try new approaches and techniques.

C. MARITIME COMPUTER VISION

CV aims to replicate and enhance human abilities in interpreting visual data. After reviewing the advancements in NNs to extract meaningful information from visual data in the previous section, researchers have made significant contributions to CV.

Before adding CV in the maritime domain, traditional methodologies served as the primary tools for navigating the complexities of this dynamic environment. They were indispensable for monitoring maritime activities, averting collisions, and facilitating efficient navigation. Born out of necessity and continually refined, these techniques have become the cornerstone of maritime navigation and surveillance, employing a comprehensive array of sensors and technologies – including Vessel Traffic Services (VTS), coastal radar systems, and satellite-based Automatic Identification Systems (AIS) [47].

In the research made by Yassir et al. [48], the limitations of traditional methodologies, such as AIS and marine radar systems, are underlined. However, the study also emphasizes that incorporating camera surveillance systems can effectively bridge these gaps. The combination of these surveillance mechanisms enhances threat prediction in a limited coverage area but also aids in preventing other illegal activities, such as drug trafficking and illegal immigration.

The integration of CV and maritime operations has led to new possibilities. This synthesis creates maritime CV, a rapidly advancing field that leverages these sophisticated NN architectures to address unique maritime challenges. Several studies [49], [50], [51], [52] show that CNN-based CV techniques have greatly improved imagery interpretation in the maritime domain, bringing significant progress.

The study in [51] generated a dataset encompassing ships, artificial platforms, and harbors. This research brought forth a CNN model composed of four convolutional layers, four pooling layers, and a fully connected layer. Using data augmentation, which involves creating multiple views of images across different channels, this model achieved a 94% success rate in classifying five types of marine targets. This achievement significantly outperformed other CNN models and conventional ML approaches [51].

The work in [52] conducted by Gallego et al. used a dataset called MASATI (Maritime Satellite Imagery) to classify ships. They developed a CNN model consisting of two convolutional, two pooling, and one fully connected layer as the baseline. Additionally, they incorporated transfer learning from well-known architectures such as VGG-16/19 and ResNet. The findings showed that pre-trained models containing weights learned from larger datasets performed significantly better than the baseline approach, achieving a maximum accuracy of 99.76% compared to the 68.31% of baseline [52].

Fang et al. [53] presented a method to identify small targets in maritime environments using CNNs combined with infrared imaging. The model includes a regularization term based on total variation and a reweighted sparse constraint to improve accuracy and eliminate non-target points. They improved the robustness and accuracy of

detection, demonstrating the potential of these methods in infrared imaging in real-world situations, particularly in maritime surveillance [53].

Recent technological advancements have significantly enhanced the capabilities of image-capture platforms and CV applications. The growing adoption of drones, combined with rapid developments in sensor technology, has forged innovative paths for utilizing CV. Unmanned Aerial Vehicles (UAVs) have emerged as invaluable tools in maritime surveillance, a development made possible partly due to substantial increases in computational speed and processing power [54]. Modern processors, often optimized for ML tasks, have facilitated the execution of intricate calculations required for advanced deep-learning models. This boost in computational power has proven crucial in deploying sophisticated real-time object detection systems, enhancing their efficiency and versatility, and further propelling the value and effectiveness of UAVs in surveillance applications. Many recent research studies, including references [55], [56], [57], have shown evidence of these developments.

Scholars like Lo et al. [56] have explored UAV-based systems for dynamic object tracking using a famous architecture inspired by GoogleNet, called YOLO (You Only Look Once) [58]. This network uses deep learning technology and is ideal for object detection in surveillance systems. Traditional detection methods require the classifier to be applied numerous times on various scales and locations within an image. However, YOLO receives that name because of its characteristic of performing object detection and classification in just one forward propagation pass through the network, making it exceptionally fast and efficient for real-time applications.

Similarly, Lygouras et al. [57] proposed an autonomous human detection system using a UAV for SAR operations, notably leveraging the YOLO structure. The use of YOLO in their system underscores the vast potential of unmanned aerial systems and the effectiveness of the CNN architecture in critical, real-time tasks.

All those papers show that Maritime operations are witnessing a significant swell in adopting CV techniques. Integrating cutting-edge technologies promises to redefine Maritime CV techniques, pushing the limits of what is possible and finding new solutions

for unique maritime challenges. Emphasizing this growing importance, the first edition of the Maritime Computer Vision (MaCVi) workshop took place in 2023 [59].

D. LABEL NOISE

In their efforts to craft high-performance deep learning models, specialists utilize numerous techniques discussed in the previous section, such as preprocessing data, changing the color spectrum, utilizing data augmentation, and transferring knowledge from established network architectures. However, this section shifts its focus towards confronting a common issue in real-world data known as label noise. This problem often presents obstacles in the learning process and significantly affects the outputs of the models.

Label noise occurs when the labels or outcomes of data points are incorrect because they may have been distorted or altered from their original, accurate labels [60]. Notably, in real-world datasets, the prevalence of such corrupted labels is reported to vary, with estimates ranging between 8.0% and 38.5% [61].

Despite the outstanding performance of deep NNs over traditional methods on large, accurately labeled datasets, real-world datasets often bear label errors. For instance, random noise errors can unintentionally emerge during data collection or labeling from human error or limitations in automated labeling systems; in [62], Liu et al. analyze the classification problem in a random data corruption environment. In contrast, another known threat is the data poisoning attack, as illustrated in [63], also called an adversarial attack, characterized by deliberate data alterations by the attacker aiming to weaken the model's performance by misleading and confusing its training.

Striving for increased robustness and precision in deep learning models, multiple researchers have endeavored to lessen the effects of label noise [64], [65], [66], [67], and. Their main goal was to augment the innate resilience of these models to withstand label noise and to boost their capability to extrapolate insights from training data. They aspired to accomplish these objectives without substantially modifying the current network structures or optimization processes.

In the context of label noise, M. Ren et al. [64] introduced a novel using a meta-learning algorithm that assigned weights to training examples based on their gradient orientations. This technique involved a gradient descent step on the batch weights to minimize the loss on a uncontaminated, unbiased validation set. The suggested approach achieved impressive results, especially in cases of disproportionate class distribution and distorted labeling, where only a limited amount of uncontaminated validation data was accessible [64].

Tackling label noise, S. Thulasidasan et al. [65] proposed a deep abstaining classifier (DAC) capable of avoiding confusing samples during training. This approach showed promise, especially in structured or systematic label noise, where features associated with unreliable labels could still contribute to representation learning.

Chen et al. in [66] introduced a hierarchical structure that embeds labels hierarchically into the model's training. The primary concept involves enhancing the conventional NN by incorporating a mapping function for label hierarchy and a weighted loss function to perform a fine adjustment to deal with different noise ratios [66]. They argued that their method increased the robustness of models without necessitating considerable modifications to the network architecture or optimization procedure. The approach showed superior performance over conventional deep NNs in label noise.

The work developed by H. Narasimhan et al. in [67] extends the work done by Thulasidasan et al. in [65] and other authors using the learning-to-reject (L2R) [68], [69], an approach combining with out-of-distribution (OOD) [70] detection. L2R aims to identify complex samples for abstention, whereas OOD detects outlier samples that do not belong to the training distribution. Impressively, they demonstrated that these issues, usually tackled individually, can be solved collectively, indicating a way to abstain from complex and outlier samples while controlling the total abstention of training data.

All these contributions have significantly advanced our understanding and management of various challenges inherent in deep learning, mainly related to label noise and the need for model robustness.

The study by Royset et al. [25] introduced Rockafellian relaxation to address the problem of inaccuracies in optimization models due to suspect assumptions and compromised data labels. They proposed an "optimistic" framework, facilitating optimization over the original decision space and a range of model perturbations, demonstrating their applicability in CV affected by label noise.

Rockafellian relaxation helps ensure the validity and stability of optimization models by analyzing the sensitivity of assumptions and parameters [71]. This approach allows for flexibility in considering potential changes and aids in understanding the robustness of the solution, guiding decision-making processes during optimization [71]. Chapter III will provide a thorough mathematical explanation of the Rockafellian relaxation algorithm used in this study.

In this thesis, we seek to extend and explore the work initiated by Royset et al. [25] and apply their approach to enhance the reliability of CV in maritime settings. We aim to embrace their positive approach and investigate the possibilities for further progress. We will specifically address the fundamental issues of label inaccuracies and the need for model stability.

Furthermore, we will investigate the impacts of the Rockafellian approach on the model's resilience to label noise. Through this research, we aspire to push the boundaries of our understanding of these techniques, ultimately contributing to the broader field of robust and noise-resilient deep learning models.

III. ROCKAFELLIAN RISK MINIMIZATION

This chapter develops Rockafellian relaxation in the context of ML, leading to Rockafellian Risk Minimization (RRM) as a means for training in settings with corrupt data. We contrast the approach with ERM and discuss computational aspects.

A. FORMULATION

Give a labeled dataset $\{x_j, y_j, j = 1, \dots, n\}$ of images, where x_j specifies the attributes for each pixel of the *j*th image and y_j is the corresponding label, we seek to determine a parameter vector *w* in a NN such that its prediction for each image matches the label. In the following, we assume that there are only two labels -1 and 1. Let $g_w(x_j)$ be the prediction by the NN for the *j*th image. It is interpreted as the probability that the *j*th image has label 1. We determine the best parameter vector *w* by minimizing binary cross entropy loss given by

$$f_j(w) = \begin{cases} -\ln g_w(x_j) \text{ if } y_j = 1 \\ -\ln\left(1 - g_w(x_j)\right) \text{ if } y_j = -1. \end{cases}$$

Classical ERM then amounts to solving the optimization problem

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \sum_{j=1}^{n} p_j f_j(w),$$

where $p_j = 1/n$ typically but the data points can also be weighted differently.

In many applications, data corruption can occur due to various reasons such as data collection and labelling entry errors, or even adversarial attacks intended to poison the training data and disrupt the system. As pointed out by Royset et al. [25], and elaborated in section D of Chapter II, a NN that has been trained using ERM may not perform well if the training data is corrupted. This observation provides a strong motivation for RRM, an adaptive method to identify and remove corrupted data points, hence improving the overall performance and reliability of the model.

