

# **Ensemble Learning and Pruning for Image Classification in Maritime Management Systems**

**Candidate name:** Niusha Mesgaribarzi

**University of South-Eastern Norway**  
Faculty of Technology, Natural Sciences and Maritime Sciences

**MASTER THESIS**

**May 2024**

## **Abstract**

In this thesis, the exploration of how maritime management systems can be optimized is considered, and it also revisits cutting-edge deep learning architectures based on ensemble learning techniques. Within the analysis and application of deep learning approaches, the study aims to fill gaps in maritime system methodologies. Those are: 1) Customizing more deep learning models for different maritime operations; and 2) Innovative ensemble learning can improve model performance and accuracy. A novel framework combining ensemble learning with pruning techniques is presented in this study. The goal of the framework is to improve efficient model selection while minimizing computational resource needs. Following diverse testing environments, for methodological assessment and considering various deep learning architectures within marine and maritime datasets, Kaggle's public datasets platform is used. In this study, the consideration is given to enhancing ensemble models through strategic pruning. To create a more efficient predictive system, this involves carefully selecting and combining individual models within the ensemble. This approach proves effective across various marine and maritime situations, demonstrating its adaptability to different requirements. By offering innovative AI-based solutions for decision support and operational optimization, this study also contributes to maritime management systems, while the findings and methodologies proposed open avenues for future research and development in this area.

Several contributions offered by this thesis. First, covering topics from machine learning to deep learning, focusing on convolutional neural networks, and progressing towards ensemble learning and pruning techniques as a literature review. Secondly, the "Maritime mAnaGement eNsemble leArning sysTem (MAGNAT)" and the "Maritime mAnaGement eNsemble prUrning sysTem (MAGNUT) are proposed as two novel frameworks and strategies for enhancing model selection and performance optimization in maritime management systems. MAGNAT uses an ensemble aggregation strategy by assigning weighted factors based on their historical performance. MAGNUT is also an intelligent aggregation strategy that combines ensemble and pruning techniques to identify the minimal subset of nonredundant models within the ensemble. Thirdly, a diverse range of maritime datasets, including jellyfish datasets relevant to marine science and ship datasets related to maritime technology, is utilized for practical evaluation and calculation of the efficiency results presented in the thesis, along with hands-on Python programming.

## **Keywords**

Deep Learning (DL)

Convolution Neural Networks (CCN)

Artificial Intelligence (AI)

Image Classification

Aggregating Models

Ensemble Learning

Ensemble Pruning

## **Acknowledgment**

I would like to express my sincere gratitude to Associate Professor Djenouri, my supervisor, for his support, guidance, and availability throughout the entirety of my master's thesis journey. Special appreciation is extended to Nabil Belbachir, my co-supervisor at NORCE (Norwegian Research Center), who is sharing his feedback for improving the quality of my work.

I am grateful to all members of the "SMAUG: Smart Maritime And Underwater Guardian" project available at <https://smaug-horizon.eu/> for providing me with the opportunity to collaborate and work alongside them on this project, supporting my research activities throughout my master's thesis period. The shared experiences and collective effort have greatly enriched my academic and research endeavours. Additionally, I am thankful to SMAUG for offering me the opportunity to begin my career in academia as a student research assistant at USN.

I extend my heartfelt thanks to my family, including my father, mother, and especially my sister, who has always been my rock, for their unwavering support and love throughout my master's studies and during the writing process of my master's thesis. Special appreciation goes to my husband for his motivation and encouragement throughout my studies and thesis writing. Lastly, I am grateful to my dear friend, Mahtab, for her continuous support throughout this study and my master's thesis activities.

I am thankful to all who have directly and indirectly supported me, playing an integral role in this academic endeavour and contributing to the successful completion of my master's thesis. Last but not least, to mention that I utilized ChatGPT and Grammarly for grammar correction and further improvement of my writing.

# Table of Contents

Abstract .....	2
Acknowledgment .....	4
Chapter 1: Introduction.....	12
1.1. Research Background And Motivation .....	13
1.2. Research Questions.....	14
1.3. Research Purpose .....	14
1.4. Research Scope .....	15
1.5. Research Structure and Contents .....	15
Chapter 2: Literature Review (PART I), Machine Learning For Image Classification ....	17
2.1. Introduction.....	17
2.2. Difference Between Conventional Machine Learning And Deep Learning Approach .....	18
2.3. Conventional Machine Learning .....	19
2.3.1. Feature Extraction .....	19
2.3.2. Classifier.....	23
2.4. Deep Learning.....	27
2.4.1. Type Of Layer:.....	30
2.4.2. Filter Hyperparameters.....	31
2.4.3. Dimensions of a Filter .....	32
2.4.4. Stride .....	32
2.4.5. Zero-Padding.....	33
2.4.6. Architectural Design In CNN.....	33
2.5. Image Classification In Maritime Applications .....	37
2.6. Conclusion .....	38
Chapter 3: Literature review (PART II), Ensemble Learning and Pruning .....	40
3.1 Introduction.....	40
3.2 Problem Definitions.....	40
3.3 Ensemble Learning Techniques .....	42
3.3.1 Bootstrap Aggregating (Bagging).....	43
3.3.2 Boosting.....	44
3.3.3 Stacking.....	46

3.3.4	Gradient Boosting .....	48
3.4	Ensemble Pruning Techniques .....	49
3.4.1	Dynamic Model Selection.....	50
3.4.2	Model Pruning By Diversity.....	52
3.4.3	Ensemble Learning .....	56
3.4.4	Ensemble Pruning.....	57
3.5	Conclusion .....	57
Chapter 4: Methodology Design (MAGNAT and MAGNUT) .....		59
4.1	Introduction.....	59
4.2	Explanation of Research Questions.....	59
4.3	Data Preprocessing .....	60
4.3.1	Resizing Images .....	61
4.3.2	Normalization .....	62
4.3.3	Image Conversion to Grayscale.....	62
4.4	Single-based Solution.....	63
4.5	Maritime mAnaGement eNsemble leArning sysTem (MAGNAT) .....	70
4.5.1	Ensemble Aggregation .....	71
4.6	Maritime mAnaGement eNsemble prUrning sysTem (MAGNUT) .....	72
4.6.1	Knowledge Base Creation.....	73
4.6.2	Intelligent Aggregation: .....	73
4.7	Challenges And Discussions .....	77
4.8	Conclusion .....	78
Chapter 5: Experiments .....		80
5.1	Introduction.....	80
5.2	Dataset Description.....	80
5.2.1	Jellyfish Dataset .....	80
5.2.2	Ship Dataset .....	81
5.3	Model Visualizations .....	82
5.4	Performance Evaluation.....	83
5.4.1	Evaluation of the models on the Jellyfish dataset.....	84
5.4.2	Evaluation of the models on the ship dataset.....	87
5.5	Further Results .....	88

5.5.1	Advanced Model Architecture .....	88
5.5.2	Result Of Using Advanced Models On The Jellyfish Dataset.....	92
5.5.3	Result Of Using Advanced Models On The Ship Dataset .....	92
5.6	Conclusion .....	93
Chapter 6: Conclusion and Future Perspective .....		94
6.1	Master Thesis Summary And Conclusions.....	94
6.2	Answering Research Questions .....	97
6.3	Related Scientific Publications .....	98
	References .....	99

## List of Figures

Figure 1. Training of a model in ML .....	19
Figure 2. Decision-making for test data.....	19
Figure 3. Scaling and blurring in SIFT .....	21
Figure 4. Keypoints detected by the ORB .....	22
Figure 5. SVM binary classification .....	24
Figure 6. Decision boundaries in k-NN algorithm.....	25
Figure 7. Matrices Illustrating L1 Distance Calculation for k-NN Classification.....	26
Figure 8. Artificial Neural Network (ANN) consists of three layers .....	27
Figure 9. General ANN architecture .....	28
Figure 10. General CNN architecture .....	29
Figure 11. General RNN architecture .....	29
Figure 12. CNN Illustration .....	30
Figure 13. Using the kernel to feature the input data.....	31
Figure 14. K filters with size $F * F$ redrawn from .....	32
Figure 15. Stride movement.....	32
Figure 16. AlexNet Architecture .....	34
Figure 17. ResNet Blocks .....	35
Figure 18. Architecture of ResNet34 .....	35
Figure 19. ResNet's Shortcut Connections for Information Flow .....	35
Figure 20. DenseNet block feature concatenation .....	36
Figure 21. DenseNet architecture with three Dense blocks and transition layers .....	37
Figure 22. Bagging.....	43
Figure 23. bagging technique, ensemble models are trained on bootstrap .....	44
Figure 24. Boosting technique .....	45
Figure 25. Sequential training of weak learners in Boosting with Weighted Voting .....	45
Figure 26. Stacking technique: Meta-model trained on Base Models' predictions .....	47
Figure 27. Gradient Boosting: Each new model predicts Residual Errors of previous model.....	49
Figure 28. Adaptive weighting in ensemble pruning for dense neural network .....	51
Figure 29. Model pruning by diverse.....	53



Figure 30. Illustration of Resizing Process .....	61
Figure 31. Python code of the normalization process.....	62
Figure 32. RGB image with its greyscale .....	63
Figure 33. Architecture Design of the first CNN model.....	65
Figure 34. Source Code of the first CNN model .....	66
Figure 35. Source Code of the second CNN model.....	67
Figure 36. Source Code of the third CNN model .....	68
Figure 37. Source Code of the fourth CNN model .....	69
Figure 38. Source Code of the fifth CNN model .....	70
Figure 39. Ensemble model Framework.....	71
Figure 40. Ensemble Pruning Framework .....	72
Figure 41. Knowledge Base Created for the Five Models.....	73
Figure 42. Knowledge base creation for the search of tree exploration .....	76
Figure 43. Example of search of tree exploration.....	77
Figure 44. Dataset illustration on different fish classes .....	81
Figure 45. Ship dataset.....	82
Figure 46. Netron Software (Roeder, n.d.) .....	83
Figure 47. Model visualization using Netron (from the left to right, Model-1, Model-2, Model-3, Model-4, Model-5).....	83
Figure 48. Illustration of test in MNIST dataset .....	87
Figure 49. Advanced models architecture, drawn by Netron (Roeder, n.d.) .....	91

## List Of Tables

Table 1. Accuracy performance with different shape sizes of jellyfish images .....	84
Table 2. Accuracy of models on jellyfish dataset with epoch 50 with 128x128 image size, and 132 batch size.....	85
Table 3. Accuracy of the VGG16 Model on the Jellyfish Dataset (Epoch: 50, Image Size: 224x224, Batch Size: 132) .....	85
Table 4. Accuracy Comparison of Five Models on the MNIST Dataset (Fixed Parameters: Epoch: 100, Batch Size: 32, Image Size: 28x28) .....	86
Table 5. Accuracy of the models with the ship dataset .....	88
Table 6. Comparison of advanced and old model accuracy on jellyfish dataset. ....	92
Table 7. Comparison of advanced and old model accuracy on ship dataset .....	93

## List Of Abbreviations

1. Two Dimensional (2D)
2. Artificial Intelligence (AI)
3. Artificial Neural Network (ANN)
4. Artificial Neural Networks (ANNs)
5. Convolutional Neural Network (CNN)
6. Convolutional Neural Networks (CNNs)
7. Bootstrap Aggregating (Bagging)
8. Deep Learning (DL)
9. Fully Connected Layer (FCL)
10. Fully Connected Layers (FCLs)
11. Global Average Pooling (GAP)
12. Machine Learning (ML)
13. Maritime Management Ensemble Learning System (MAGNAT)
14. Maritime Management Ensemble Pruning System (MAGNUT)
15. Rotated BRIEF (ORB)
16. Research Question 1 (RQ1)
17. Research Question 2 (RQ2)
18. Red, Green and Blue (RGB)
19. Scale-Invariant Feature Transform (SIFT)
20. Smart Maritime And Underwater Guardian (SMAUG)
21. Support Vector Machine (SVM)
22. SURF (Speeded-Up Robust Features)
23. Visible Maritime Image (VMI)

## Chapter 1: Introduction

The maritime industry usually sticks to its traditional ways. These days by using new digital tools such as Artificial Intelligence (AI) and Machine Learning (ML), this field has taken a big step to make itself more digitalized. AI is changing how the ship works and it shows why it is important (Chin & Venkateshkumar, 2022).

Despite big advances, there is still a big gap in fully using AI and machine learning in many maritime tasks. With most of our planet covered by water and most goods transported by sea, it is crucial to come up with new tech solutions. For example, as self-operating technology gets better, projects like self-driving ferries show the urgent need for systems that can understand sea environments well, where deep learning can be very helpful.

AI is important for managing the complex issues of eco-friendly shipping and overall digital changes in the shipping industry. However, challenges remain, like the need for skilled data experts and the intense computer work needed for tasks such as data labelling in machine learning. It is also important that everyone can see how these AI systems make predictions, which calls for a team skilled in both tech and legal issues.

Deep learning, a key development in machine learning, is crucial for creating smart maritime systems. It is used in many parts of maritime science and tech, like detecting objects and avoiding collisions. Deep learning uses remote sensing, which is very important for keeping an eye on the environment, securing marine areas, and for business (Chin & Venkateshkumar, 2022; LI et al., 2021). But, the variety in ship designs and environmental conditions brings many challenges.

Ensemble learning, which utilizes multiple methods together to enhance accuracy, seems like an interesting solution for overcoming these issues. By combining strengths and reducing weaknesses of various methods, ensemble learning tries to work well with different types of data. However, these methods can be demanding in terms of time and memory.