RRM leverages auxiliary decision variables $u_1, \dots, u_n$, to adjust the probability associated with each data point. This leads to the formulation

$$\underset{w\in\mathbb{R}^d,u\in U}{\text{minimize}} \sum_{j=1}^{n} \left((p_j + u_j)f_j(w) + \theta|u_j|\right),$$

where θ represents a penalty parameter, and u is a perturbation vector with n dimensions that alters the nominal probability vector p, and the set

$$U = \left\{ u \in \mathbb{R}^n \middle| u_j \geq -p_j, \sum_{j=1}^{n} u_j = 0 \right\}.$$

We adjust the probability associated with each data point by adding u_j to p_j. To identify corrupted data points, we optimize u based on their calculated loss with the goal of reducing the probability that affects $f_j(w)$ to zero. This effectively eliminates the data point from being considered in the training process.

B. TRAINING ALGORITHMS

The RRM method consists of the usage of an Alternating Direction Heuristic (ADH), which alternates between optimizing different sets of variables while keeping the others fixed, hence the name "alternating direction." This strategy begins by optimizing *w*, which adjusts the neural network weights. Subsequently, we optimize *u* to modify each data point's probability, isolating and discarding potential data corruption. The model is refined progressively in each cycle until it reaches the number of iterations defined by the user. The details of the algorithm are given next.

Alternating Direction Heuristic (ADH)

Data. Number of epochs κ, number of iterations τ, initial weights w^0.

Step 0. Set iteration counter $i = 1, p^1 = p, u^1 = 0$.

Step 1. Starting from w^{i-1}, apply SGD-type algorithm for κ epochs to the problem

$$\underset{w\in\mathbb{R}^d}{\text{minimize}} \sum_{j=1}^{n} p_j^i f_j(w).$$

Let w^i be resulting solution.

Step 2. Select u^{i+1} using some subroutine, which in turn is based on w^i and u^i.

Step 3. If $i < \tau$, set $p^{i+1} = p + u^{i+1}$, replace i by $i + 1$, and back to Step 1.
Else, stop.

We are considering two approaches for **Step 2** of ADH: a Linear Programming Based approach (ADH-LP) and a Subgradient Based approach (ADH-SUB).

In ADH-LP, we use linear programming to determine the values of *u* by optimizing a linear objective function and considering linear equality and inequality constraints. The parameter $\mu \in (0,1]$ is the stepsize of that approach to update the *u* values in each iteration *i*. In the following, we have the ADH-LP steps:

ADH-LP

Data. Set stepsize μ.

LP-Step 2a. Solve the linear optimization problem

$$\underset{u\in U\,,v\in\,\mathbb{R}^n}{\text{minimize}} \sum_{j=1}^{n} \left(u_j f_j(w^i) + \theta v_j\right) s.t. u_j \le v_j, -u_j \le v_j, j = 1, \dots, n.$$

Let (u^*, v^*) be a minimizer.

LP-Step 2b. Set $u^{i+1} = \mu u^* + (1 - \mu)u^i$.
Go to Step 3.

We note that components of u^i sum to zero and are always greater than $-p_j$.

In an alternate approach to the second stage of ADH minimization, we utilize the subgradient method [71, Section 2.I], augmented with projection onto the probability simplex [72]. Rather than relying on gradients, this method uses subgradients at points where the function may not be differentiable. The method produces practical solutions by combining subgradients with projection onto the probability simplex, even when the objective function is not smooth. The parameter $\lambda, (\lambda > 0)$, will be the stepsize of that approach. In the following we have the ADH-SUB steps:

ADH-SUB

Data. Set stepsize $\lambda > 0$ and number of sub-iterations σ.

SUB-Step 2a. Set $u^{i+1} = q^{\sigma+1} - p$, where $q^{\sigma+1}$ is computed as follows.

SUB-Step 2b. Set $q^1 = p + u^i$ and iteration counter $k = 1$.

SUB-Step 2c. Compute the vector a^k be setting

$$a_j^k = f_j(w^i) + \begin{cases} \theta \text{ if } q_j^k > p_j \\ 0 \text{ if } q_j^k = p_j \\ -\theta \text{ if } q_j^k < p_j \end{cases}.$$

SUB-Step 2d. Compute $\hat{q}^{k+1} = q^k - \lambda a^k$.

SUB-Step 2e. Project $\hat{q}^{k+1}$ onto the probability simplex to obtain q^{k+1} by carrying out the following substeps:

Sort $\hat{q}^{k+1}$ into a vector r with $r_1 \geq r_2 \geq \ldots \geq r_n$.

Find $j^* = \max\{j \mid r_j + (1 - \sum_{i=1}^{j} r_i)/j > 0\}$

Define $\delta = (1 - \sum_{j=1}^{j^*} r_j)/j^*$.

Set $q_j^{k+1} = \max\{\hat{q}_j^{k+1} + \delta, 0\}$ for each $j = 1, \ldots, n$.

SUB-Step 2f. If $k < \sigma$, replace k by $k + 1$, and go to Step 1. Else, stop.
Go to Step 3.

To solve the classic ERM problem, we consider the empirical distribution, a discrete probability distribution assigning equal probabilities to the observed data points within the training set. RRM can improve the overall performance during training and achieve better results by adjusting these probabilities.

In our current methodology, we have extended the set of hyperparameters to include new ones in ADH-LP (θ and μ) and ADH-SUB (θ, λ, and σ) alongside the regular hyperparameters (learning rate, number of epochs, batch size). By carefully fine-tuning these hyperparameters, our objective is to enhance the model's accuracy and robustness. This iterative process empowers our model to effectively address data corruption, such as label errors and adversarial attacks, enabling it to perform reliably in real-world scenarios.

IV. EXPERIMENTS

In this chapter, we delve into the model's results. However, we first discuss the datasets, data preprocessing, optimizers, and network structures. Then, we present the results our ERM and RRM model achieve without contamination and with different degrees of label contamination in training data. Next, we analyze the behavior of our perturbation vector values in certain cases. Finally, with all results presented, we can assess the adaptability and robustness of RRM compared to ERM across different contamination levels.

We are utilizing two separate datasets: the Airbus Ship Detection (AIRBUS) [73] and Maritime Satellite Imagery (MASATI) [52] datasets. Each of these datasets is comprehensively explored in their respective subsections, with additional details provided to understand their unique attributes and the specific challenges they pose. In both datasets, we consider an environment without label contamination and another where we gradually increase levels of label contamination in the training set, starting with 10%, 20%, 30%, and 40% by randomly swapping training example labels. We have tested higher contamination levels of 50% and 60%, but the CNNs used have difficulty learning patterns at those levels.

In this research, we utilize Tensorflow version 2.11.0 to conduct our experiments. During the first phase of the ADH process, concerning w-optimization, we try two distinct NN optimizers: Adam [74] and SGD [71, Section 3.G]. The Adam optimizer is configured with a learning rate of $1.0 \cdot 10^{-3}$, while the SGD optimizer has two distinct learning rates being employed based on the dataset used: a learning rate of 0.1 for the AIRBUS dataset, and a learning rate of 0.02 for the MASATI dataset. The momentum is consistently set at 0.9 for SGD in both datasets. We run our experiments on a singular GPU, the Nvidia Tesla V100, with 32GB of memory capacity.

As we progress into the second phase of the ADH process, we can employ either the ADH-LP or the ADH-SUB algorithm for u-optimization. The ADH-LP method incorporates Pyomo version 6.4.0, in conjunction with the CPLEX solver. The ADH-SUB technique is written in Python without requiring supplementary libraries or solvers.

We investigate numerous network architectures, learning rates, and other hyper-parameter combinations. Our work, however, is not centered on enhancing performance through network architecture modifications. Rather, we aim to show that for any given choice of network architecture, an RRM approach can confer benefits over an ERM approach. To this end, we settle on two adequate network configurations to conduct our comparison.

Both CNNs receive input images resized to 128 x 128 x 3 at their input layers. In their hidden layers, the activation function used is ReLU, while the output layer consists of two output units with softmax activation applied for binary classification. The total number of trainable weights is significantly different for the two networks. In the case of the CNN designed for the Adam optimizer, there are 2,228,002 trainable parameters, while the CNN designed for the SGD optimizer has 8,409,026. Tables 1 and 2 describe the CNN structure used for Adam and SGD optimizers. The loss function we used for training both networks is binary cross-entropy.

Once we establish a satisfactory baseline model using ERM, we execute the RRM algorithms ADH-LP and ADH-SUB using identical network configurations.

Table 1. Description of the CNN used for Adam

#	Layer	Filters	Kernel	Output Size	# Parameters
1	Convolution	16	3x3	128 x 128 x 16	448
	Max-Pooling		2x2	64 x 64 x 16	
2	Convolution	32	3x3	64 x 64 x 32	4,640
	Max-Pooling		2x2	32 x 32 x 32	
3	Convolution	64	3x3	32 x 32 x 64	18,496
	Max-Pooling		2x2	16 x 16 x 64	
4	Convolution	128	3x3	16 x 16 x 128	73,856
	Max-Pooling		2x2	8 x 8 x 128	
5	Fully-connected	256		1 x 256	2,097,408
6	Fully-connected	128		1 x 128	32,896
7	Softmax (Output)	2		1 x 2	258

Table 2. Description of the CNN used for SGD

#	Layer	Filters	Kernel Size	Output Size	# Parameters
1	Convolution	32	3x3	128 x 128 x 32	896
2	Batch Normalization Activation				128
3	Max-Pooling		2x2	64 x 64 x 32	-
4	Convolution Activation	64	3x3	64 x 64 x 64	18,496
	Max-Pooling		2x2	32 x 32 x 64	
5	Fully-connected	128		1 x 128	8,388,736
6	Batch Normalization Activation				512
7	Softmax (Output)	2		1 x 2	258

For ERM, we set the number of epochs $\kappa = 500$. For RRM algorithms, we establish $\kappa = 10$ and iterations $\tau = 50$, which guarantees that all heuristic runs will include 500 epochs. The ADH-LP uses the default stepsize parameter $\mu = 0.5$. At the same time, the ADH-SUB algorithm uses a stepsize parameter $\lambda = 0.3$. and a default number of sub-iterations parameter $\sigma = 10$. Concerning the penalty parameter θ in both RRM, we show 5 different levels of θ to evaluate their performance in the u-optimization across the values of $\theta \in \{0.15, 0.20, 0.25, 0.30, 0.35\}$. Table 3 summarizes the parameters described.