This study focuses on ensemble pruning, a technique to make multiple models work better together by removing fewer effective parts. This helps make better decisions and use resources more efficiently in managing maritime activities. This research aims to show how AI and machine

learning, especially deep learning and ensemble methods, are changing the shipping industry. It looks to address technical problems and find new ways to improve the efficiency, accuracy, and flexibility of maritime operations and technologies. The findings could also be useful in other academic areas and industries.

### **1.1. Research Background And Motivation**

Across vast oceans, the maritime industry relies on ships as primary transportation tools. For strategic decision-making and real-time battlefield awareness in various operational environments, Ship detection and classification are crucial. Advancements in remote sensing imaging technology have led to increased interest in marine target detection and classification, making monitoring the wide sea surface feasible. In both military and civilian domains, these techniques hold significant application value. They are used for supervising marine resources, military patrols, and territorial rights protection. In observing dynamic surface changes, optical remote sensing images play a vital role. They are essential for future research and development in ship detection and classification (LI et al., 2021).

AI research has changed. It used to copy human decisions. Now, it does things that were thought impossible before. Since 2012, people have been studying big data and AI more. Especially for autonomous ships, they are using AI to change how maritime operations work (Munim et al., 2020). Also, a transition towards innovative solutions in maritime transportation is because of the rise of autonomous ships, ferries, and vehicles. This transition is facilitated by improvement in sensing technologies such as lidars, radars, and cameras (Viken Grini, 2019).

One of the main reasons for the improvements in image classification and object detection in recent years is the advancement of deep learning algorithms. This improvement has made more people interested in using data from a distance to classify ships. This is important for monitoring the environment, ensuring maritime security, and for commercial purposes (Chávez et al., 2024; Chin & Venkateshkumar, 2022). However, ships come in many different types, which can be challenging. Ensemble learning helps by combining many classifiers to improve accuracy and performance. Even though ensemble learning requires a lot of computational demands, it can really improve decision-making and operational efficiency in maritime management (Yan et al., 2022),(Wang et al., 2021),(Salem et al., 2023).

Using an ensemble approach provides a potential solution to exceed the predictive capabilities of any single model. This is done by using the combined strengths of various methods. This strategy enhances the stability and adaptability of the ensemble to different data types. It balances out individual weaknesses of the methods involved (Yan et al., 2022),(Wang et al., 2021),(Salem et al., 2023). Each method within the ensemble uses a unique strategy. This contributes to a robust and reliable predictive model. The computational complexity of ensemble classifiers increases linearly with the number of base models. This means that the cost of computation goes up as the number of models used goes up. Choosing the right base models is really important. It affects how complicated the computations are. So, ensemble learning has two main challenges. These challenges are explained in the research question. This research ultimately aims to optimize model selection in maritime management. This will make decisions better and work faster in a field where making good decisions is very important.

## **1.2. Research Questions**

Research Question 1 (RQ1)

*How can demanding resource requirements, leading to substantial computational time and memory consumption in loading and executing individual models during the inference phase of ensemble learning, be effectively addressed?*

Research Question 2 (RQ2)

*How can ensemble learning models effectively address the challenge of potential detrimental effects caused by specific models within the ensemble, ensuring cooperative dynamics enhance rather than obstruct overall learning outcomes and performance improvements in their intended applications?*

## **1.3. Research Purpose**

The author's purpose is to explore ensemble pruning to remove the models that are not necessary in the group. The author is interested in designing an intelligent solution to make multiple models work together better in the ensemble. This is done by selectively pruning those components that contribute less effectively to the overall task. This ensemble pruning methodology works to make the decision-making process more efficient. It also helps improve

resource allocation in maritime management scenarios. This new model combines the strengths of ensemble learning with a strategic pruning mechanism. It finds and keeps the most important parts. It also gets rid of or reduces the effect of less important ones. The model seeks to achieve a higher level of precision by this action. It also works to achieve greater efficiency and adaptability in dealing with the complexities of maritime management tasks. This innovation represents a significant advancement in the field of maritime management. It provides a better and faster way to make decisions and improve operations in maritime settings.

#### **1.4. Research Scope**

This study is in support of the SMAUG project (<https://smaug-horizon.eu/>) and the DARWIN group (<https://www.norce-research.no/en/research-group/darwin>) at NORCE. SMAUG is a project funded by Horizon Europe that focuses on maritime safety and security. It comprises several work packages. The author works as a student research assistant at the University of Southeastern Norway (USN), primarily involved in work package 5. The goal is to develop an intelligent solution based on advanced deep-learning architectures for detecting abnormal objects in the sea. The author will utilize a novel framework explained in this study within the SMAUG project, employing ensemble pruning techniques to enhance model efficiency and performance.

#### **1.5. Research Structure and Contents**

The next chapter's structure is as follows: Chapters 2 and 3 explore the current state of research in ML and DL with a focus on ensemble learning and pruning. They highlight the research need and identify gaps through a literature review. Chapter 4 details the MAGNAT and MAGNUT solutions proposed in this master's thesis, which are among the main contributions. Chapter 5 presents experimental results on the proposed solutions using various maritime datasets, including fish as an example of marine and maritime science and ships as examples of maritime technology. Chapter 6 summarizes the findings and proposes future research directions, as well as presenting the scientific publication of the author related to this thesis.

The following chapters' contents include: Firstly, each chapter provides a chapter conclusion to offer a brief overview and summary, as well as an introduction at the beginning of each chapter. Secondly, all figures, tables, and equations in this thesis indicate their sources, whether directly or indirectly used, using terms such as "taken from," "redrawn from," and

“extracted from,” in adherence to authorship rules. Lastly, the Python programming codes used for generating results and evaluations in the paper are not mentioned in this report, as per the expectations of the supervisor and project requirements.



## **Chapter 2: Literature Review (PART I), Machine Learning For Image Classification**

The chapter first introduces the main concepts focused on ML and advanced ML which is called DL for image classification. It starts with the main difference between them. Then the ML approaches from scratch by dividing them into two main steps, feature extraction and classifiers are reviewed. For each of the steps, popular algorithms are explained. After ML, Special analysis has been done on DL. Convolutional Neural Networks (CNNs) which are the main objective of the paper related to Image Classification are summarized in detail. Then popular architectures that are based on CNN are described. Some research papers that have been conducted on the classification of ships with different algorithms are conducted. The last part of the chapter is a summary.

### **2.1. Introduction**

The AI study in the scope of maritime management systems has become an important topic for several research studies. Notably, the application of DL, a subset of AI, has increased its position as a means of addressing many challenges within the context of the marine area. In this chapter, we will discover significant studies on ML and deep DL which are the subsets of AI, with specific attention on image classification. Image classification is a key task in computer vision where images are arranged into predefined categories. It places the source for other tasks similar to localization, detection, and classifier in computer vision (Li et al., 2015). While humans understand this task quite upfront, it creates substantial challenges for automated systems. By studying previous ML algorithms and considering their challenges, we develop an understanding of the signs of progress in the DL domain, particularly as they are used for image recognition. This review goal is to put together studies that demonstrate the growth of these technologies in image classification, presenting how learning from previous algorithmic challenges has directed an important advancement in this particular domain and how researchers used a subset of AI in image recognition, particularly in marine applications.

## **2.2. Difference Between Conventional Machine Learning And Deep Learning Approach**

Conventional machine learning and DL methods are two famous subfields of AI with different methodologies and capabilities.

In conventional machine learning, human specialists are accountable for choosing and curating the features of data that the machine studies. This process needs human involvement to recognize the related pieces of data for training the machine (Munoz, 2017). On the contrary, in DL, the computer autonomously distinguishes which parts of the data are important for learning, removing the necessity for understandable feature selection by humans (L.Hosch, 2023). This autonomous feature extraction ability builds DL mostly proficient at handling complicated and massive datasets (L.Hosch, 2023).

DL is a sophisticated ML method that empowers machines to automatically extract understandable insights from raw data. In contrast to traditional ML methods, DL frequently provides greater outcomes due to its capability to capture complex patterns and relationships in data (Setiowati et al., 2017). The essential tool in DL is the deep neural network, which contains numerous layers capable of learning complicated mappings between input and output data (LeCun et al., 2015).

Both conventional machine learning and DL work with supervised and unsupervised learning techniques. In supervised learning, the machine studies from labelled training data to forecast results or answer certain problems (Setiowati et al., 2017). On the contrary, in unsupervised learning, the machine must distinguish patterns or structures within unlabelled data without the direction of predefined labels (Chauhan & Singh, 2018a; Setiowati et al., 2017).

The difference between conventional machine learning and DL methods falls into one of the famous benefits of DL: its capability to avoid the necessity for manual feature engineering, a labour-intensive task common in conventional machine learning. Unlike conventional machine learning, DL models can autonomously recognize related features from raw data, process them, and make decisions based on learned illustrations (LeCun et al., 2015; M. Geetha & Neena Aloysius, 2017, p. 7). Also, DL achieves fine when handling complicated and massive data (L.Hosch, 2023).

## 2.3. Conventional Machine Learning

ML, as described by Arthur L (1959) and extended by Chauhan and Singh (2018), is a domain of AI that works without explicit programming (Chauhan & Singh, 2018a; L. Samuel, 1959). It is based on the computational concept of learning within AI, creating the groundwork for its methodologies. Within ML, algorithms are trained to utilize raw data. These algorithms employ training utilizing the provided data, empowering them to create predictions based on the learned patterns and relationships, as mentioned by Kohavi (1998) and more emphasized by Chauhan and Singh (Chauhan & Singh, 2018b; Kohavi, 1998).

Conventional machine learning algorithms conform to a process where they are trained utilizing existing data, resulting in the making of a trained model, shown in Figure 1. This model, having been pre-trained, is then used to distinguish and classify test data, as shown in Figure 2 (Chauhan & Singh, 2018c).

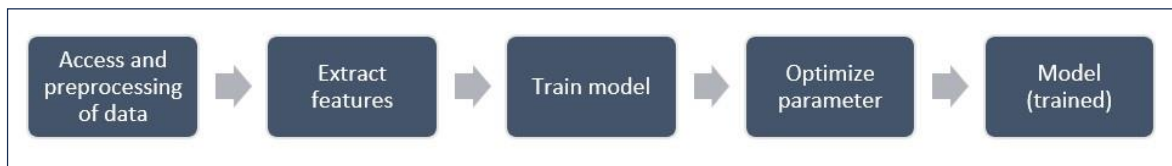


Figure 1. Training of a model in ML redrawn from (Chauhan & Singh, 2018c)

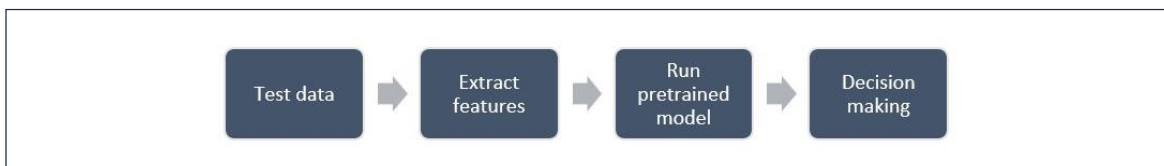


Figure 2. Decision-making for test data redrawn from (Chauhan & Singh, 2018c)

In the next, we will revisit some of the ML algorithms included with two steps: “Feature Extraction,” and “Classifier.”

### 2.3.1. Feature Extraction

Feature extraction is a computing process that analyses image data to recognize particular patterns, edges, textures, or additional characteristics that are valuable at each point in the image (Subasi, 2019). This process is critical in computer vision, with its importance increasing across numerous fields. An operative feature detection method should be qualified to deal with image transformations such as rotation, scaling, changes in lighting, noise, and affine transformations

(Karami et al., 2017). In the next, we will discover the three famous methods for feature extraction from images: “Scale-Invariant Feature Transform (SIFT),” “Oriented FAST and Rotated BRIEF (ORB),” and “Speeded-Up Robust Features (SURF).”

#### **a. Scale-Invariant Feature Transform**

The SIFT algorithm is one of the solid computer vision techniques. This algorithm is used for detecting and describing features within images. It works by identifying the main points in an image that leftovers specific despite changes in scale, rotation, or transformation (Lorencin et al., 2021). These main points are detected based on their brightness and are categorized by brief descriptors, including information about nearby image details. This competence empowers several applications such as image matching, object recognition, and image retrieval.

In computer vision, a mutual problem is recognizing objects in images that exhibit rotation or are taken from diverse angles and scales. The SIFT algorithm relates to the challenges by pinpointing precise points, identified as “key points,” within an image. These key points hold their individuality even under resizing or rotation, making them priceless for a multitude of computer vision responsibilities. SIFT features help as descriptors during model training with images. A substantial benefit of SIFT features, compared to others, is their consistency across diverse image sizes and orientations (Lowe, 2004).

To demonstrate the SIFT process, consider a set of images showing an object from numerous angles and sizes, each image blurred to different extents utilizing Gaussian blur. Figure 3 shows the four-time scales of the image, and for each scale, the image has to be blurred five times. These blurred images are then subtracted, resulting in a stack of images where extreme points, highlighting of key points, and stand out. This process is presented not only for a single image resolution but also for an image pyramid, containing of images scaled dissimilarly to recognize key points across various scales.