Table 3. Parameters for ERM and RRM

	Parameters				
Algorithm	Number of epochs (κ)	Number of iterations (τ)	Stepsize (μ or λ)	Sub-iterations (σ)	Penalty (θ)
ERM	500	1	-	-	-
RRM(ADH-LP)	10	50	0.5	-	0.15, 0.20, 0.25, 0.30, 0.35
RRM(ADH-SUB)	10	50	0.3	10	

In order to compare results, we ensure that the random seed is the same at two specific times for all runs: when we divide the data into training and testing sets and when we execute closed-set contamination through label swapping.

Table 4 summarizes the algorithm runtimes over 500 epochs, separating the time used for the *w*-optimization and for *u*-optimization processes in both datasets.

Table 4. Computational time of algorithms

	Dataset							
	MASATI				**AIRBUS**			
	CNN optimizer				**CNN optimizer**			
	Adam		**SGD**		**Adam**		**SGD**	
Optimization phase	*w*	*u*	*w*	*u*	*w*	*u*	*w*	*u*
Algorithm	Total Runtime in seconds over optimization phases							
ERM	500	-	1,400	-	22,500	-	68,500	-
RRM (ADH-LP)	500	600	1,400	600	22,500	1,000	68,500	1,000
RRM (ADH-SUB)		400		400		600		600

There are noticeable differences between the MASATI and AIRBUS datasets and the Adam and SGD optimizers. When using the Adam optimizer on the MASATI dataset, the ADH-LP method takes 1,100 seconds, while the ADH-SUB method takes 800 seconds. However, if we switch to the SGD optimizer, ADH-LP takes 2,000 seconds, while ADH-SUB only requires 1,800 seconds. Looking at the AIRBUS dataset, with the Adam optimizer, RRM with ADH-LP takes 23,500 seconds to complete, while ADH-SUB is slightly faster at 23,100 seconds. Under the SGD optimizer, ADH-LP requires 69,500 seconds, while ADH-SUB needs 69,100 seconds. It is worth noting that the ADH-SUB method is considerably faster than ADH-LP in the u-optimization process, especially in the larger AIRBUS dataset.

A. MASATI DATASET

This dataset provides maritime scenes of optical aerial images from the visible spectrum. The MASATI dataset contains color images in dynamic marine environments, and it is used to evaluate ship detection methods. The images may have one or more targets and vary in weather and lighting. All files are RGB images with a resolution of 512 × 512 pixels, stored in PNG format.

The dataset comprises 7,389 satellite images labeled according to the following seven classes: coast, land, ship, sea, coast-ship, multi, and detail. However, as we performed a binary classification problem, we used just two classes: “ship,” containing 1,027 images, and “sea,” containing 1,022 images, yielding a set of 2,049 images.

While this total of 2,049 images constitutes a relatively small dataset, despite its size, it presents a unique opportunity to assess whether the algorithms are robust and versatile enough to deliver accurate results even when the available data is limited. Figure 21 illustrates some images of MASATI.

Figure 21. MASATI dataset images. Source: [52].

1. Accuracy Results with MASATI

In this section, we show our results in the MASATI dataset, explicitly focusing on the accuracy score, which indicates how effectively our models can classify images within the test set. Along with presenting the results, we also provide a thorough analysis to interpret the observed performance.

In MASATI, the train-test split is done using a 90/10 proportion, which results in 1,844 images in the training set and 205 in the test set.

The final test accuracy results for ERM and RRM with ADH-SUB and ADH-LP using Adam optimizer are shown in Tables 5 and 6, respectively, at various levels of corruption.

Table 5. Final test accuracy in MASATI for ERM and RRM (ADH-SUB / Adam)

	Corrupted training data percentage				
Method	40%	30%	20%	10%	0%
ERM	0.624	0.653	0.785	0.858	0.941
RRM ($\lambda = 0.3$)					
$\theta = 0.15$	0.585	0.721	0.746	0.843	0.956
$\theta = 0.20$	0.560	0.678	0.795	0.858	0.951
$\theta = 0.25$	0.575	0.643	0.829	0.863	0.941
$\theta = 0.30$	0.629	0.600	0.765	0.834	0.951
$\theta = 0.35$	0.663	0.712	0.770	0.848	0.912

The values highlighted in gray represent the cases where RRM outperform or match ERM.

In both ERM and RRM approaches, an increased proportion of corrupted data in the training set negatively affects the accuracy. However, this degradation can be mitigated by employing RRM and selecting an appropriate value for the penalty parameter θ.

While the RRM method underperforms ERM at higher corruption levels (40% and 30%) for most values of θ, it consistently outperforms or matches ERM as the corruption percentage decreases (20%, 10%, and 0%), given an appropriate θ. Notably, at 0% corrupted data, RRM surpasses ERM for every θ except for $\theta = 0.35$.

RRM also shows a more nuanced response to the data's corruption percentage. Choosing θ = 0.20 or 0.25 in the RRM method presents an attractive balance between maintaining performance in scenarios of high data corruption while also taking advantage of cases with lower corruption levels.

Table 6. Final test accuracy in MASATI for ERM and RRM (ADH-LP / Adam)

	Corrupted training data percentage				
Method	40%	30%	20%	10%	0%
ERM	0.624	0.653	0.785	0.858	0.941
RRM (μ = 0.5)					
θ = 0.15	0.624	0.668	0.800	0.863	0.951
θ = 0.20	0.609	0.726	0.848	0.878	0.931
θ = 0.25	0.668	0.682	0.814	0.863	0.941
θ = 0.30	0.604	0.702	0.804	0.843	0.926
θ = 0.35	0.614	0.692	0.756	0.834	0.921

The values highlighted in gray represent the cases where RRM outperform or match ERM.

The RRM method exhibits performance at par or even superior to ERM at various levels of data corruption, especially for specific penalty values. Remarkably, when the values of 0.15, 0.20, and 0.25 are chosen for θ, they yield the best overall performance. For example, θ = 0.15 performs exceptionally well at lower corruption levels, θ = 0.20 outperforms all other θ values at a 20% corruption level, and θ = 0.25 performs best at a 30% corruption level.

The performance difference is relatively minimal in scenarios where the RRM method does not surpass ERM. This finding implies that even with non-optimal selections of θ, the RRM method can deliver performance nearly equivalent to ERM. It accentuates the versatility of the RRM method as a promising alternative to ERM, especially in environments fraught with corrupted data.

These results suggest that a set of $\theta \in \{0.15, 0.20, 0.25\}$ provides an optimal balance in modulating the penalty for model complexity versus the fit to training data. This

balance contributes to superior generalization of the test data, particularly at intermediate corruption levels.

Now turning to SGD optimizer, Tables 7 and 8 display the final test accuracy results for ERM and RRM in ADH-SUB and ADH-LP at various levels of corruption, respectively.

Table 7. Final test accuracy in MASATI for ERM and RRM (ADH-SUB / SGD)

	Corrupted training data percentage				
Method	40%	30%	20%	10%	0%
ERM	0.556	0.546	0.581	0.663	0.648
RRM ($\lambda = 0.3$)					
$\theta = 0.15$	0.502	0.560	0.575	0.629	0.634
$\theta = 0.20$	0.512	0.526	0.619	0.604	0.639
$\theta = 0.25$	0.541	0.536	0.600	0.629	0.658
$\theta = 0.30$	0.517	0.560	0.614	0.653	0.634
$\theta = 0.35$	0.546	0.521	0.624	0.678	0.639

The values highlighted in gray represent the cases where RRM outperform or match ERM.

The findings indicate the RRM method's best performances, which match or exceed ERM's, particularly at the 20% and 30% corruption levels.

When examining how well different settings handle varying levels of corruption, the configuration with $\theta = 0.35$ stands out as a great option. It achieves the highest accuracy when there is 20% corruption, showing it can easily handle moderately corrupted situations. While there may be slight differences in performance at other corruption levels, $\theta = 0.35$ consistently performs well and can adapt to different data conditions.

Meanwhile, the $\theta = 0.30$ configuration is not to be overlooked. Though it may not consistently outperform other configurations, it provides valuable stability across all corruption levels. It often matches, if not slightly surpasses, the ERM's accuracy, suggesting its capability to resist the adverse effects of data corruption.

In conclusion, $\theta = 0.35$ and $\theta = 0.30$ configurations represent different advantages. While $\theta = 0.35$ exhibits superior performance in specific corruption scenarios, $\theta = 0.30$ offers consistent performance across all levels.

Table 8. Final test accuracy in MASATI for ERM and RRM (ADH-LP / SGD)

	Corrupted training data percentage				
Method	40%	30%	20%	10%	0%
ERM	0.556	0.546	0.581	0.663	0.648
RRM ($\mu = 0.5$)					
$\theta = 0.15$	0.604	0.648	0.648	0.639	0.634
$\theta = 0.20$	0.663	0.692	0.648	0.648	0.609
$\theta = 0.25$	0.639	0.687	0.658	0.629	0.614
$\theta = 0.30$	0.668	0.585	0.600	0.643	0.634
$\theta = 0.35$	0.624	0.556	0.639	0.648	0.653

The values highlighted in gray represent the cases where RRM outperform or match ERM.

From Table 8, the RRM method, when compared with ERM, delivers better or comparable performance at different levels of data corruption.

Examining the selection of the $\theta \in \{0.20, 0.25, 0.30\}$, we observe that these selections yield the highest accuracies across the range of data corruption levels.