The subsequent step includes making descriptors by analyzing the local neighborhoods of the key points. Gradients within these neighborhoods are computed, building robustness against rich changes and individual viewpoint modifications. By aggregating these gradients into histograms and perceiving their incidence and magnitude within local regions, a descriptor vector is shaped around each key point in a four-by-four region. This descriptor vector successfully

categorizes the local key points. However, one substantial downside of the SIFT algorithm is its high computational cost, which requires considerable computing power and resources to function efficiently. Accordingly, researchers are continuously discovering other methods that propose similar performance while demanding less computational resources (Bay et al., 2008).

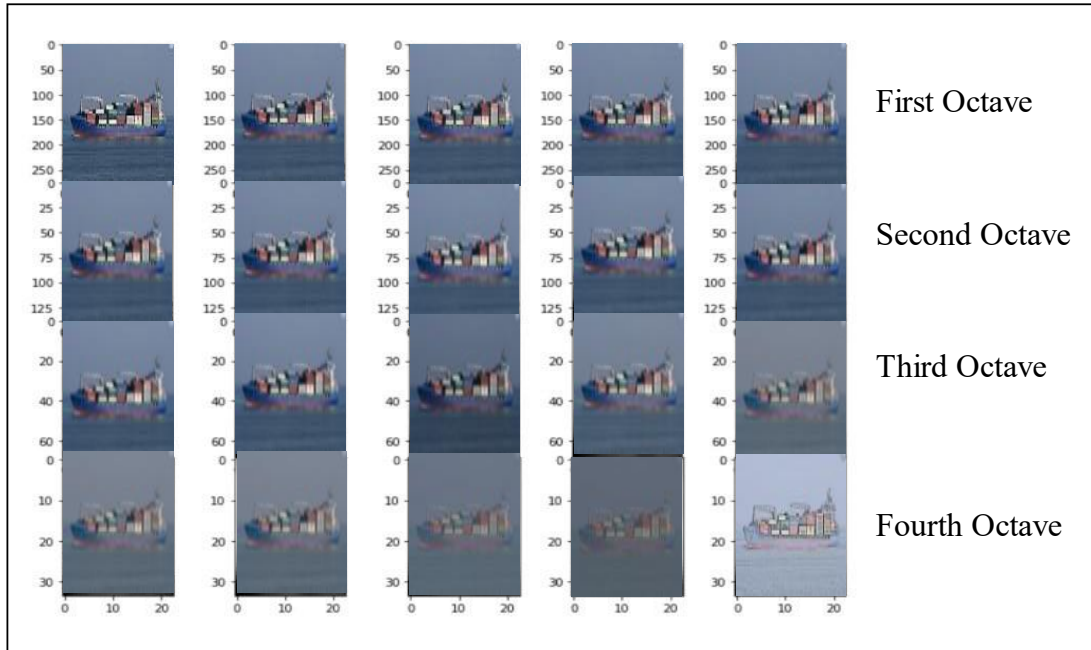


Figure 3. Scaling and blurring in SIFT extracted idea from (Lowe, 2004)

#### b. Oriented FAST And Rotated BRIEF

The Oriented FAST and ORB algorithm established in the OpenCV labs. This algorithm offers as an effective option to SIFT, offering comparable feature detection while surpassing it in speed and robustness to noise. Also, ORB presents a cost-effective solution, particularly considering patent considerations, serving as a viable substitute for both SIFT and SURF (Rublee et al., 2011).

Combining the FAST key point detector and BRIEF descriptor, ORB incorporates several enhancements to enhance its performance. Initially, it employs the FAST algorithm to identify key points. FAST evaluates the brightness of a given pixel “p” in comparison to its surrounding pixels arranged in a small circle, categorizing them as lighter, darker, or similar to “p.” If more than 9 pixels fall into the darker or brighter categories, “p” is selected as a key point. Figure 4 shows an example of some key points that the ORB detects. ORB also adopts a pyramid structure

to generate multi-scale features, where each level in the pyramid represents the image at different resolutions. After constructing the pyramid, ORB utilizes the FAST algorithm to detect key points at each level, enabling the identification of key points at various scales and imparting partial scale invariance to the algorithm. However, a limitation arises as FAST does not compute orientation. To address this issue, ORB utilizes the intensity centroid to detect intensity changes. This approach assumes that the intensity of a corner is offset from its center, and the resultant vector is employed to infer the orientation of key points (Rublee et al., 2011).



Figure 4. Keypoints detected by the ORB extracted idea from (Rublee et al., 2011)

### c. Speeded-Up Robust Features

SURF proposes a feature extraction algorithm in computer vision and image processing. This algorithm suggests an alternative option to methods like SIFT in terms of speed and repeatability of detector and descriptor. Advanced for real-time applications and resource-constrained situations, SURF reaches computational efficiency through the operation of integral images and box filters. Similar to SIFT, SURF keeps scale and rotation invariance. SURF empowers it to recognize and match features despite variations in object size or orientation. SURF uses a descriptor that mixes information about image intensity and gradient orientation distribution. This descriptor, more computationally resourceful than SIFT descriptors, collaborates with the whole effectiveness of the algorithm. In the detection of interest points,

SURF works with the Hessian matrix. SURF operates the identification of regions with substantial intensity variations that possibly correspond to distinguished features. Furthermore, the practice of SURF may be focused on patents, and compliance with licensing attention might be needed for commercial applications. The computer vision area is frequently growing, and there may be fresher advances or replacements in feature extraction algorithms (Bay et al., 2008).

### **2.3.2. Classifier**

Classifiers are algorithms that works on assigning labels or categories to input data based on the patterns they study from the training data. In the subsequent discussion, we will discover three of the most popular methods for classifiers used with images: “Support Vector Machine (SVM),” “k-Nearest Neighbors (k-NN),” and “Artificial Neural Networks (ANNs).”

#### **a. Support Vector Machine**

SVM is considered a supervised learning system used for both regression and classification responsibilities (Cortes & Vapnik, 1995, pp. 273–297). Renowned for its ability to provide clear boundaries between different groups, SVM was originally designed for binary classifications, separating data into two groups. Over time, scientists developed SVMs to incorporate further than two groups, a methodology acknowledged as multiclass classification (Crammer & Singer, 2001). An operative approach for multiclass challenges contains building a set of binary classifiers, each understanding between one label and the remaining classes (Y.-C. Wu et al., 2008).

As shown in Figure 5, the SVM algorithm is described in a graphical format, where each point describes a set of features cast off for classifying into two separate groups. Those groups are labeled as Class-1 (blue) and Class-2 (red). The figure illustrates an “Optimal Hyper Plane”—the green line, which successfully separates the two classes through the training phase. The margin is specified by the space between the hyperplane and the nearest data points from both classes. The margin plays a vital role in SVM’s training by confirming the maximum distance between the plane and the data points, so minimizing classification errors. Through the test phase, this optimal hyperplane is utilized to categorize new data points. That demonstrates the robustness of SVM in dealing with binary classifications.

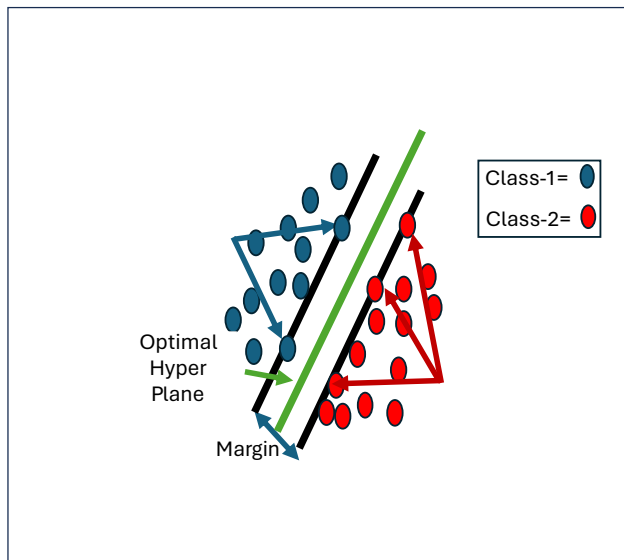


Figure 5. SVM binary classification extracted idea from (Cortes & Vapnik, 1995, pp. 273–297)

### b. k-Nearest Neighbors

k-Nearest Neighbors (k-NN) is an upfront algorithm classified under supervised ML. It requires learning from a labeled training dataset, where both the input data (X) and their corresponding labels (Y) are visible. Through this circumstance, the algorithm studies to associate input data with their respective demanded outputs (Y). In this model, the whole trained dataset works as a critical component, as the model understands the nuances of the training data. Through prediction, the algorithm outputs a class based on the majority among the "K" nearest neighbors. Once the model has assimilated the training dataset for prediction, it starts by calculating the distance between the test image and all the images kept in the training dataset. This distance measurement helps in finding the training images that closely resemble the test image (Damastuti et al., 2019).

Based on the distance calculation, the model chooses the “K” training images that closely resemble the test image, and the most similar one is selected for prediction. Each of the nearby data points has a class label associated with it. As an example, if the classifying image is a ship, each of these “K” neighbors might have labels such as “Cargo Ship,” “Carrier Ship,” or “Tanker Ship” based on their illustration. Subsequently, the model counts the frequency of each class label among the “K” neighbors, and the label with the uppermost frequency is assumed as the prediction for the test image (M. Zhang et al., n.d.). **Error! Reference source not found.** indicates a 3-class c



lassification applying the K-NN algorithm with the  $k = 15$  and uniform weights. Each point of the plot defines a data point in the dataset, colored points concerning to their class. For example, red points are considered into one category, blue points including to another category, and green points are into the third category. The background color in the plot area is shaded presenting to the k-NN prediction for that region. This offers how new data points would be classified based on their location. The borders between these colors show where the k-NN algorithm changes its prediction from one class to another proving the decision boundaries created based on the closeness of the 15 nearest neighbors.

The distance between the test data point and the trained data point is named the distance metric. This distance calculation allows k-NN to specify which training data points are in nearness to the test data point (M. Zhang et al., n.d.). The election of distance metric is critical. This election may affect the performance of the k-NN algorithm and how it clusters data points into "neighborhoods" (Bir, 2019; Cunningham & Delany, 2021; Z. Zhang, 2016). Figure 7 demonstrates how the L1 distance metric is utilized within the k-NN algorithm. The figure displays three matrices introducing diverse points in a two-dimensional space. The figure shows each matrix demonstrating values that correspond to features of distinct data points. The L1 distance between these points is measured by adding the differences of their corresponding features. For example, to measure the L1 distance between the first and second data points, the absolute differences between corresponding entries of the matrices are computed and added. In this figure, the goal is to visualize how closer feature values specify closeness between points in feature space, which k-NN uses to find the “neighborhood” of a data point for classification.

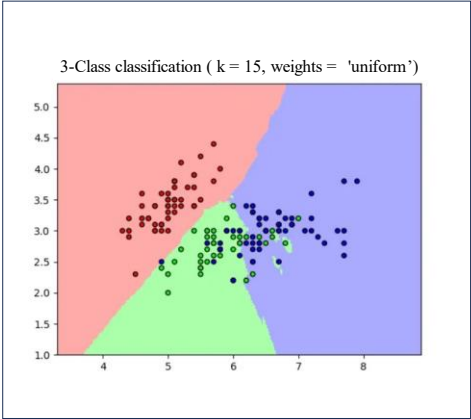


Figure 6. Decision boundaries in k-NN algorithm taken form (Bir, 2019)

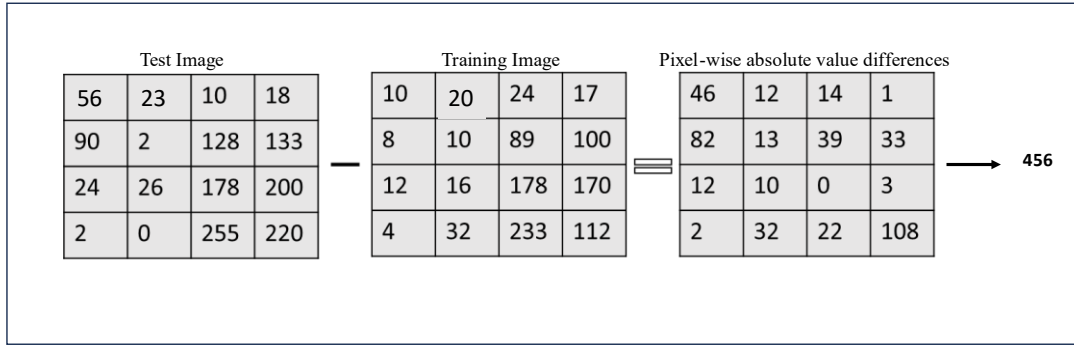


Figure 7. Matrices Illustrating L1 Distance Calculation for k-NN Classification taken from (H. Zhang & Fan, 2020)

### c. Artificial Neural Networks

ANNs introduce another family of ML models broadly applied in image classification. That might often be considered under the umbrella of DL technologies. Modelled after the operations of the human brain, ANNs encompass three main sets of units or layers. Those are input, hidden, and output, as shown in Figure 8. In this model, the input layer is located at the highest of the network to obtain incoming data. Although the output layer exists at the top level. That layer generates results such as probability distributions including all possible classes. The hidden layer shows between the input and output layers, where information is processed through interconnected layers of nodes or neurons, as depicted in Figure 9. Training ANNs usually contain backpropagation, a technique that regulates the connection weights between neurons to diminish a loss function, calculating the inequality between predicted and accurate outputs. The backpropagation algorithm computes gradients of the loss function based on the weights and uses them to update the weights. Though, ANNs face troubles in image classification tasks. One important challenge is that ANNs assume each pixel as an independent feature, possibly overlooking spatial associations between pixels and regions within an image. Accordingly, ANNs may struggle with identifying complicated patterns or objects. As they may fail to recognise the interconnection of several parts considering to the same object. To fall into these restrictions, CNNs have arisen as improvements in the field of neural networks and ML, as depicted in Figure 10. CNNs incorporate particular layers and mechanisms, such as convolutional layers, qualifying them to capture spatial patterns crucial for image classification (Qamar & Zardari, 2023; P. Sharma, 2023).