At a 40% corruption level, the RRM outperforms the ERM method. For $\theta = 0.20$, the accuracy improvement is a substantial 10.7%. Similarly, for $\theta = 0.30$, the performance is enhanced by 11.2% compared to ERM. When faced with 30%, the accuracy improvement for $\theta = 0.20$ over ERM is an impressive 14.6%, the highest observed in this data set. In the 20% corruption level, the RRM with a $\theta = 0.25$ exhibits a promising 7.7% improvement in accuracy over the ERM.

Interestingly, at lower corruption levels of 10% and 0%, the performance of RRM remains comparable to ERM. This reveals the versatility of the RRM method, maintaining robust performance even as the corruption level diminishes.

The superior performance of RRM at higher corruption levels (20%, 30%, and 40%) across all θ values, combined with its comparable performance at lower corruption levels, indicates its resilience.

2. *U*-optimization Analysis with MASATI

In the following section, we aim to examine the u-vector behavior in some MASATI cases. We show the dynamic evolution of the perturbation vector values across the iterative steps of the RRM algorithm and how it influences performance and resilience, thereby illustrating its crucial role within this methodology.

The u-vector helps improve the model's performance by adjusting the probability of each data point. It assigns lower values to mislabeled examples excluding them from the NN training process. The penalty parameter θ regulates the u-values and determines the amount of non-zero u-values. It is important to note that increasing θ value results in a higher penalty for non-zero u-values. However, setting θ to excessively low values can cause too many examples to be assigned low u-values, resulting in a small training set for the NN that cannot effectively learn patterns. Conversely, high θ values may incentivize the model to select zero for all u-values, making the RRM model work as the traditional ERM. Therefore, one must search for a θ value that can outperform ERM.

Table 9 illustrates the relationship between θ and u-values; we take one experiment using RRM (ADH-LP) in various contamination levels and compare θ with mislabeled images excluded from the NN training. An image is considered excluded from training if its associated u-value achieves a value of $-1/N$, where N represents the total number of training observations and $1/N$ is the nominal probability.

Table 9. Penalty parameter and perturbation vector relationship in MASATI

ADH-LP / SGD RRM ($\mu = 0.5$)	Contamination levels							
	40% (737 mislabeled images)		30% (553 mislabeled images)		20% (368 mislabeled images)		10% (184 mislabeled images)	
Value of penalty θ	Model accuracy	# of mislabeled images excluded	Model accuracy	# of mislabeled images excluded	Model accuracy	# of mislabeled images excluded	Model accuracy	# of mislabeled images excluded
0.15	0.604	327	0.648	286	0.648	211	0.639	99
0.20	0.663	338	0.692	290	0.648	197	0.648	91
0.25	0.639	318	0.687	263	0.658	163	0.629	94
0.30	0.668	295	0.585	230	0.600	159	0.643	87
0.35	0.624	253	0.556	213	0.639	140	0.648	71

Looking at the 30% label contamination column in Table 8, we take a closer look at the cases where the best ($\theta = 0.20$) and the worst ($\theta = 0.35$) test accuracies were achieved. Starting with the worst case, where $\theta = 0.35$, Figure 22 displays the accuracy of training (red curve) and test (blue curve) of ERM (left) and RRM (right) over 500 epochs. The test accuracies are very comparable in both approaches, where RRM achieves 0.556 against 0.546 of ERM.

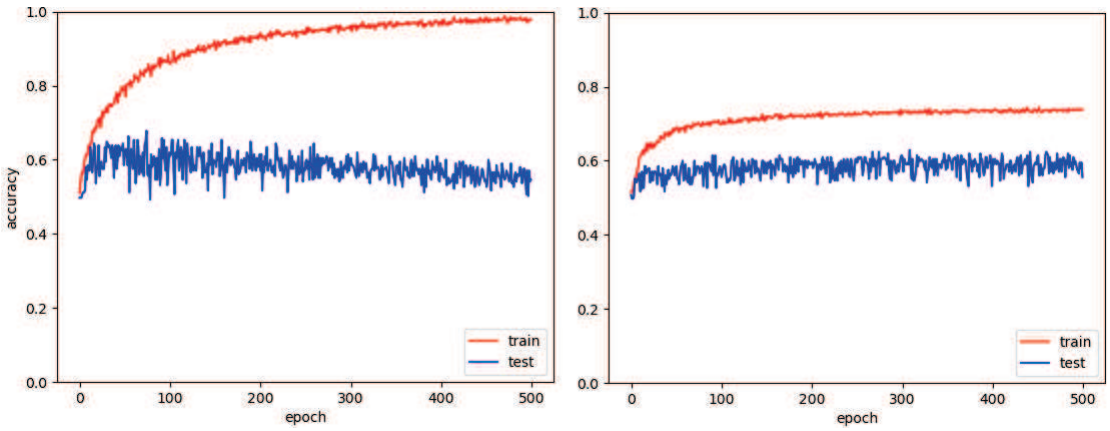

Figure 22. Training and test accuracy for ERM (left) and RRM ADH-LP/θ = 0.35 (right) on MASATI with 30% of contamination.

Now turning to Table 10 we have the report of the u-vector evolution in the two early updates and how it finished in its last update. The columns are labeled by the i-counter

value indicating in which update the u-vector is and show the distribution of u_i-values after their respective u-optimization across the 553 mislabeled images and 1,291 correctly labeled images.

During the initial iteration, some images from both label categories receive a u_i-value of $-2.7 \cdot 10^{-4}$. A larger number from both groups retains their initial u_i-value of zero, indicating no changes are made. With the second update, there is a discernible shift within the range of u-values as the algorithm seeks to adjust probabilities in its u-optimization process.

Upon the final update, the algorithm identifies 213 out of 553 incorrectly labeled images. These images are assigned a u_i-value of $-5.4 \cdot 10^{-4}$, effectively excluding them from further NN training. This exclusion stems from the fact that this u_i-value nullifies the nominal probability assigned to the images at the inception.

While a minor proportion of the correctly labeled images receive this lowest u_i-value, the majority (1,005 out of 1,291) retain their initial probability or a value close to it and continue contributing to the NN training process.

Table 10. Evolution of u-vector across ADH-LP / SGD in MASATI (30% of contamination and $\theta = 0.35$)

Nominal probability ($1/N$) $5.4 \cdot 10^{-4}$ u_i-values	Iteration number					
	i=1		i=2		i=49	
	mislabeled images	correct labeled images	mislabeled images	correct labeled images	mislabeled images	correct labeled images
>>0	0	1	0	1	1	1
≈ 0	304	813	266	782	338	1,005
$-1.5 \cdot 10^{-4}$	0	0	37	92	1	5
$-2.7 \cdot 10^{-4}$	249	477	38	41	0	1
$-4.0 \cdot 10^{-4}$	0	0	212	375	0	1
$-5.4 \cdot 10^{-4}$	0	0	0	0	213	278
Total of images	553	1,291	553	1,291	553	1,291

The scenario is different when we shift to the best accuracy of RRM where θ = 0.20. Figure 23 displays the accuracy of training (red curve) and test (blue curve) of ERM (right) and RRM (left) over 500 epochs. The test accuracies are different, and we clearly note the improvement of RRM against ERM. In that case, RRM achieves 0.692 against 0.546 of ERM. The accuracy plots show much less noise and more stability in the RRM test accuracy curve compared to the ERM and the previous RRM test accuracy curve.

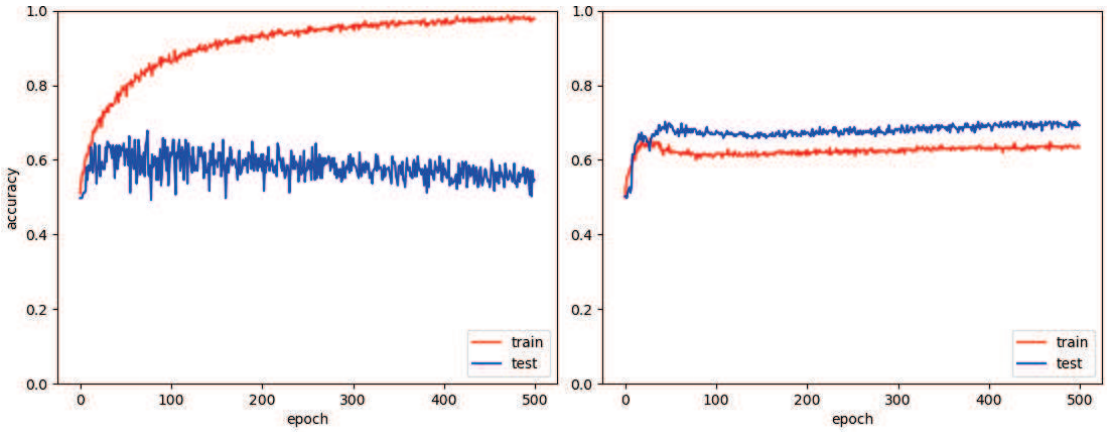

Figure 23. Training and test accuracy for ERM (left) and RRM ADH-LP/θ = 0.20 (right) on MASATI with 30% of contamination.

Table 11 displays the u-vector evolution of the RRM best case scenario in the two early updates and how it finished in its last update.

During the initial iteration, we have more images from both label categories receiving the u_i-value of $-2.7 \cdot 10^{-4}$, and the minority from both groups retaining their initial u_i-value of zero. In the second update just 93 of the 553 incorrectly labeled images remained with the initial u_i-value of zero. Observing RRM test accuracy in Figure 23, we notice that it stabilizes around epoch 75 and even slightly improves with additional epochs. This suggests that RRM may benefit from additional training past 500 epochs. Moreover, the stabilization of the test accuracy corresponds to the stabilization of u-value assignments.

Upon the final update, the algorithm identifies 290 out of 553 incorrectly labeled images, 77 more than the worst case. These images are assigned a u_i-value of $-5.4 \cdot 10^{-4}$, effectively excluding them from further NN training.

While a considerable proportion of the correctly labeled images receive this lowest u_i-value, the majority (863 out of 1,291) retain their initial probability or a value close to it and continue contributing to the NN training process.

Table 11. Evolution of u-vector across ADH-LP / SGD in MASATI (30% of contamination and $\theta = 0.20$)

Nominal probability $(1/N)$ $5.4 \cdot 10^{-4}$	Iteration number					
	$i = 1$		$i = 2$		$i = 49$	
u_i-values	mislabeled images	correct labeled images	mislabeled images	correct labeled images	mislabeled images	correct labeled images
>>0	0	1	0	1	0	1
≈ 0	133	453	93	370	260	863
$-1.5 \cdot 10^{-4}$	0	0	39	105	3	1
$-2.7 \cdot 10^{-4}$	420	837	40	83	0	0
$-4.0 \cdot 10^{-4}$	0	0	381	732	0	0
$-5.4 \cdot 10^{-4}$	0	0	0	0	290	426
Total of labels	553	1,291	553	1,291	553	1,291

B. AIRBUS DATASET

The Airbus Ship Detection dataset was part of a challenge realized in Kaggle in 2018 and comprises satellite images provided by the Airbus company. All files are RGB images with a resolution of 768 x 768 pixels, stored in JPG format.

The whole dataset contains a total of 18,392 images. We extracted a balanced random sample of this set to perform our analysis; We selected 10,428 images evenly divided between the “ship” and “no ship” categories, with 5,214 images in each category. Figure 24 illustrates some images of AIRBUS.

Figure 24. AIRBUS dataset images. Source: [73].

The MASATI dataset contains 2,049 images. In contrast, the AIRBUS dataset, containing nearly five times more images, allows us to evaluate our algorithm in a more data-rich environment. Both data-limited and data-rich situations are likely in practical scenarios. Therefore, analyzing our algorithms in a more data-rich environment adds an important dimension.

1. Accuracy Results with AIRBUS

In this section, we present the results of the AIRBUS dataset, focusing on the test accuracy score, which measures how effectively our models can classify images in the test set. We also provide a comprehensive analysis to interpret the observed performance.

In this dataset, the train-test split was done using an 80/20 proportion, which resulted in 8,340 images in the training set and 2,086 in the test set.

The final test accuracy results for ERM and RRM with ADH-SUB and ADH-LP using Adam optimizer are shown in Tables 12 and 13, respectively, at various levels of corruption.

Table 12. Final test accuracy in AIRBUS for ERM and RRM (ADH-SUB / Adam)

	Corrupted training data percentage				
Method	40%	30%	20%	10%	0%
ERM	0.588	0.657	0.729	0.797	0.867
RRM ($\lambda = 0.3$)					
$\theta = 0.15$	0.554	0.657	0.734	0.790	0.877
$\theta = 0.20$	0.555	0.643	0.744	0.796	0.873
$\theta = 0.25$	0.569	0.651	0.733	0.799	0.870
$\theta = 0.30$	0.545	0.673	0.741	0.809	0.877
$\theta = 0.35$	0.562	0.644	0.735	0.797	0.871

The values highlighted in gray represent the cases where RRM outperform or match ERM.

Upon evaluating the findings, a comparative analysis highlights the consistent performance of RRM over ERM, regardless of the penalty parameter selected. This is particularly evident across varying degrees of data corruption represented by the different percentage categories.

RRM has comparable, if not slightly superior, accuracy to ERM at all levels of corruption. This extends to the 0% corruption scenario, where RRM still holds up favorably against ERM, suggesting that RRM's effectiveness is not limited to handling corrupted data.

The performance of RRM varies depending on the θ value selected. Notably, all θ configurations demonstrate competitive performance, although certain θ values exhibit marginal advantages in specific corruption scenarios.

Regarding the overall best choices for the θ parameter, the $\theta = 0.25$ and $\theta = 0.30$ configurations stand out. Specifically, $\theta = 0.25$ offers promising results across all corruption levels, including higher contamination scenarios, signifying its adaptability and

resilience. On the other hand, the $\theta = 0.30$ configuration displays consistent performance across all corruption levels, thus asserting its robustness.

Table 13. Final test accuracy in AIRBUS for ERM and RRM (ADH-LP / Adam)

Method	Corrupted training data percentage				
	40%	30%	20%	10%	0%
ERM	0.588	0.657	0.729	0.797	0.867
RRM ($\mu = 0.5$)					
$\theta = 0.15$	0.602	0.692	0.758	0.831	0.868
$\theta = 0.20$	0.607	0.668	0.751	0.819	0.875
$\theta = 0.25$	0.616	0.690	0.763	0.823	0.867
$\theta = 0.30$	0.619	0.686	0.783	0.824	0.872
$\theta = 0.35$	0.602	0.695	0.767	0.819	0.872

The values highlighted in gray represent the cases where RRM outperform or match ERM.

Despite the data corruption level, RRM consistently delivers better accuracy than ERM. This performance advantage holds across all tested θ values.

In all corruption levels, the superior performance of RRM is evident, even in a no-corruption setting. This highlights the effectiveness of the RRM methodology not just in handling corrupted data but also in solving complex hidden data patterns.

At the 20% corruption level, RRM with a $\theta = 0.30$ demonstrates a substantial increase in accuracy, as high as 5.4% over ERM, which showcases the potential effectiveness of RRM when dealing with moderate data corruption. Similarly, the most remarkable improvement for the 30% corruption scenario was 3.8% with a $\theta = 0.35$.

Looking at the overall performance across different corruption levels, the $\theta = 0.25$ setting emerges as a consistently strong choice. This configuration delivers higher or equivalent accuracy compared to ERM across all contamination levels.

In summary, these findings underscore RRM's potential as a resilient and practical approach to handling various levels of data corruption, consistently matching or even

slightly outperforming the ERM. The analysis supports the robustness of RRM and its broader applicability in tasks that involve varying degrees of data corruption.

Now turning to the SGD optimizer, Tables 14 and 15 display the final test accuracy results for ERM and RRM in ADH-SUB and ADH-LP at various levels of corruption, respectively.

Table 14. Final test accuracy in AIRBUS for ERM and RRM (ADH-SUB / SGD)

Method	Corrupted training data percentage				
	40%	30%	20%	10%	0%
ERM	0.560	0.629	0.681	0.735	0.769
RRM ($\lambda = 0.3$)					
$\theta = 0.15$	0.544	0.628	0.699	0.723	0.793
$\theta = 0.20$	0.571	0.628	0.687	0.708	0.788
$\theta = 0.25$	0.553	0.630	0.685	0.728	0.776
$\theta = 0.30$	0.564	0.607	0.692	0.748	0.783
$\theta = 0.35$	0.574	0.620	0.703	0.752	0.785

The values highlighted in gray represent the cases where RRM outperform or match ERM.

At a high corruption level of 40%, the RRM approach, when using θ set to 0.20 and 0.35, displays slightly better accuracy than ERM. In situations of moderate corruption, precisely the 30% level, the performance of RRM and ERM is comparable, mainly when the $\theta = 0.15$ or $\theta = 0.20$ are employed for RRM. This comparable performance further showcases RRM's robustness across different corruption levels.

Interestingly, at a 20% corruption level, RRM's performance exceeds ERM's across all tested θ values, demonstrating its effectiveness in scenarios of lower data corruption. Furthermore, at a 10% corruption level, RRM with $\theta = 0.30$ and $\theta = 0.35$ again displays slightly better accuracies.

Even in the absence of corruption, RRM performed better in all θ configurations; it shows slightly better accuracies, asserting that RRM's efficiency.

The set of $\theta \in \{0.25, 0.30, 0.35\}$ consistently displays competitive performance. Sometimes, these θ configurations might deliver slightly lower accuracies than ERM. However, the differences are minor, making the performance broadly equivalent to ERM.

Table 15. Final test accuracy in AIRBUS for ERM and RRM (ADH-LP /SGD)

Method	Corrupted training data percentage				
	40%	30%	20%	10%	0%
ERM	0.560	0.629	0.681	0.735	0.769
RRM ($\mu = 0.5$)					
$\theta = 0.15$	0.603	0.739	0.745	0.774	0.764
$\theta = 0.20$	0.671	0.755	0.753	0.775	0.769
$\theta = 0.25$	0.687	0.733	0.757	0.769	0.767
$\theta = 0.30$	0.684	0.744	0.764	0.765	0.774
$\theta = 0.35$	0.661	0.747	0.764	0.770	0.779

The values highlighted in gray represent the cases where RRM outperform or match ERM.

Similar to the previous example the table shows that the RRM consistently delivers better accuracy than ERM. This performance advantage holds across all tested θ values.

At the 20% corruption level, the RRM method exhibits superior performance, particularly when configured with a θ value of 0.30. It shows a remarkable 8.3% improvement in accuracy over ERM. As the corruption level rises to 30% and $\theta = 0.20$, RRM demonstrates an impressive accuracy improvement of 12.6% compared to ERM. Finally, at the 40% corruption level, the robustness of RRM becomes even more evident. In the set $\theta \in \{0.20, 0.25, 0.30\}$, the RRM method delivers substantial accuracy improvements of 11.1%, 12.7%, and 12.4% over ERM, respectively.

In the lower corruption levels of 10% and 0%, the performance of RRM remains comparable or slightly superior to ERM. This underlines the flexibility of the RRM method, as it maintains strong performance even when the level of corruption decreases.

The most consistently effective choice of θ across the range of corruption levels appears to be $\theta = 0.30$. It is the best choice for managing different levels of corruption in

the Airbus dataset when using ADH-LP with SGD, as it consistently performs well, even in high corruption levels, and maintains similar results to other values in scenarios with 10% or no corruption.

2. *U*-optimization Analysis with AIRBUS

In the following section, we aim to examine the *u*-vector behavior in some AIRBUS cases. Table 16 illustrates the relationship between θ and excluded images by the perturbation vector using RRM (ADH-LP) in different levels of label contamination.