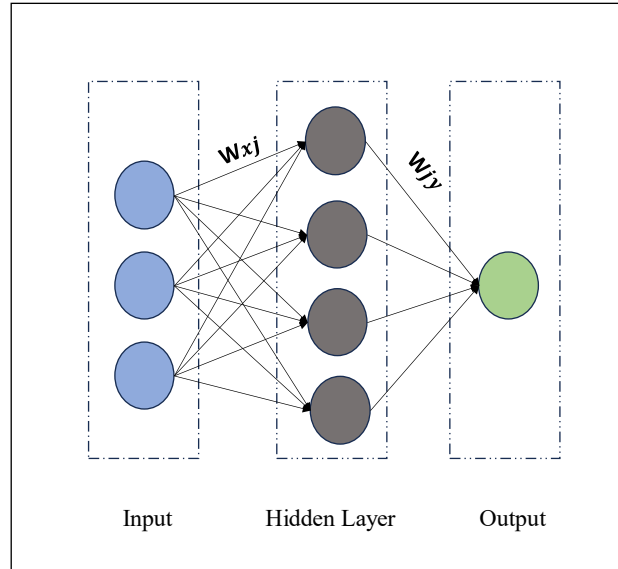


Figure 8. Artificial Neural Network (ANN) consists of three layers extracted idea from (Shiruru, 2016)

## 2.4. Deep Learning

DL has lately increased consideration in the field of AI. One of the key downsides of conventional machine learning is a struggle with the selectivity invariance issue. It is hard to figure out which features in the data encompass respected information and which ones are less significant. Nominated data should be well-defined from each other, and traditional ML methods occasionally struggle with it. This trouble inspires scientists to discover DL as an advancement in ML. DL is branded as representation learning. DL is considered with various layers. Utilizing nonlinear models that convert the raw data into higher-level abstractions for the process of creating decisions. DL can make the process of finding the solution easier for complicated and non-linear functions (Ramachandran et al., 2015). This model efforts on feature learning automatically which proposes modularity and transfer learning capability. As the name presents DL regularly includes architectures with deep layers. Opposite to conventional machine learning, DL wants a huge volume of data to efficiently train a neural network.

The most famous DL is the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) (Sherstinsky, 2020). CNN is usually used for imagery classification, while RNN is used for time-variant difficulties such as speed recognition as depicted in Figure 11 (Hoffmann et al., 2017). As CNN is used for image classification, it will be studied in the upcoming section. CNN model can also be considered for object recognition, text recognition, object detection

(Jaderberg et al., 2014), sense labelling (Farabet et al., 2012), and numerous other applications (Nithin & Sivakumar, 2015). This model is comparable to ANNs. In both CNNs and ANNs, the hidden layers of neurons are mainly connected to the previous layers. However, CNNs are mostly considered and used for tasks associated with pattern recognition within images. They shine at capturing image-specific features. On the other hand, ANNs are more general-purpose and not inherently focused on image-focused tasks. They can be useful for a broader range of data types and tasks (O’Shea & Nash, 2015). It provides an opportunity to involve learning of features. Each part of the CNN layer encompasses two or more dimensional Filters which are convoluted along with the input of the layer. Deep Convolutional Networks are suited to finding both simple and complicated data patterns (Munoz, 2014). These layers cooperate to learn from input data operating CNNs more efficiently in tasks such as Image classification. Figure 12 presents an instance of the CNN sequence to classify a ship, which starts with an image as input. This image is proposed through diverse layers. As indicated in Figure 12, those layers include “a convolutional layer,” “a pooling layer,” and “a fully connected layer (FCL).” Also, the output of this system would be a label.

In the upcoming subsections, we will explain the components of CNNs in more detail. Those include the “Type of Layer,” “Filter Hyperparameters,” and “Dimensions of a Filter,” “Stride,” “Zero-padding,” and “Architectural Design in CNN.”

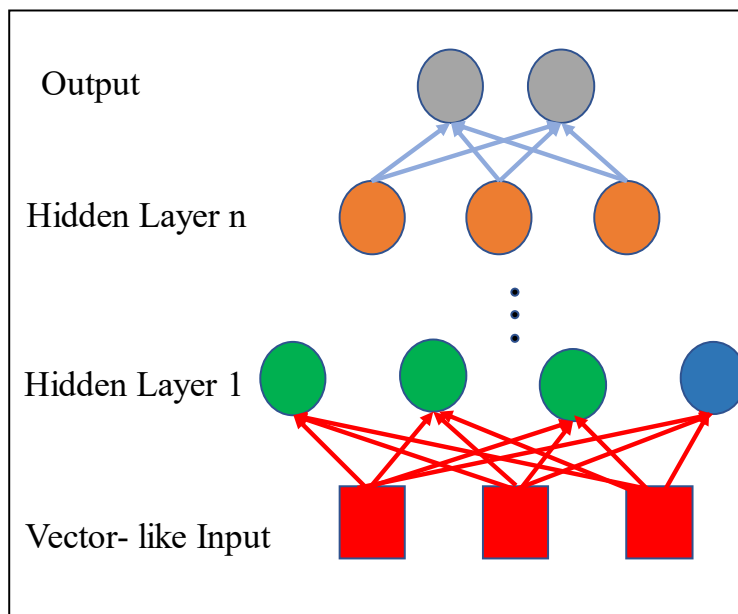


Figure 9. General ANN architecture taken from (Khruschev et al., 2022)

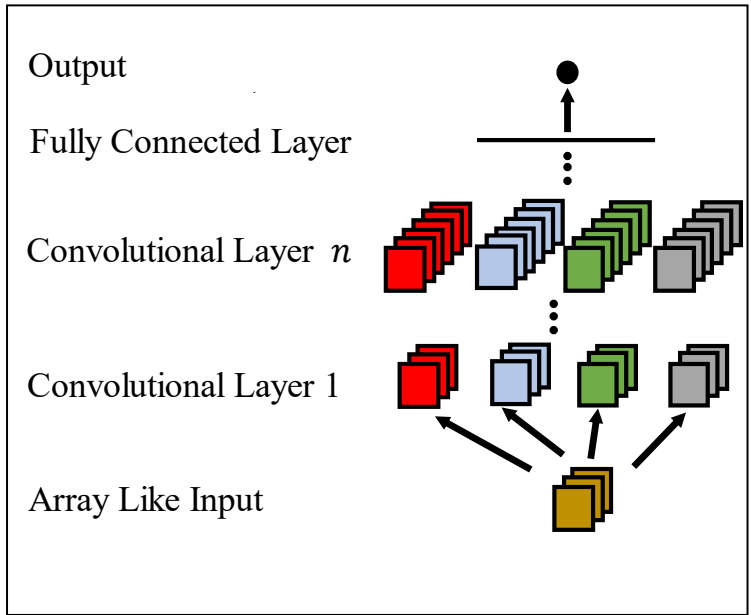


Figure 10. General CNN architecture taken from (Khrushev et al., 2022)

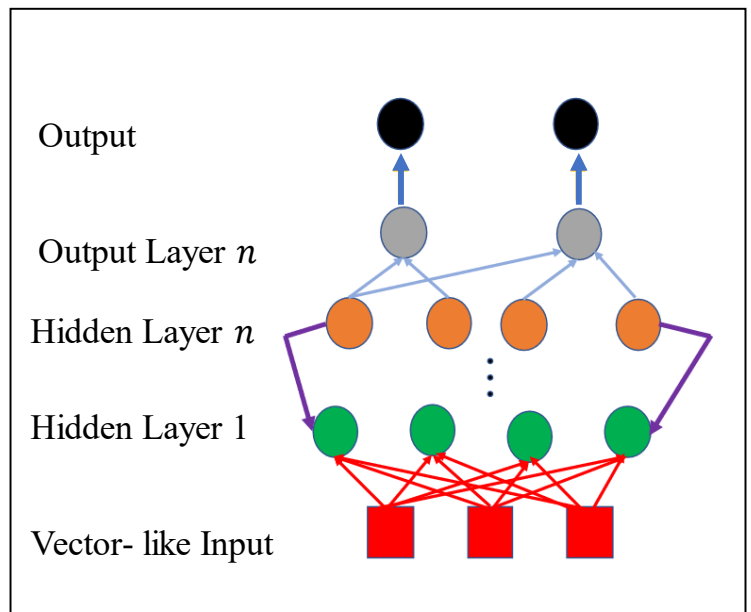


Figure 11. General RNN architecture taken from (Khrushev et al., 2022)

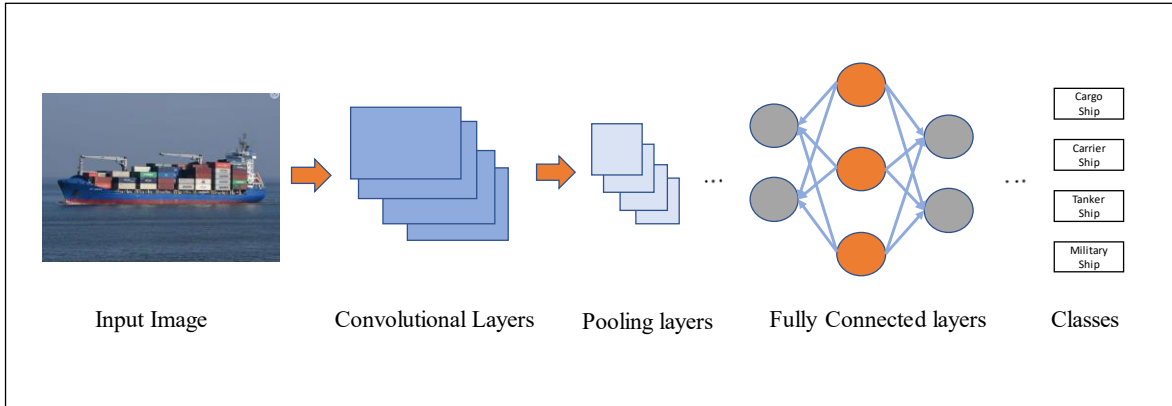


Figure 12. CNN Illustration redrawn from (Reimers & Requena-Mesa, 2020)

### 2.4.1. Type Of Layer:

CNN contains diverse layers (Shi et al., 2019). Below, the three main layers are described: “Convolutional Layer, “Pooling Layer,” and “Fully Connected Layer.”

#### a. Convolutional Layer

The Convolutional Layer is considered one of the key components within the CNN. This layer performs the majority of computations. These layers, along with kernels (filters), perform convolutions with input features. This results in the generation of a clear two-dimensional (2D) activation map. This process provides a reduced number of features, gained through weight sharing among neurons. The result supports the simplification of the network’s complexity. (Fronzetti, 2019; Hinton et al., 2012). Training the CNN includes adjusting the weights of the filters within the convolutional layer (O’Shea & Nash, 2015).

#### b. Pooling Layer

The Pooling Layer is one of the important components in CNNs. These layers are frequently organised in a sequence along with convolutional layers. Pooling layers contribute to reducing the spatial dimensions of image activation maps and simplifying the identification of features without losing information. This results in an overall reduction of computational complexity (Ren et al., 2019). Additionally, this layer addresses overfitting concerns and issues. Notably, there exist numerous pooling operations. Average pooling and max pooling are among the most common (Gallego et al., 2018). Maximum pooling returns the maximum value from the

portion of the image covered by the kernel. However, the average pooling returns the average of all the values.

**c. Fully Connected Layer**

The FCL in CNN is a layer that every neuron in the layer is linked to every neuron in the previous layer as depicted in Figure 12. It finds the output from the previous layers and operates them to classify the input into different classes based on the training dataset. These layers work on the flattened output from previous layers. Each input feature is connected to all neurons, similar to traditional neural networks. This operation takes complex patterns and simplifies them. These outputs help determine a pattern’s possibility of fitting into a precise category (Basha et al., 2020).

**2.4.2. Filter Hyperparameters**

Filters are small matrixes that operate in extracting features from input data. Filters have associated hyperparameters. These hyperparameters determine their behavior during the training process. Filters are employed in the convolutional layer to illustrate how the filter can be applied to the convolutional layer, as shown in Figure 13 (Hristov et al., 2019; O’Shea & Nash, 2015). The process of applying a convolutional kernel with a size of (3,3) to input data to generate convolved features is also depicted in Figure 13. By the corresponding numbers in the kernel matrix, the input pixels in the matrix are multiplied. These products are accumulated together. This produces a single output value for each position the kernel covers in the input matrix. As the kernel slides over the entire input, this operation is repeated. This processes a feature map.

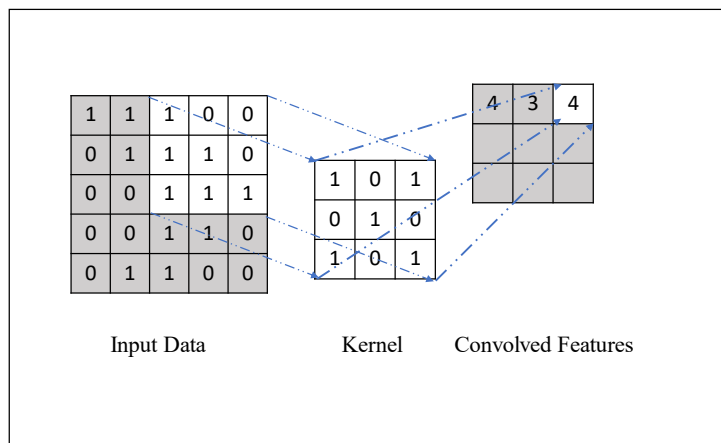


Figure 13. Using the kernel to feature the input data redrawn from (Panchal et al., 2023)

### 2.4.3. Dimensions of a Filter

The dimensions of each filter in a CNN are “ $F \times F$ .” They are designed to process input with  $C$  channels. An output volume with the dimensions “ $F \times F \times C$ ” is generated when a filter of this size is applied to the input volume. The depicted filters, denoted as “Filter 1,” “Filter 2,” through “Filter K,” each conduct convolution operations. This is done on an input of size “ $I \times I \times C$ ,” as shown in Figure 14. The result of these operations from all the filters forms an output feature map. It is commonly known as an activation map. The dimensions of this activation map are “ $O \times O \times K$ .” Here, “ $K$ ” represents the number of filters applied (Sert, 2020).

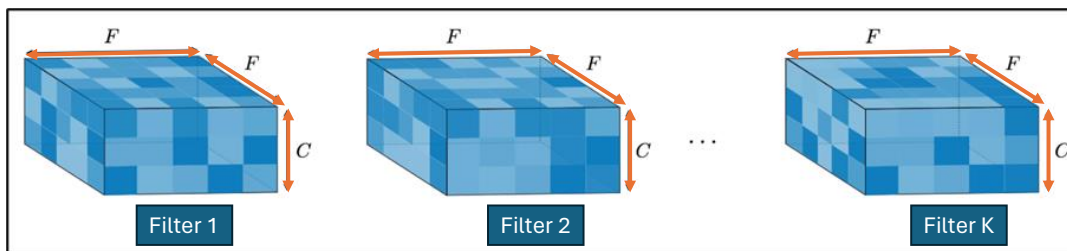


Figure 14.  $K$  filters with size  $F * F$  redrawn from (Sert, 2020)

### 2.4.4. Stride

CNNs propose several options to enhance performance and alleviate side effects. One such option is the adjustment of the “stride” parameter (Albawi et al., 2017). Stride is symbolised as “ $S$ ” in Figure 15. This determines the number of pixels the convolutional or pooling window moves after each operation (O’Shea & Nash, 2015). With this development governed by the stride setting, it improves pixel by pixel while applying a filter to an input (Zaniolo & Marques, 2020). Stride values can differ. That suggests flexibility in controlling the movement of the filter. Figure 15 determines the result of setting the stride to 2. It showcases how the array shifts towards the right side.

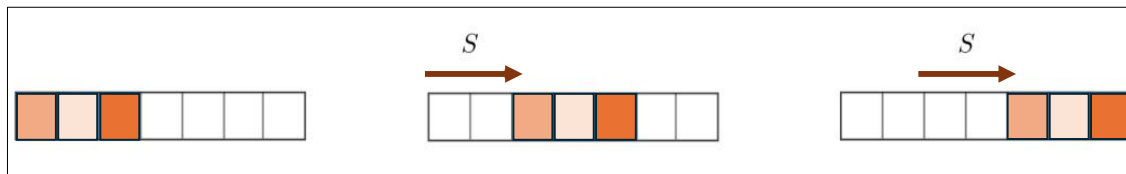


Figure 15. Stride movement redrawn from (Sert, 2020)



### **2.4.5. Zero-Padding**

The potential loss of information located at the edges of the image is considered one of the disadvantages of the convolution step. Such information is frequently lost due to it being only captured as the filter slides across the image. A straightforward approach is to employ zero-padding to indicate this issue efficiently. Zero-padding raises the process of adding extra rows and columns of zeros to the edge of the input to extend the input size. The original content remains unchanged. This technique is mutual in CNNs. The input image size has to be matched with the filter size. (Albawi et al., 2017; Nguyen et al., 2019).

### **2.4.6. Architectural Design In CNN**

For resolving the image classification issues, CNN-based architectures are important. Alongside discussing CNN components, three popular architectures in this domain will be discussed below: “AlexNet,” “ResNet,” and “DenseNet.”

#### **a. AlexNet**

AlexNet is one of the CNNs architectures. In 2012 in the ImageNet challenge, this architecture gained the best performance. This architecture can learn features from data automatically. For computer vision tasks, it does not involve designing features manually. The other difference is that AlexNet is much deeper than LeNet. LeNet is a pioneering 7-level convolutional network. It cannot process higher-resolution images due to the low number of layers (Kuo, 2016). AlexNet architecture consists of five convolutional layers, as shown in Figure 16. It also includes three max-pooling layers, two normalization layers, two fully connected layers (FCLs), and one softmax layer. AlexNet used ReLU, which is an activation function, for each of these layers excluding the output layer. ReLU aims to speed up the training process by almost six times. The dropout model within this architecture aims to prevent the overfitting of the architecture. In addition, the ImageNet dataset is used to train the model. This dataset contains almost 14 million images spanning a thousand classes (Anwar, 2022; Leonidas & Jie, 2021; Saxena, 2021; Varshney, 2020).

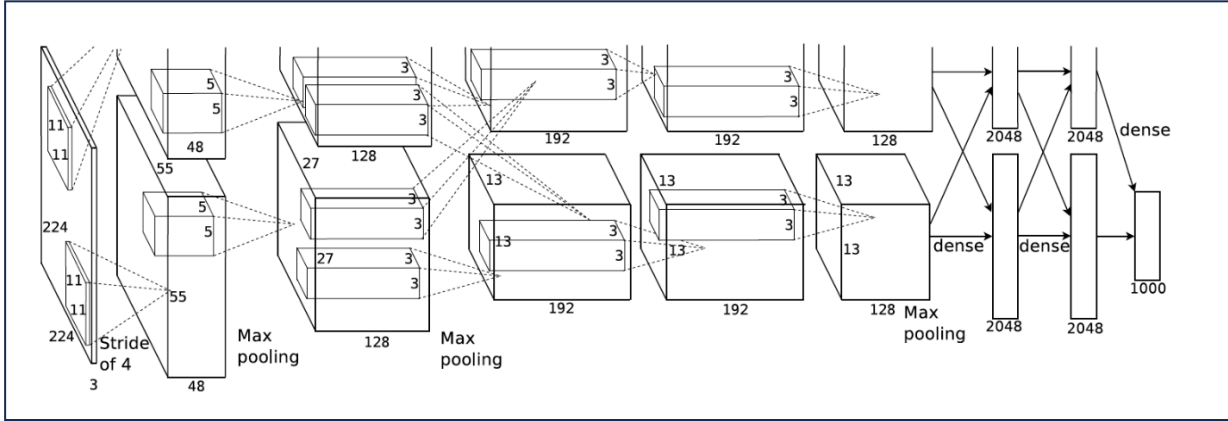


Figure 16. AlexNet Architecture taken from (Varshney, 2020)

## b. ResNet

In 2015, a Residual Network, or ResNet architecture, was proposed by researchers at Microsoft Research introduced. Deep neural networks, when more layers are added, face the Vanishing/Exploding gradient issue. This results in increased error rates. ResNet fights this problem with skip connections. These links connect one layer's activations to later layers, skipping some layers in between, as shown in Figure 19. As shown in Figure 17, this method makes residual blocks. ResNets are constructed by tapping these blocks on top of each other. Skip connections help training by letting the network learn residual mappings instead of underlying mappings as layers increase. This avoids gradient-related issues. ResNet-34 has 34 layers, and this architecture was outlined in the paper (He et al., 2016). Its structure includes various types of layers, as displayed in Figure 18. Initial convolutional layers capture low-level features. Residual blocks introduce skip connections, allowing the addition of layer outputs to subsequent layers. Intermediate layers between the blocks contribute to hierarchical feature learning. Towards the network's end, global average pooling (GAP) is applied to reduce spatial dimensions. This is followed by a final FCL with softmax activation, commonly used for classification (He et al., 2016). ResNet is suited at capturing different levels of features and dealing with gradient issues during training. However, it becomes more complex and require more computing power because of the skip connections. Also, the accuracy might stop improving and even start getting worse as the network gets deeper (He et al., 2016; Leonidas & Jie, 2021; C. Zhang et al., 2021).

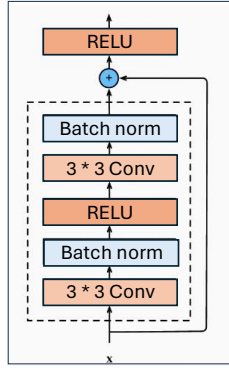


Figure 17. ResNet Blocks redrawn from (H. Wu et al., 2022)

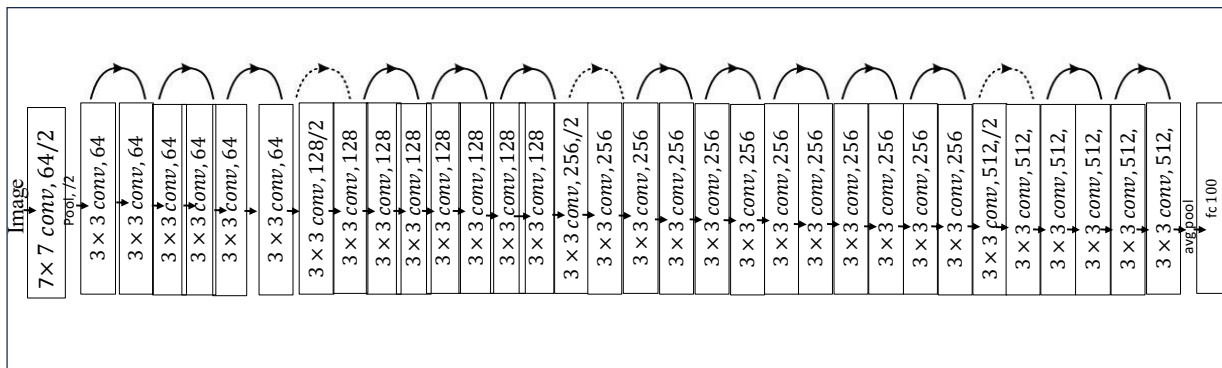


Figure 18. Architecture of ResNet34 redrawn from (He et al., 2016)

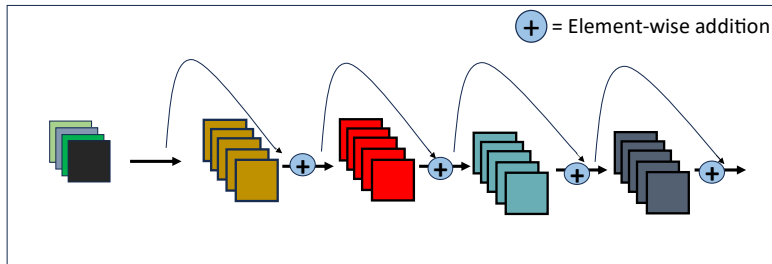


Figure 19. ResNet's Shortcut Connections for Information Flow extracted idea from (Huang et al., 2016)

### c. DenseNet

DenseNet architecture was highlighted (Huang et al., 2016). This architecture is a DL framework. This architecture allocates Dense Blocks. In these blocks, each layer's output is connected to every other layer in the same block. This enhances information flow and encourages feature reuse, leading to better gradient flow, improved feature propagation, and more efficient parameter usage (Huang et al., 2016). In other words, in a DenseNet, each layer gets input from not only the layer before it but also directly from all the previous layers in the network, as shown

in Figure 20. Transition layers are important in DenseNet. They are placed between dense blocks, as seen in Figure 21. These layers help control how many features are created and decrease the number of parameters. Transition layers have a batch normalization layer, then a 1x1 convolution layer, and finally a 2x2 average pooling layer. These layers work together to compress information and decrease the size of the data. As the network goes on, it uses GAP when it finishes. This process makes the size of the feature maps smaller. It ends up with just one value for each feature map. This gives a condensed version of the whole input. This pooling step helps to summarize the extracted features and get them ready for the final parts of the network. Bottleneck layers are important parts of DenseNet architectures. These layers use 1x1 convolutions to make fewer channels before using the 3x3 convolutions. This helps make the calculations less complex while still keeping the capacity to represent things. Using bottleneck layers makes DenseNet models work better for different computer vision tasks. This includes jobs like image classification and object detection. In the picture of DenseNet architecture, there is a five-layer dense block depicted in Figure 20. The process begins by taking input and doing convolution, batch normalization, and convolution operations. All of these together make a feature map, which we call  $x_2$ . Importantly, this convolutional operation, combined with batch normalization, helps create  $x_2$ . As the network moves from  $x_2$  to  $x_3$ , it is important to note that DenseNet’s dense connectivity plays a role. In this transition, the model doesn’t just use information from  $x_2$ . It also brings together features from earlier layers like  $x_1$ ,  $x_0$ , and more. This connection happens through concatenation, forming a strong link within the block (Huang et al., 2016).