Table 16. Penalty parameter and perturbation vector relationship in AIRBUS

ADH-LP / SGD	Contamination levels in AIRBUS							
	40% (3,336 mislabeled images)		30% (2,502 mislabeled images)		20% (1,668 mislabeled images)		10% (834 mislabeled images)	
Value of penalty θ	Model accuracy	# of mislabeled images excluded	Model accuracy	# of mislabeled images excluded	Model accuracy	# of mislabeled images excluded	Model accuracy	# of mislabeled images excluded
0.15	0.603	1,985	0.739	1,812	0.745	1,210	0.774	621
0.20	0.671	2,108	0.755	1,767	0.753	1,200	0.775	587
0.25	0.687	2,127	0.733	1,701	0.757	1,158	0.769	573
0.30	0.684	2,019	0.744	1,636	0.764	1,154	0.765	564
0.35	0.661	1,737	0.747	1,635	0.764	1,109	0.770	538

Looking at the 40% label contamination column in Table 15, when the $\theta = 0.25$ we have the highest number of mislabeled images excluded (2,127 images) and the best test accuracy achieved (0.687) at this contamination level. The worst accuracy (0.603) is achieved when $\theta = 0.15$. Figure 25 displays the accuracy of training (red curve) and test (blue curve) of ERM (left) and RRM (right) over 500 epochs. The test accuracies differ in 4.3%, where RRM achieves 0.603 against 0.560 of ERM.

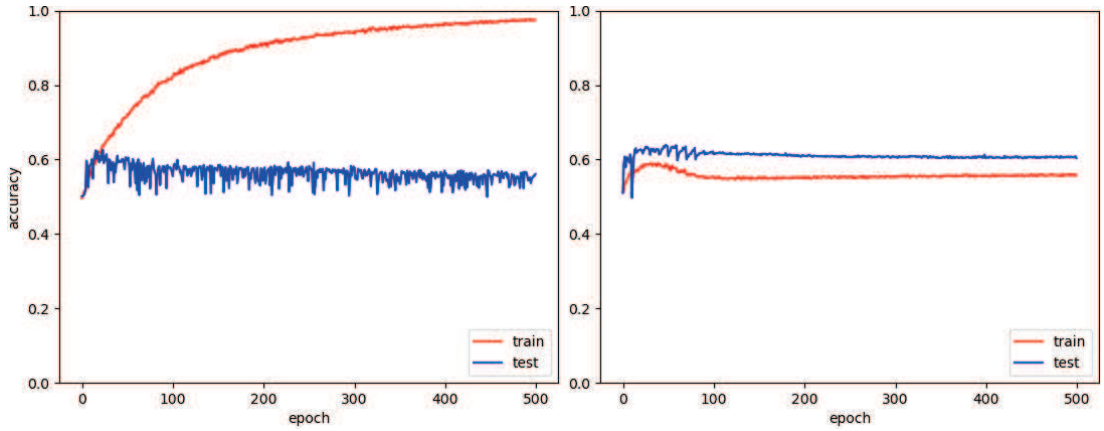

Figure 25. Training and test accuracy for ERM (left) and RRM ADH-LP/θ = 0.15 (right) on AIRBUS with 40% of contamination.

Table 17 reports the u-vector evolution of u-optimization across 3,336 mislabeled images and 5,004 correctly labeled images, when $\theta = 0.15$.

During the initial iteration, as the penalty is very low ($\theta = 0.15$) the vast majority of images from both label categories receives the u_i-value of $-6.0 \cdot 10^{-5}$, and the rest retains their initial u_i-value of zero. In the second update just 64 of the 3336 incorrected labeled images remained with the initial u_i-value of zero. Eventually, over iterations, the u-value distribution stabilizes, along with test accuracy.

At the final update, the algorithm identifies 1985 out of 3336 incorrectly labeled images. These images are assigned a u_i-value of $-12 \cdot 10^{-5}$, effectively excluding them from further NN training, as this value represents the nominal probability in AIRBUS dataset. In the correct labeled images column, we notice a very similar number of images (1860) receives the lowest u_i-value and the majority (3,135 out of 5,004) retain their initial probability or a value close to it and contributes to the NN training process.

Table 17. Evolution of u-vector across ADH-LP / SGD in AIRBUS (40% of contamination and $\theta = 0.15$)

Nominal probability ($1/N$) $12.0 \cdot 10^{-4}$	Iteration number					
	i=1		i=2		i=49	
u_i-values	mislabeled images	correct labeled images	mislabeled images	correct labeled images	mislabeled images	correct labeled images
>>0	0	1	0	1	0	2
≈ 0	131	264	64	194	1,346	3,135
$-3.0 \cdot 10^{-5}$	0	0	24	63	2	1
$-6.0 \cdot 10^{-5}$	3,205	4,739	67	153	3	6
$-9.0 \cdot 10^{-5}$	0	0	3,181	4,593	0	0
$-12.0 \cdot 10^{-5}$	0	0	0	0	1,985	1,860
Total of images	3,336	5,004	3,336	5,004	3,336	5,004

We also analyze the case where $\theta = 0.25$, and it shows how crucial is the choice of our penalty parameter θ. Using RRM, this experiment reaches the best test accuracy of 0.687, an improvement of 12.7% in accuracy compared to the 0.560 of ERM. Figure 26 shows the accuracy of training (red curve) and test (blue curve) of ERM (left) and RRM (right) over 500 epochs.

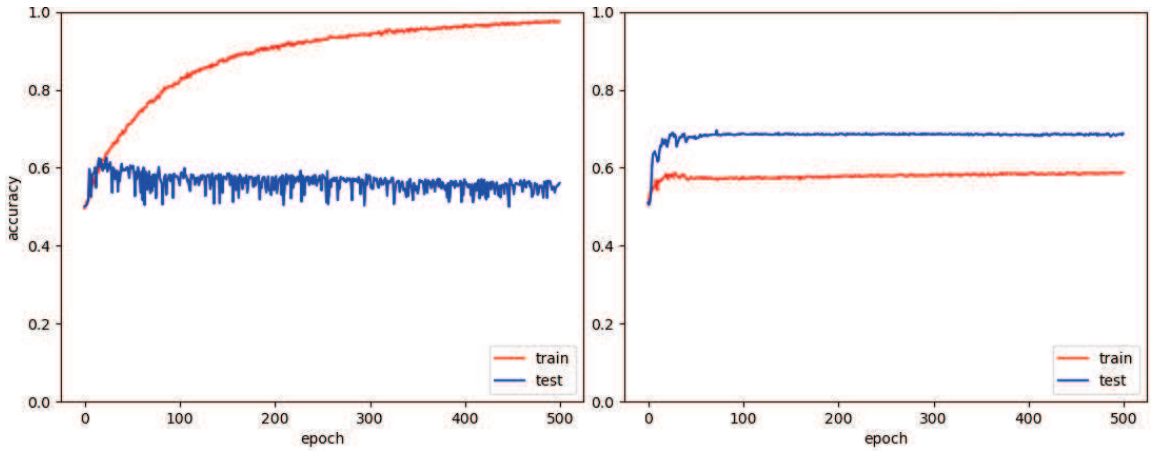

Figure 26. Training and test accuracy for ERM (left) and RRM ADH-LP/θ = 0.25 (right) on AIRBUS with 40% of contamination.

Table 18 reports the u-vector evolution of u-optimization across 3,336 mislabeled images and 5,004 correctly labeled images, when $\theta = 0.25$.

During the initial iteration, a considerable number of images from both label categories receives the u_i-value of $-6.0 \cdot 10^{-5}$, 2,917 of 3,336 in the mislabeled portion and 3747 of 5004 in the correct labeled portion, the rest retains their initial u_i-value of zero. In the second update, until the run reaches 50 epochs, we have a picture of how the u-optimization process of assigning more mislabeled images with lower and lower u-values iteratively coincides with iterative improvements in test accuracy. Looking at Figure 26, this appears to manifest as "sawtooth" jumps improvements in test accuracy early iterations.

At the final update, the algorithm identifies 2,129 out of 3,336 incorrectly labeled images, 144 more than when $\theta = 0.15$. These images are assigned a u_i-value of $-12 \cdot 10^{-5}$, and they are effectively excluded of the NN learning process. Otherwise, in the correct labeled images column, the number of images retaining zero u_i-value increased to 3,591, which guarantees 456 more correctly labeled images to the NN training process.

Table 18. Evolution of u-vector across ADH-LP / SGD in AIRBUS (40% of contamination and $\theta = 0.25$)

Nominal probability (1/*N*) $12.0 \cdot 10^{-4}$	Iteration number					
	i=1		*i*=2		*i*=49	
u_i-values	mislabeled images	correct labeled images	mislabeled images	correct labeled images	mislabeled images	correct labeled images
>>0	1	0	1	0	1	0
≈ 0	418	1,257	312	1,117	1,192	3,591
$-3.0 \cdot 10^{-5}$	0	0	142	590	7	1
$-6.0 \cdot 10^{-5}$	2,917	3,747	106	140	2	2
$-9.0 \cdot 10^{-5}$	0	0	2,775	3,157	5	6
$-12.0 \cdot 10^{-5}$	0	0	0	0	2,129	1,404
Total of images	3,336	5,004	3,336	5,004	3,336	5,004

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

In this study, we offer a different strategy for training Neural Networks (NNs) that deviates from the traditional Empirical Risk Minimization (ERM) approach. Our method, known as Rockafellian Risk Minimization (RRM), yields a more robust model that may exceed the performance of ERM in situations where the data is somewhat contaminated with label noise.

We apply our methodology to two separate datasets (MASATI and AIRBUS). We settle on a reasonable NN architecture for both sets and conduct our analysis under two choices of optimizer configurations (Adam and SGD). In each case, we compare ERM against RRM ADH-LP and RRM ADH-SUB.

In the context of MASATI dataset, when using the Adam optimizer context, ADH-SUB and ADH-LP benefit from the RRM method, with appropriate θ values outperforming or matching ERM. For ADH-SUB, the best balance between handling data corruption and optimizing performance is found at θ values of 0.20-0.25. ADH-LP's balance is observed at θ values of 0.15-0.25. Shifting to SGD optimizer context, RRM also performs well with ADH-SUB and ADH-LP, specifically when data corruption ranges from 20–40%. The optimal θ values are approximately 0.30-0.35 for ADH-SUB and 0.20-0.30 for ADH-LP.