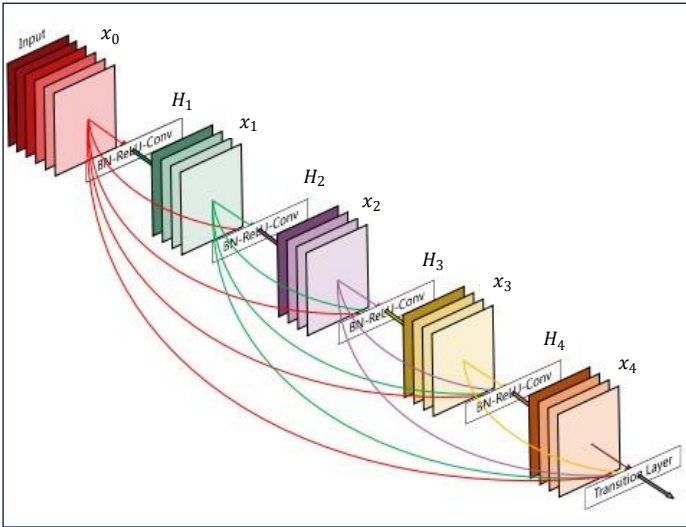


Figure 20. DenseNet block feature concatenation taken from (Pradhan et al., 2024)

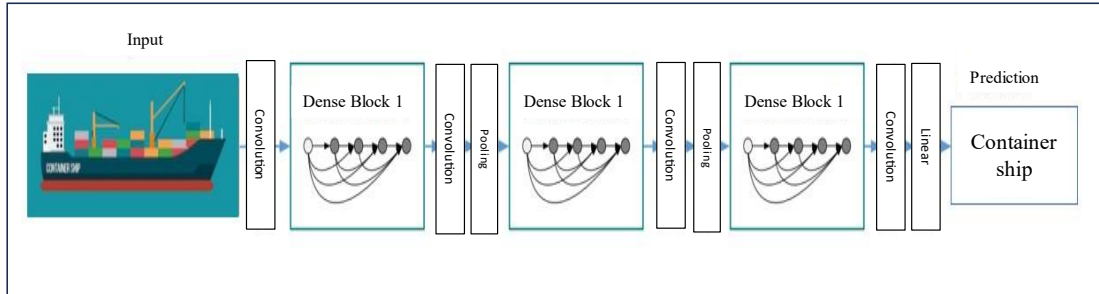


Figure 21. DenseNet architecture with three Dense blocks and transition layers redrawn from (Kumar et al., 2022)

## 2.5. Image Classification In Maritime Applications

There are a variety of maritime applications for image classification. In this section, some research papers that have been conducted on the classification of ships using different algorithms will be described, sorted by the proposal date of the algorithms as specified by the authors.

Solmaz et al. (2017) introduced an algorithm (Solmaz et al., 2017). It is used to classify diverse ships and gather various information about each ship. The research used a large-scale image dataset known as MARVEL. This dataset consists of 2 million user-uploaded images and their various attributes. The tasks performed encompassed vessel type classification, verification retrieval, recognition, and prediction and classification of vessel attributes like length, summer deadweight, draught, and gross tonnage. The paper used CNNs for feature extraction from the images. Two CNN architectures, AlexNet and VGG-F, were worked for extracting features from images. The author did not utilize the classification ability of CNNs. Multi-class SVM is used for the classification tasks after feature extraction. Also, the paper used a Siamese neural network, similar to the AlexNet architecture, for the verification process. In other words, it verified whether the SVM classification is classified correctly or not. This comparison revealed that the combination of CNN and SVM was more effective for their specific task. Using CNN alone for both feature extraction and classification was less effective.

Zhao et al. (2020) used several visible maritime image (VMI) datasets (Zhao et al., 2020). The categories involve various types of maritime vessels such as passenger ships, sailing vessels, yachts, tankers, and more. The authors presented an algorithm for classifying the vessel. It is a combination of one type of CNN suited for image classification tasks. Those are known as EfficientNet and fine-tuning. EfficientNet was used for the lower layers. These layers have been pre-trained on a large dataset like ImageNet. This part is used to extract features at the beginning.

The upper layers of the network were fine-tuned on the VMI datasets. This adjustment made these layers more specific to the task of classifying maritime images.

Salem et al. (2023) presented another algorithm for ship classification (Salem et al., 2023). It includes five types of ships such as cargo, military, carrier, cruise, and tanker. They used different datasets such as Kaggle's public "Game of Deep Learning Ship" dataset and the "MARVEL" dataset. The MARVEL dataset includes 10,000 image samples for each class and 26 types of ships. This helps in generalizing the classification system across a wide range of ship types. The paper mentioned two techniques used in this research work: the Transfer learning technique and the ensemble learning technique. In the Transfer learning technique, the lower layers of the pre-trained models were used in this paper. They were sourced from ImageNet because of their ability to detect generic features. The top layers, specific to the tasks they were originally trained for, were adjusted with 8 CNN models. These models contain Xception, VGG-16, ResNet-50, Inception V3, InceptionResNetV2, DenseNet121, MobileNet, MobileNetV2, and EfficientNetB0. This guaranteed that the network functions correctly for the precise task of ship classification. Also, another fine-tuning is done by adding 3 CNN layers to each model. These layers comprise a FCL, a dropout layer, and a ReLU layer. Then, ensemble learning technique was used to make predictions better overall.

Emre Gulsoylu et al. (2024) did the vessel classification as well as classified the ship's information (Gülsoylu et al., 2024). The system was trained with images of ships. Additional data like the ship type, size, and destination from the Automatic Identification System (AIS) was used to improve accuracy. The system becomes more effective by training the model with this extra information. This is true even in challenging weather conditions. This enhanced accuracy is particularly useful. It helps reduce port congestion, manage waterway traffic, and improve maritime surveillance. Basically, the system is better equipped to recognize and monitor sea activities. This is because it combines visual data with detailed ship information.

## **2.6. Conclusion**

In this chapter, many ways that ML techniques can be used to classify images were studied. Special attention was given to how these techniques are applied in maritime applications. The literature review has clarified the key differences between conventional machine learning

and DL. How each method is suited for different parts of the image classification task is demonstrated.

The study began by examining conventional machine learning techniques. The role of feature extraction and classification through algorithms was presented. Those are like SIFT, ORB, and SURF. These methods have been foundational in improving systems. They could interpret and analyze images. However, they depend on feature selection and engineering manually.

The DL importance and its profound impact on the field were discussed. That leads to the ability to autonomously learn from data. Also, it avoids the hard work of manually designing features needed in traditional methods. The world of image classification has been advanced by the learning skills of CNNs and other deep network setups. Particularly, in recognizing and categorizing images, CNNs are effective. Recognizing complex patterns and relations automatically can be achieved through. That might often be hidden from conventional algorithms.

In maritime applications, the necessity of implementing these learning technologies was mentioned. Various studies were reviewed that employed ML algorithms to address challenges specific to maritime environments. These contain ship classification and the detection of maritime vessels under diverse conditions. The adaptability and robustness of ML in use-case scenarios are highlighted through these applications. Diverse conditions and requirements necessitate robust and flexible solutions.

## Chapter 3: Literature review (PART II), Ensemble Learning and Pruning

### 3.1 Introduction

In this chapter, ensemble learning techniques will be presented as a powerful approach to ML. Ensemble learning improves predictions by merging multiple models. This chapter also looks at how ensemble learning increases accuracy as well as its significance in the field of learning. This chapter will focus on different ensemble methods. These contain bagging, boosting, stacking, and gradient boosting. This chapter also breaks down their workings for easier comprehension and demonstrates their practical applications in real-world scenarios.

Importantly, ensemble pruning will also be presented. This simplifies ensemble learning, making it easier and faster without compromising accuracy. The goal is to find a balanced solution between diverse model selection, computational efficiency, and precise results within ensemble learning strategies and their applied pruning techniques.

### 3.2 Problem Definitions

Ensemble learning is an advanced ML technique. It includes joining predictions from numerous individual models, often called base or weak learners.

The main goal of ensemble learning is to generate a stronger, more accurate model, called the ensemble or meta-model. This is accomplished by leveraging the variety and complementary strengths of base models (Sagi & Rokach, 2018).

- **Problem Definition 1 [Ensemble Learning]**

Consider a set of base models  $\{M_1, M_2, \dots, M_n\}$ . Here,  $n$  signifies the total number of base models. Also, assume the set of training samples  $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ . Here,  $m$  presents the number of training samples,  $x_i$  is the  $i$ -th input sample, and  $y_i$  is the corresponding target label. An ensemble prediction function is defined, denoted as “ $E(x)$ .” This function mixes the predictions of the base models to generate a final prediction for a given input sample  $x$ .



$$E(x) = \text{MergeFunction}(M1(x), M2(x), \dots, Mn(x))$$

(Equation 1. Extracted idea from (Rokach, 2019, p. 19) )

“ $M_i(x)$ ” is the prediction made by the  $i$ -th base model for input sample  $x$ . MergeFunction aggregates these predictions, using methods like averaging, voting, or weighted combination. Each base model, “ $M_i$ ”, learns a mapping from input samples “ $(x)$ ” to predictions “ $M_i(x)$ ” on the training dataset. The training process may include optimizing a loss function, “ $L_i$ ”, for each model.

$$M_i = \text{argmin}(L_i(M_i(x), y))$$

(Equation 2. Extracted idea from (Rokach, 2019) )

“ $L_i$ ” represents the loss function associated with the  $i$ -th base model, and  $y$  is the target label.

- **Problem Definition 2 [Objective Function]**

The goal of ensemble learning is to minimize prediction error. It achieves this by finding the best combination of base models and their prediction functions. This is typically measured using a loss function, “ $(L(E(x), y))$ ”, which quantifies the difference between the ensemble’s prediction and the true target label (Rokach, 2019, p. 4).

$$E^* = \text{argmin}(L(E(x), y))$$

(Equation 3. Extracted idea from (Rokach, 2019) )

“ $E^*$ ” is the optimal ensemble prediction function that minimizes prediction error.

The ensemble learning process encompasses choosing suitable base models, training them, and defining the ensemble prediction function. This makes a final model with enhanced predictive performance compared to individual base models (Rincy & Gupta, 2020). From a theoretical perspective, the complexity of an ensemble classifier can be analyzed in terms of both computational and data complexities (Kuruvayil & Palaniswamy, 2022). The complexity of an ensemble classifier depends on the number and complexity of base models. The computational complexity scales linearly with the number of base models  $O(n)$ . Considering that each base model requires training and evaluation during both phases. So, with “ $n$ ” base models, the computational cost is “ $n$ ” times that of a single base model. The selection of base models considerably influences the computational complexity. Complex models such as deep

neural networks or large ensembles require extensive computational resources and time. However, simpler models, such as linear models, are less computationally intensive (Dong et al., 2020). Ensemble learning faces two primary challenges. The first is demanding resource requirements, consuming substantial computational time and memory. This stems from loading and executing all individual models during inference, potentially causing significant delays. The second challenge relates to detrimental effects introduced by one or more models. Cooperative dynamics can lead to specific models exerting a counterproductive influence, hindering performance improvements intended by ensemble methods. To overcome these challenges, ensemble pruning has been developed. (Roçah, 2010, p. 119). Ensemble pruning improves the efficiency and simplicity of an ensemble classifier. It selects a subset of its base models while maintaining or enhancing predictive performance (Fu et al., 2013; Roçah, 2010, p. 119). Later, this thesis will introduce the ensemble pruning paradigm.

- **Problem Definition 3 [Ensemble Pruning]**

Consider an original ensemble classifier with  $n$  base models:  $M = \{M_1, M_2, \dots, M_n\}$ . Ensemble pruning aims to return a subset of pertinent models:  $M^* = \{M_{i_1}, M_{i_2}, \dots, M_{i_k}\}$ , where  $k$  is less than or equal to  $n$ . This is based on a pruning criterion, denoted as “ $P(M_i)$ ,” quantifying the effectiveness or utility of each base model “ $M_i$ .”

A decision rule, “ $D(P(M_i))$ ,” determines whether a base model  $M_i$  should be included in the pruned ensemble. If “ $D(P(M_i))$ ” evaluates to “retain,” the model is kept; otherwise, it is pruned (Rokach, 2019, p. 121).

### 3.3 Ensemble Learning Techniques

Ensemble learning is one of the potent techniques used in the ML domains. It leverages the collective wisdom of numerous learning models to advance the accuracy and robustness of a single base model. It combines the predictions of these models to generate a more reliable and powerful predictor. Ensemble learning techniques are priceless tools in the ML practitioner’s toolkit. They can be categorized into four core techniques: “Bagging,” “Boosting,” “Stacking,” and “Gradient Boosting,” as described below.

### 3.3.1 Bootstrap Aggregating (Bagging)

The bagging technique includes training several copies of a single base model on different subsets of the training data. The subsets are regularly found through random sampling with replacement. Each model learns from a slightly different view of the data, reducing overfitting, particularly in high-variance models (Breiman, 1996). The ultimate prediction is gained by averaging or voting on the predictions of all individual models, as shown in Figure 22. Bagging is particularly useful for enhancing model robustness, especially when the base model is prone to overfitting. For each bootstrap sample, a copy of the base model is trained independently. These models learn from different views of the data due to the variations presented by bootstrap sampling. This diversity is critical for reducing overfitting in models with high variance. After training, all individual models make predictions on new, hidden data points. The ultimate prediction is gained by aggregating the outputs of all individual models as shown in Figure 23. The most common aggregation methods are: “Averaging” and “Voting.” Averaging includes averaging predictions from each model to extract the final prediction. Voting involves treating each model’s prediction as a vote. Then, the class with the most votes is selected as the final prediction, known as majority voting (Breiman, 1996; Mohammed & Kora, 2023).