While both the ADH-SUB and ADH-LP algorithms benefit from the RRM method with an appropriately chosen θ across varying levels of data corruption and different optimizers (Adam and SGD), ADH-LP consistently emerges as the superior performer. Whether operating under Adam or SGD, ADH-LP strikes an exceptional balance between handling data corruption and optimizing performance. ADH-LP excels in the SGD context, showing remarkable resilience against data corruption while maintaining high-performance levels. This unique ability to delay performance degradation in the face of high data corruption sets ADH-LP apart as the preferred algorithm for both the Adam and SGD optimizer contexts.

For the AIRBUS dataset, using Adam in RRM for ADH-SUB and ADH-LP exhibits competitive or superior performance compared to the ERM method across all corruption levels. Specifically, the optimal θ values were around 0.25 and 0.30 for ADH-SUB and 0.25 for ADH-LP. This demonstrates RRM's robustness in handling data corruption and discovering complex hidden data patterns. For the SGD context, RRM again shows a similar trend. In the ADH-SUB architecture, RRM performance is on par or better than ERM, with $\theta \in \{0.25, 0.30, 0.35\}$ offering competitive performance across all corruption levels. For the ADH-LP structure, RRM outperforms ERM, in almost all corruption scenarios (20%, 30%, 40%), especially with $\theta = 0.30$, showing impressive improvements of up to 12.6%.

An intriguing observation from the AIRBUS dataset is the superior performance of ADH-SUB in scenarios where we do not introduce label contamination in the training set. In these cases, it achieves the highest accuracy in both CNN architectures, suggesting that RRM can effectively manage complex patterns within the dataset. This capability allows it to outperform the traditional ERM method.

To conclude, RRM demonstrates its potential as a robust and resilient method for handling varying degrees of label corruption in the AIRBUS dataset. In all cases, the ADH-LP architecture under the SGD optimizer performed remarkably well, showing the ability to delay performance degradation even under high corruption levels, establishing it as the top choice across all scenarios.

Overall, the analysis of our methodologies across the MASATI and AIRBUS datasets, with varying optimizers and degrees of data corruption, consistently establishes the ADH-LP as the superior performer. The key strength of ADH-LP lies in its exceptional balance between handling data corruption and optimizing performance, demonstrating remarkable resilience against label contamination while maintaining high-performance levels. Even under high corruption levels, this ability to delay performance degradation sets ADH-LP apart as the preferred algorithm in Adam and SGD optimizer contexts. With respect to total runtimes of the RRM ADH-SUB method tends to complete the u-optimization phase slightly quicker than ADH-LP. Nonetheless, the choice of algorithm should be more than just a matter of processing time. In terms of test accuracy-

performance, ADH-LP stands out as the overall best choice, offering robustness and resilience despite the slightly longer processing time in the face of varying degrees of data corruption.

B. FUTURE WORK

Despite the notable improvements in accuracy and robustness achieved through the implementation of the RRM method, there is still room for improvements. A key aspect yet to be fully investigated is the u-optimization process. In the current setup, u-optimization excludes up to 64% of mislabeled images in all experiments where RRM outperforms ERM. Perhaps a greater percentage can be achieved by employing novel approaches in the *u*-optimization procedure.

Another point of interest is the further refinement of the ADH-SUB algorithm. This algorithm shows promise, especially regarding computational time, as it runs faster than ADH-LP. However, when considering accuracy levels, it falls short compared to ADH-LP. Thus, further research and optimization efforts are needed to enhance the performance of ADH-SUB to the level of its counterpart or beyond, creating an overall more efficient and accurate solution.

As we advance toward more autonomous solutions, there is a rising need to automate ship detection accurately. Maritime surveillance capabilities can be significantly improved by integrating advanced models like the enhanced ADH-LP and ADH-SUB algorithms in automated surveillance systems, particularly those employing unmanned vehicles.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] M. Wiesebron, “Blue Amazon: Thinking the Defense of Brazilian Maritime Territory,” *Austral: Brazilian Journal of Strategy & International Relations,* vol. 2, no. 3, pp. 101–124, Jan-Jun. 2013 [Online]. Available: https://www.files.ethz.ch/isn/166009/37107-147326-1-PB.pdf#page=102

[2] Brazilian Navy Command, “Brazilian Navy Naval Policy.” Accessed: Apr. 09, 2023 [Online]. Available: https://www.naval.com.br/blog/wp-content/uploads/2019/04/PoliticaNavalMB.pdf

[3] Interdisciplinary Observatory on Climate Change, “Last Frontier at Sea.” Accessed: Apr. 07, 2023 [Online]. Available: https://obsinterclima.eco.br/mapas/ultima-fronteira-no-mar/

[4] Brazilian Navy Social Communication Center, “Blue Amazon – The Heritage Brazilian at Sea.” *Villegagnon Journal Supplement – VII Academic Congress on National Defense.* Accessed: Apr. 09, 2023 [Online]. Available: http://www.redebim.dphdm.mar.mil.br/vinculos/000006/00000600.pdf

[5] Andrade et al., “DP 0261 – Blue Amazon Management System (SisGAAz): Sovereignty, Surveillance and Defense of the Brazilian Jurisdictional Waters,” Discussion Paper. Instituto de Pesquisa Economica Aplicada – IPEA, pp. 1–35, Dec. 16, 2021 [Online]. Available: http://dx.doi.org/10.38116/dp261

[6] I. Andrade et al., “DP 2471 – Science, Technology and Innovation in The Brazilian Navy’s Strategic Programs,” Discussion Paper. Instituto de Pesquisa Economica Aplicada – IPEA, Apr, 2019 [Online]. Available: https://www.researchgate.net/publication/335925079_Ciencia_Tecnologia_e_Inovacao_nos_Programas_Estrategicos_da_Marinha_do_Brasil

[7] K. P. Murphy, *Machine Learning: A Probabilistic Perspective* (Adaptive Computation and Machine Learning Series). Cambridge, MA: MIT Press, 2012.

[8] RainerGewalt, “Supervised vs. unsupervised vs. reinforcement learning – the fundamental differences,” *Fly spaceships with your mind.* Accessed: Apr. 10, 2023 [Online]. Available: https://starship-knowledge.com/supervised-vs-unsupervised-vs-reinforcement

[9] F. Sohil, M. U. Sohali, and J. Shabbir, *An Introduction to Statistical Learning with Applications in R,* Statistical Theory and Related Fields, vol. 6, no. 1. Informa UK Limited, Sep. 26, 2021. doi: 10.1080/24754269.2021.1980261.

[10] Petercour, “Machine Learning Classification vs. Regression,” *DEV Community*. Accessed: Apr. 10, 2023 [Online]. Available: https://dev.to/petercour/machine-learning-classification-vs-regression-1gn

[11] S. Gunjal, “Logistic regression from scratch with python,” *Quality Tech Tutorials*. Accessed: Apr. 24, 2023 [Online]. Available: https://satishgunjal.com/binary_lr/

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA, MIT Press, 2016 [Online]. Available: http://www.deeplearningbook.org

[13] A. Burkov, *The Hundred-Page Machine Learning Book*. Andriy Burkov, 2019 [Online]. Available: http://ema.cri-info.cm/wp-content/uploads/2019/07/2019BurkovTheHundred-pageMachineLearning.pdf

[14] S. S. Haykin, *Neural Networks and Learning Machines*, 3. ed. New York Munich: Prentice-Hall, 2009.

[15] M. Banoula “What is Perceptron? A beginner’s guide [updated]: Simplilearn,” *Simplilearn.com.* Accessed: Apr. 24, 2023 [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron

[16] R. Shanmugamani, *Deep Learning for Computer Vision: Expert techniques to train advanced neural networks using TensorFlow and Keras*. Birmingham, UK: Packt Publishing, 2018.

[17] “Activation function,” *Activation Function – AI Wiki*. Accessed: Apr. 24, 2023 [Online]. Available: https://machine-learning.paperspace.com/wiki/activation-function

[18] R. Szeliski, *Computer Vision: Algorithms and Applications.* 2nd ed. England: Springer London Ltd, 2022.

[19] L. J. Varghese, S. S. Jacob, C. Sundar, and J. R. I, “Design and Implementation of a Machine Learning Assisted Smart Wheelchair in an IoT Environment.” Research Square Platform LLC, Jun. 03, 2021. doi: 10.21203/rs.3.rs-490123/v1.

[20] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun, A Guide to Convolutional Neural Networks for Computer Vision. Springer International Publishing, 2018 [Online]. Available: http://dx.doi.org/10.1007/978-3-031-01821-3

[21] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, “Review of Image Classification Algorithms Based on Convolutional Neural Networks,” Remote Sensing, vol. 13, no. 22. MDPI AG, p. 4712, Nov. 21, 2021 [Online]. Available: http://dx.doi.org/10.3390/rs13224712

[22] R. Torén, “Comparing CNN methods for detection and tracking of ships in satellite images,” M.S. thesis, Dept. of Comp. and Information Science Linköping University, Linköping, SWE, 2020.

[23] “Introduction to automatic encoders | TensorFlow Core,” *TensorFlow*. Accessed: Apr. 28, 2023 [Online]. Available: https://www.tensorflow.org/tutorials/generative/autoencoder?hl=pt-br

[24] “Convolutional Neural Networks Cheatsheet,” *CS 230 – Convolutional Neural Networks Cheatsheet*. Accessed: Apr. 28, 2023 [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks

[25] J. O. Royset, L. L. Chen, and E. Eckstrand, “Rockafellian Relaxation in Optimization under Uncertainty: Asymptotically Exact Formulations.” *arXiv*, 2022 [Online]. Available: https://arxiv.org/abs/2204.04762

[26] M. E. Maron, “Automatic Indexing: An Experimental Inquiry,” *J. ACM*, vol. 8, no. 3, pp. 404–417, Jul. 1961, doi: 10.1145/321075.321084.

[27] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.

[28] J. R. Quinlan, “Induction of Decision Trees,” *Mach. Lang.*, vol. 1, no. 1, pp. 81–106, Mar. 1986, doi: 10.1023/A:1022643204877.