Bagging helps mitigate overfitting by decreasing the variance of the final prediction, particularly in complex models. Training on various subsets makes the model more robust to noise and outliers in the data. Additionally, bagging frequently leads to better generalization performance, resulting in a more accurate model for hidden data (Mohammed & Kora, 2023).

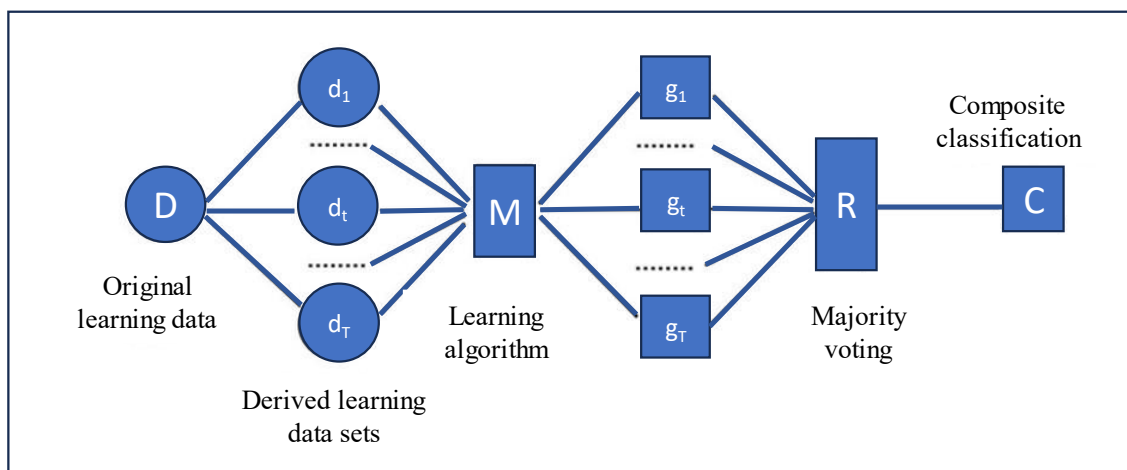


Figure 22. Bagging redrawn from (Di Ciaccio & Giorgi, 2016)

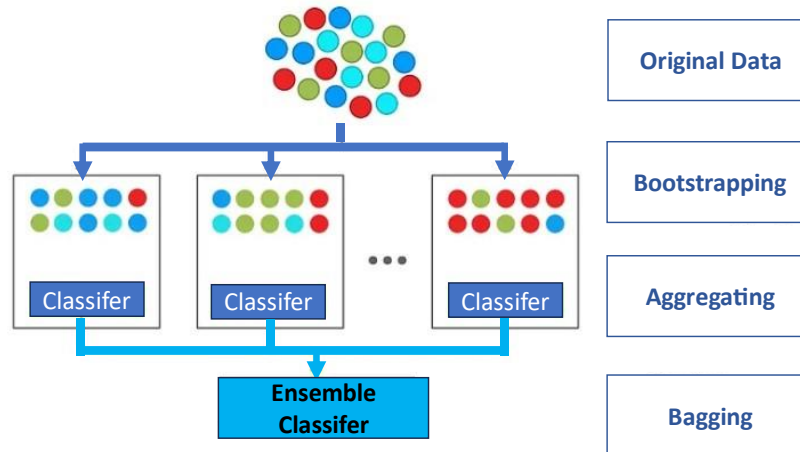


Figure 23. bagging technique, ensemble models are trained on bootstrap redrawn from (Zuchniak, 2023)

### Example:

In a binary classification task, the goal is to classify emails as either spam or not spam. Considering there is a dataset with 1,000 email samples. Bagging can be applied to these email sample scenarios as follows: It creates multiple bootstrap samples, each containing a random subset of the original data. For example, one sample might contain [Email1, Email5, Email7, Email10, ...], while another could contain [Email2, Email3, Email6, Email9, ...], and so forth. These subsets are usually the same size as the original dataset but with some instances repeated and others omitted. During training, each bootstrap sample is used to train a copy of the base model (e.g., a decision tree). Each model learns to classify emails as spam or not spam based on its individual data subset. During prediction, when a new email receives, all individual models forecast whether it is spam or not. To make the final decision, the predictions from all models are aggregated. For example, if you have 10 models and 7 predict “spam” while 3 predict “not spam,” the email is classified as “spam” based on the majority vote prediction.

### 3.3.2 Boosting

Boosting is a technique designed to enhance the predictive accuracy of weak learners. The weak learner turns into a strong learner, as shown in Figure 24. This process executes slightly better than random guessing. Unlike bagging, boosting trains models in sequence. Each new model focuses on the samples that were wrongly classified by the previous ones (Freund & Schapire, 1996). Boosting algorithms broadly work for tasks related to classification and

regression, such as AdaBoost (Schapire, 2013) and Gradient Boosting (Konstantinov & Utkin, 2021).

Boosting begins with a base model, frequently considered as a weak learner. These weak learners' models perform slightly better than random guessing and can be simple linear models or other models with less complexities but better than chance. During training, each weak learner emphasizes on the samples that were mistakenly classified by the previous models. This adaptive learning process decreases bias and leads to improved accuracy over time. In each iteration, the training data is given different weights. Primarily, all data points have equivalent weights. Though, as boosting progresses, the misclassified data points are assigned higher weights, making them more significant in the subsequent model's training. After training all the weak learners, their predictions are combined to form the final prediction. Boosting usually combines predictions by weighted voting, where each model's contribution to the final prediction is weighted based on its performance during training, as depicted in Figure 25. Boosting regularly yields extremely accurate models, even when using weak learners. It can also manage noisy data and outliers efficiently by concentrating on the most challenging samples. It typically outcomes in models that generalize well to unseen data (Mohammed & Kora, 2023).

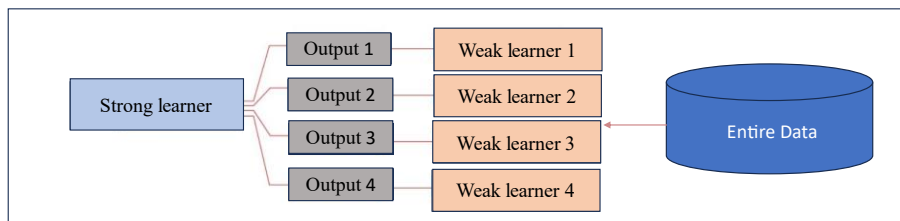


Figure 24. Boosting technique extracted idea from (Freund & Schapire, 1996)

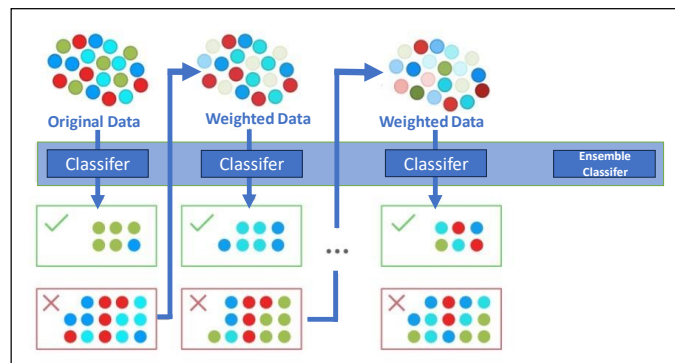


Figure 25. Sequential training of weak learners in Boosting with Weighted Voting redrawn from (Tiu et al., 2022)

**Example:**

Consider boosting in a binary classification task: identifying whether a bank loan application will be approved (1) or denied (0). The first weak learner is trained on the original dataset, where all samples have equivalent weights. It may misclassify some loan applications. After the first model's training, the misclassified loan applications are assigned higher weights, making them more significant in the subsequent training iterations. The second weak learner emphasizes on the misclassified loan applications from the first round. It attempts to correct the errors built by the first model and, in turn, may generate some new errors. After the second model's training, the next iteration assigns higher weights to the recently misclassified samples. This process repeats for a predefined number of iterations or until a desired performance threshold is reached. Each new model pays more consideration to the samples that previous models struggled with, successfully "boosting" their performance. While all the weak learners are trained, their predictions are combined. The final prediction is usually determined through weighted voting. This means that more accurate models have more influence in making decisions.

**3.3.3 Stacking**

Stacking is also known as stacked generalization and is a more sophisticated ensemble learning approach. It encompasses training various diverse models, often of diverse types or with different hyperparameters, on a similar dataset (Smyth & Wolpert, 1997). Instead of directly combining their predictions, a meta-model is trained to learn how to best combine the outputs of the base models, usually considering a simple linear regression or another ML model. Stacking can capture the strengths of each base model and provide a more robust and accurate prediction. It begins with a set of diverse base models. These base models can be of diverse types, such as decision trees, neural networks, or SVMs, or they can have different hyperparameters. The main concept is to ensure diversity among the base models to capture diverse data aspects. The original training data is operated on to train each of the base models separately. Each base model learns to predict based on its unique perspective of the data. After training, the base models are utilized to predict a validation set or a hold-out subset of the training data. These predictions are considered as the input features for the meta-model. The meta-model is then trained on the predictions made by the base models in the previous step. This meta-model takes the base models' predictions as input features. It also learns how to combine them optimally to make the ultimate prediction.

Once the meta-model is trained, it can be used to predict new, unseen data. Leveraging the learned relations between the base models' predictions, the meta-model produces the final output, as shown in Figure 26. Stacking regularly outperforms individual base models. This leads to advanced predictive accuracy. Stacking allows for harnessing the strengths of different models. This makes it operative for complicated tasks. It can also manage diverse types of base models. This makes it more robust to different data patterns (Mohammed & Kora, 2023).

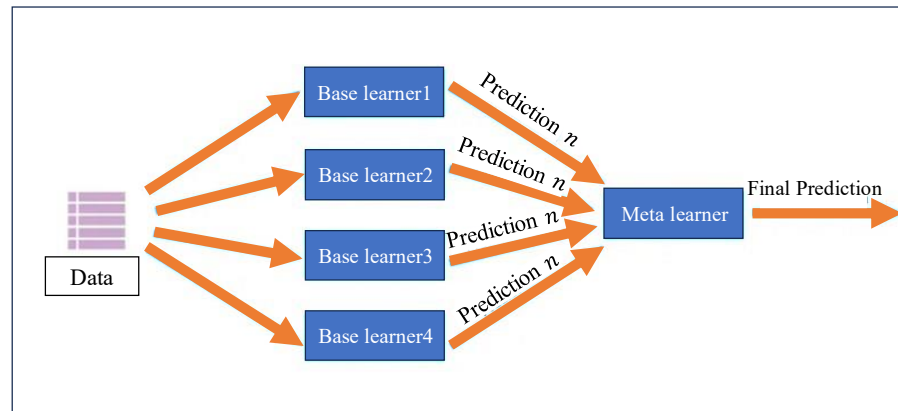


Figure 26. Stacking technique: Meta-model trained on Base Models' predictions redrawn from (Siswantining & Parlindungan, 2021)

### Example:

Assuming stacking is used in a binary classification task for sentiment analysis. In a dataset of movie reviews, the objective is to predict whether a review is positive (1) or negative (0). Three diverse base models are chosen: “a decision tree,” “a SVM,” and “a neural network.” The decision tree will learn patterns in the text data. SVM will capture linear relations between words, and the neural network will capture complex, nonlinear relationships in the data. Each base model is trained on the original movie review dataset, learning from diverse data aspects. They create individual predictions on a validation set or a subset of the training data. Below are the sets of predictions defined as an example:

- The decision tree might predict [1, 0, 1, 0, ...]
- The SVM might predict [0.8, 0.1, 0.9, 0.2, ...]
- The neural network might predict [0.9, 0.3, 0.95, 0.1, ...]

A meta-model, such as logistic regression, is then trained using the predictions from the base models as input features. The meta-model learns how to consider and combine these

predictions optimally to make the final sentiment prediction. Once trained, the meta-model can take a new movie review, pass it through the base models, and combine their predictions to produce the final prediction (Smyth & Wolpert, 1997),(Mohammed & Kora, 2023). In this example, the meta-model will learn from the set of predictions which are [1, 0, 1, 0, ...], [0.8, 0.1, 0.9, 0.2, ...], and [0.9, 0.3, 0.95, 0.1, ...] to find the final prediction.

### **3.3.4 Gradient Boosting**

Gradient Boosting is one of the robust techniques in ensemble learning. This technique combines boosting with gradient descent optimization. As shown in Figure 27, weak learners are trained sequentially. Also, each learner goals to decrease the residual error of the previous one (Konstantinov & Utkin, 2021). This process emphasizes areas where the previous models struggled. This leads to making the ensemble increasingly accurate with each iteration. In various ML tasks, particularly in computer vision, the most popular gradient-boosting implementations that have achieved state-of-the-art performance are XGBoost (T. Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018). Gradient Boosting usually uses decision trees as weak learners. These trees are often shallow, referred to as “stumps” or “shallow trees,” to avoid overfitting.