[29] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[30] Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997, doi: 10.1006/jcss.1997.1504.

[31] L. Breiman, “Random Forests,” *Mach. Lang.*, vol. 45, no. 1, pp. 5–32, Outubro 2001, doi: 10.1023/A:1010933404324.

[32] G. Hinton, S. Osindero, and Y.-W. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” *Neural Comput.*, vol. 18, pp. 1527–54, Aug. 2006, doi: 10.1162/neco.2006.18.7.1527.

[33] W. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, vol. 5, pp.49-50, 1943.

[34] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” Psychological Review, vol. 65, no. 6. American Psychological Association (APA), pp. 386–408, 1958 [Online]. Available: http://dx.doi.org/10.1037/h0042519

[35] K. Steinbuch and B. Widrow, “A Critical Comparison of Two Kinds of Adaptive Classification Networks,” IEEE Transactions on Electronic Computers, vol. EC-14, no. 5. Institute of Electrical and Electronics Engineers (IEEE), pp. 737–740, Oct. 1965 [Online]. Available: http://dx.doi.org/10.1109/PGEC.1965.264220

[36] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088. Springer Science and Business Media LLC, pp. 533–536, Oct. 1986 [Online]. Available: http://dx.doi.org/10.1038/323533a0

[37] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[38] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, Mar. 2010, pp. 249–256. Accessed: May 08, 2023 [Online]. Available: https://proceedings.mlr.press/v9/glorot10a.html

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2012. Accessed: May 08, 2023 [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[40] C. Szegedy *et al.*, “Going Deeper with Convolutions.” arXiv, Sep. 16, 2014. Accessed: May 08, 2023 [Online]. Available: http://arxiv.org/abs/1409.4842

[41] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv, Apr. 10, 2015. Accessed: May 08, 2023 [Online]. Available: http://arxiv.org/abs/1409.1556

[42] G. A. Shadeed, M. A. Tawfeeq, and S. M. Mahmoud, “Automatic Medical Images Segmentation Based on Deep Learning Networks,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 870, no. 1, p. 012117, Jun. 2020, doi: 10.1088/1757-899X/870/1/012117.

[43] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[44] V. Fortunati, “Deep learning applications in radiology: a deep dive on classification.” Accessed May 09, 2023 [Online]. Available: https://www.quantib.com/blog/deep-learning-applications-in-radiology/classification

[45] A. Veit, M. J. Wilber, and S. Belongie, “Residual Networks Behave Like Ensembles of Relatively Shallow Networks,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2016. Accessed: Jun. 07, 2023 [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/hash/37bc2f75bf1bcfe8450a1a41c200364c-Abstract.html

[46] B. Hanin, “Which Neural Net Architectures Give Rise To Exploding and Vanishing Gradients?” arXiv, Oct. 26, 2018. Accessed: Jun. 07, 2023 [Online]. Available: http://arxiv.org/abs/1801.03744

[47] B. J. Tetreault, “Use of the Automatic Identification System (AIS) for maritime domain awareness (MDA),” in *Proceedings of OCEANS 2005 MTS/IEEE*, Sep. 2005, pp. 1590–1594 Vol. 2. doi: 10.1109/OCEANS.2005.1639983.

[48] Z. Yassir, A. Abdelali, and B. Mohammed, “Correlations of Biomechanical Characteristics with Ball Speed in Penalty Corner Push-In A Comparison of AIS, X-Band Marine Radar Systems and Camera Surveillance Systems in the Collection of Tracking Data,” vol. 7, no. 4, 2020.

[49] M. Ma, J. Chen, W. Liu, and W. Yang, “Ship Classification and Detection Based on CNN Using GF-3 SAR Images,” *Remote Sens.*, vol. 10, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/rs10122043.

[50] N. Ødegaard, A. O. Knapskog, C. Cochin, and J.-C. Louvigne, “Classification of ships using real and simulated data in a convolutional neural network,” in *2016 IEEE Radar Conference (RadarConf)*, May 2016, pp. 1–6. doi: 10.1109/RADAR.2016.7485270.

[51] C. Bentes, D. Velotto, and B. Tings, “Ship Classification in TerraSAR-X Images With Convolutional Neural Networks,” *IEEE J. Ocean. Eng.*, vol. 43, no. 1, pp. 258–266, Jan. 2018, doi: 10.1109/JOE.2017.2767106.

[52] A.-J. Gallego, A. Pertusa, and P. Gil, “Automatic Ship Classification from Optical Aerial Images with Convolutional Neural Networks,” *Remote Sens.*, vol. 10, no. 4, p. 511, Mar. 2018, doi: 10.3390/rs10040511.

[53] H. Fang, M. Chen, X. Liu, and S. Yao, “Infrared Small Target Detection with Total Variation and Reweighted ℓ 1 Regularization,” *Math. Probl. Eng.*, vol. 2020, pp. 1–19, Jan. 2020, doi: 10.1155/2020/1529704.

[54] C. Kanellakis and G. Nikolakopoulos, “Survey on Computer Vision for UAVs: Current Developments and Trends,” *J. Intell. Robot. Syst.*, vol. 87, no. 1, pp. 141–168, Jul. 2017, doi: 10.1007/s10846-017-0483-z.

[55] G. Cruz and A. Bernardino, “Aerial Detection in Maritime Scenarios Using Convolutional Neural Networks,” in *Advanced Concepts for Intelligent Vision Systems*, J. Blanc-Talon, C. Distante, W. Philips, D. Popescu, and P. Scheunders, Eds., in Lecture Notes in Computer Science, vol. 10016. Cham: Springer International Publishing, 2016, pp. 373–384. doi: 10.1007/978-3-319-48680-2_33.

[56] L.-Y. Lo, C. H. Yiu, Y. Tang, A.-S. Yang, B. Li, and C.-Y. Wen, “Dynamic Object Tracking on Autonomous UAV System for Surveillance Applications,” *Sensors*, vol. 21, no. 23, p. 7888, Nov. 2021, doi: 10.3390/s21237888.

[57] E. Lygouras, N. Santavas, A. Taitzoglou, K. Tarchanidis, A. Mitropoulos, and A. Gasteratos, “Unsupervised Human Detection with an Embedded Vision System on a Fully Autonomous UAV for Search and Rescue Operations,” *Sensors*, vol. 19, no. 16, p. 3542, Aug. 2019, doi: 10.3390/s19163542.

[58] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.” arXiv, May 09, 2016. Accessed: May 20, 2023 [Online]. Available: http://arxiv.org/abs/1506.02640

[59] “WACV 2023 – Maritime Workshop.” https://seadronessee.cs.uni-tuebingen.de/wacv23 (accessed May 20, 2023).

[60] R. J. Hickey, “Noise modelling and evaluating learning from examples,” *Artif. Intell.*, vol. 82, no. 1–2, pp. 157–179, Apr. 1996, doi: 10.1016/0004-3702(94)00094-8.

[61] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, “Learning from Noisy Labels with Deep Neural Networks: A Survey.” arXiv, Mar. 09, 2022. Accessed: May 21, 2023 [Online]. Available: http://arxiv.org/abs/2007.08199

[62] T. Liu and D. Tao, “Classification with Noisy Labels by Importance Reweighting,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 3, pp. 447–461, Mar. 2016, doi: 10.1109/TPAMI.2015.2456899.

[63] C. Yang, Q. Wu, H. Li, and Y. Chen, “Generative Poisoning Attack Method Against Neural Networks.” arXiv, Mar. 03, 2017. Accessed: May 22, 2023 [Online]. Available: http://arxiv.org/abs/1703.01340

[64] M. Ren, W. Zeng, B. Yang, and R. Urtasun, “Learning to Reweight Examples for Robust Deep Learning.”

[65] S. Thulasidasan, T. Bhattacharya, J. Bilmes, G. Chennupati, and J. Mohd-Yusof, “Combating Label Noise in Deep Learning Using Abstention.” arXiv, Aug. 01, 2019. Accessed: May 21, 2023 [Online]. Available: http://arxiv.org/abs/1905.10964

[66] L. Chen *et al.*, “Deep Learning with Label Noise: A Hierarchical Approach.” arXiv, May 27, 2022. Accessed: May 21, 2023 [Online]. Available: http://arxiv.org/abs/2205.14299

[67] H. Narasimhan, A. K. Menon, W. Jitkrittum, and S. Kumar, “Learning to reject meets OOD detection: Are all abstentions created equal?” arXiv, Jan. 31, 2023. Accessed: May 21, 2023 [Online]. Available: http://arxiv.org/abs/2301.12386

[68] C. Ni, N. Charoenphakdee, J. Honda, and M. Sugiyama, “On the Calibration of Multiclass Classification with Rejection.” arXiv, Oct. 29, 2019. Accessed: May 22, 2023 [Online]. Available: http://arxiv.org/abs/1901.10655

[69] H. G. Ramaswamy, A. Tewari, and S. Agarwal, “Consistent algorithms for multiclass classification with an abstain option,” *Electron. J. Stat.*, vol. 12, no. 1, Jan. 2018, doi: 10.1214/17-EJS1388.

[70] J. Katz-Samuels, J. Nakhleh, R. Nowak, and Y. Li, “Training OOD Detectors in their Natural Habitats.” arXiv, Jun. 28, 2022. Accessed: May 22, 2023 [Online]. Available: http://arxiv.org/abs/2202.03299

[71] J. O. Royset and R. J.-B. Wets, *An Optimization Primer*. in Springer Series in Operations Research and Financial Engineering. Cham: Springer International Publishing, 2021. doi: 10.1007/978-3-030-76275-9.

[72] W. Wang and M. Á. Carreira-Perpiñán, “Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application.” arXiv, Sep. 06, 2013. Accessed: Apr. 22, 2023 [Online]. Available: http://arxiv.org/abs/1309.1541

[73] “Airbus Ship Detection Challenge.” https://kaggle.com/competitions/airbus-ship-detection (accessed May 30, 2023).

[74] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. Accessed: May 08, 2023 [Online]. Available: http://arxiv.org/abs/1412.6980

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California