Gradient Boosting begins with a preliminary prediction. This is usually set as the average value of the target variable for regression tasks or as the most common class for classification tasks. This initial prediction acts as a baseline. A decision tree is trained to predict the residuals from the initial prediction, which are the distinct between the real target values and the current model’s predictions. This tree is suited to decrease the residual error. After the first iteration, another decision tree is trained to predict the residuals, concentrating on the errors built by the first tree. The process stays with additional iterations, each new tree targeting to decrease the remaining residuals. The final prediction is found by adding the predictions from all the trees in the sequence. The cumulative result of these trees is to correct the errors built by previous models and generate a more accurate overall prediction. Gradient Boosting often generates highly accurate models, mainly when combined with shallow trees, which are less prone to overfitting. Similar to other ensemble learning techniques, gradient boosting can manage noisy data and outliers efficiently by concentrating on decreasing residuals. It can also provide insights into



feature importance, helping recognize which features are most influential in making predictions (Guillen et al., 2023; Konstantinov & Utkin, 2021).

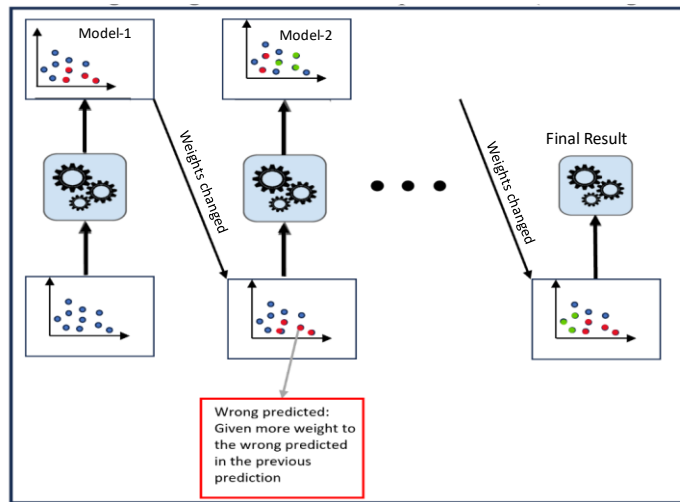


Figure 27. Gradient Boosting: Each new model predicts Residual Errors of previous model redrawn from (Aouadi et al., 2023)

### Example:

Consider a regression task where the target is to predict the price of a house based on its features. The process begins with a simple initial prediction, such as the average house price. For example, in this case, the average house price in the dataset is assumed to be NOK 3,000,000. The first decision tree is trained to predict the differences between actual prices and NOK 3,000,000 from the initial prediction. This tree recognizes patterns in the data that define the differences. A second decision tree is trained to predict the residuals from the first iteration. It focuses on capturing the remaining errors not described by the first tree. This process continues with additional trees and iterations. Each new tree is planned to correct the residuals left by the previous ones, gradually decreasing the prediction error. The final prediction is the sum of the initial prediction and the predictions from all the following trees. It precisely estimates the house price by considering a series of corrections to the initial estimate.

### 3.4 Ensemble Pruning Techniques

Ensemble pruning techniques include choosing a subset of base models from an existing ensemble of models. This differs from building an ensemble of models from scratch. The objective of ensemble pruning is to simplify and improve the efficiency of an existing ensemble while maintaining or even increasing predictive performance. This goal can be accomplished

through two main techniques: “Dynamic Model Selection” and “Model Pruning By Diversity,” each of which is explained below in detail.

### 3.4.1 Dynamic Model Selection

Dynamic model selection encompasses various techniques that adapt model ensembles based on performance and changes in the data. These techniques contain adaptive weighting, which adjusts the effect of each model within the ensemble based on its recent performance. Another technique is the windowed ensemble, which regularly updates the ensemble by incorporating new models and excluding the least-performing ones within a particular time frame (Roçak, 2010, p. 121).

#### a. Adaptive Weighting

Adaptive Weighting is a dynamic model selection technique. It adjusts the weights of distinct models within an ensemble based on their performance over time. Models that operate well on recent data receive higher weights, while underachieving models are assigned minor weights, as shown in Figure 28. This permits the ensemble to adapt to changes in the data distribution. It gives more effect to models that are presently more precise (Gale et al., 2013; Hagiwara, 1993; Zhou et al., 2002).

Consider “ $w(i, t)$ ” representing the weight assigned to model “ $i$ ” at time “ $t$ .” The adaptive weighting can be stated as follows: Initially, all models might have the same weights:

The initial weights are set as follows: “ $w(i, 0) = 1/N$ ” for “ $i$ ” in “[1, N],” where “ $N$ ” is the number of models. At each time step “ $t$ ,” the weights are updated based on the model’s performance on the most recent data:

$$w(i, t+1) = f(w(i, t), performance(i, t))$$

(Equation 4. Extracted idea from (Gale et al., 2013))

Where performance ( $i, t$ ) is a measure of the model “ $i$ ” performance on the data at time “ $t$ ,” and “ $f$ ” is a function that updates the weights. This function can be designed based on specific criteria, such as a weighted moving average or an adaptive learning rate.

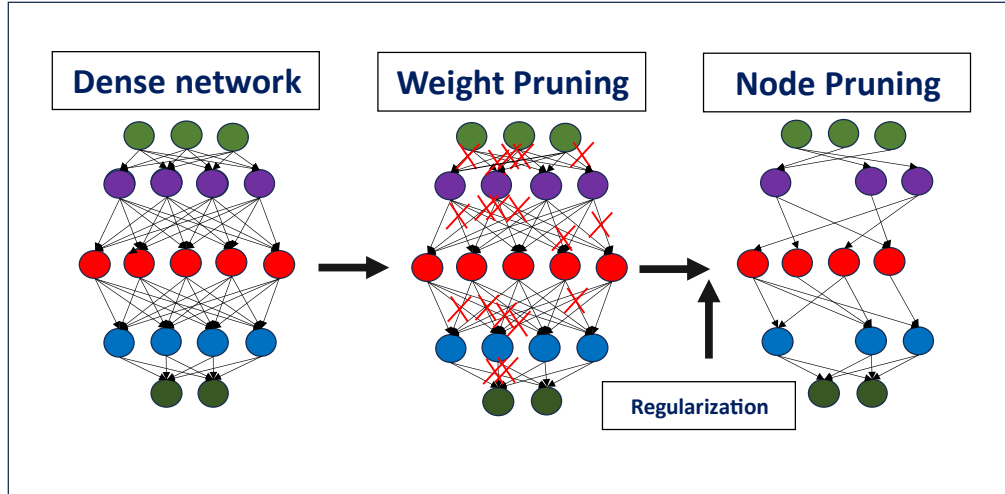


Figure 28. Adaptive weighting in ensemble pruning for dense neural network redrawn from (Ziv et al., 2021)

**Example:**

Consider an ensemble of three classifiers for a sentiment analysis task: “Model A,” “Model B,” and “Model C.” Initially, all models have equal weights: “ $w(A, 0) = w(B, 0) = w(C, 0) = 1/3$ .” After analyzing data from a new time period, the models’ performance is assessed, and it is found that Model B performed exceptionally well, Model A performed adequately, and Model C underperformed. The adaptive weighting can be updated based on their performance scores, as indicated below.

- $w(A, 1) = f(w(A, 0), \text{performance}(A, 1))$
- $w(B, 1) = f(w(B, 0), \text{performance}(B, 1))$
- $w(C, 1) = f(w(C, 0), \text{performance}(C, 1))$

The update function  $f$  might increase the weight of Model B, decrease the weight of Model C, and leave Model A’s weight relatively unchanged based on their performance.

**b. Windowed Ensemble**

The windowed ensemble is a dynamic model selection technique. It maintains a sliding window of models and data. At each time step, the least-performing model within the window is substituted with a new model. This ensures the ensemble always has the most suited models for the present data distribution (Gupta & Jha, 2020).

In a windowed ensemble technique, denote the ensemble at a time “ $t$ ” as “ $E(t)$ ” and the size of the sliding window as “ $W$ .” At each time step “ $t$ ,” a new model, denoted as “ $M(t)$ ,” is trained on the newest data. The ensemble “ $E(t)$ ” includes the best-performing models from the

last “ $W$ ” time steps, and the least-performing model within “ $E(t)$ ” is substituted with the recently trained model “ $M(t)$ .”

### **Example:**

Consider an ensemble of decision trees for a fraud detection task, with a sliding window size “ $W$ ” set to 5. At time step “ $t=1$ ,” the ensemble comprises Decision Trees 1, 2, 3, 4, and 5, trained on the data from time steps 1 to 5. At time step “ $t=6$ ,” a new decision tree, Decision Tree 6, is trained on the data from time step 6. The ensemble at time step  $t=6$  will contain Decision Trees 2, 3, 4, 5, and 6. Decision Tree 1, which was the least-performing model in the former ensemble, is changed by Decision Tree 6.

### **3.4.2 Model Pruning By Diversity**

The success of ensemble methods often depends on the diversity of constituent models. Diversity stands from variations in model architectures, training data subsets, hyperparameters, or other factors. Models that are diverse in their predictions can suggest complementary insights, leading to enhanced total performance. Model pruning by diversity depends on measuring how similar or different individual models are (Fu et al., 2013). To achieve this, similarity metrics are used to compare the predictions or behaviors of models. As shown in Figure 29, the process starts with individual models including M1, M2, M3, and M4. Each process is also evaluated separately at the highest layer. Subsequent layers define blends of these models. Thus, pairs are evaluated first, followed by triples, and eventually all four models together. This hierarchical structure is used for a systematic evaluation of diversity within the ensemble. Models are combined and pruned based on their performance and the diversity they contribute. This is evaluated by the following common similarity metrics: “Correlation,” “Distance,” and “Overlap of Predicted Classes.”

1. **Correlation:** This measures the linear relationship between the predictions of different models. A high correlation indicates that models tend to produce similar outputs (Mana & Sasipraba, 2021).
2. **Distance:** Euclidean distance and other distance-based metrics assess the proximity of models in a multi-dimensional space defined by their predictions. Models that are close to this space are more similar (Peters, 2017).

3. **Overlap of Predicted Classes:** Another approach is to evaluate how often models agree on the predicted classes or outcomes. The high agreement indicates a similarity (Ren et al., 2020).

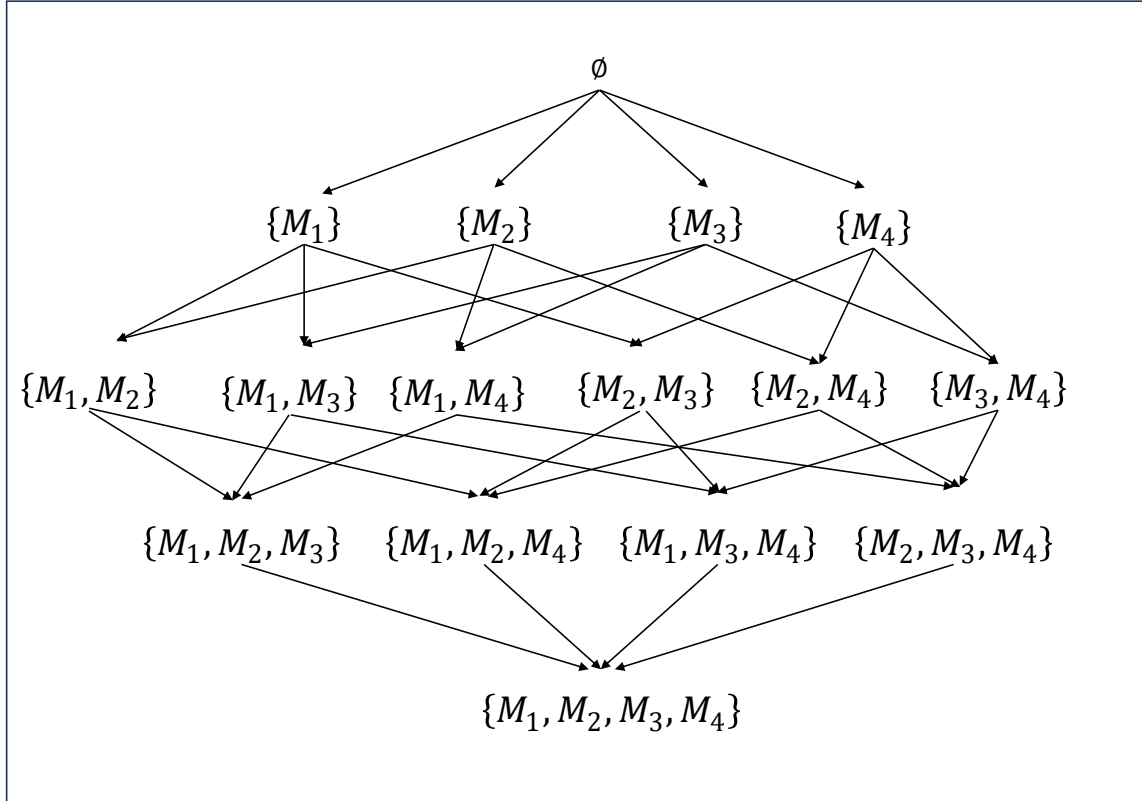


Figure 29. Model pruning by diverse redrawn from (Partalas et al., 2008)

Based on the computed similarity metrics, models that are extremely similar to others are pruned or downweighted. Different approaches to pruning include “Hard Pruning,” “Soft Pruning,” and “Threshold-based Pruning,” each described below with an example.

1. **Hard Pruning:** Hard pruning includes eliminating models from an ensemble if their similarity to other models exceeds a predefined threshold. The threshold is typically determined based on a similarity metric such as correlation or distance. This method simplifies the ensemble by removing redundant models (Ding et al., 2021; Valerio et al., 2022).

Consider “sim(i, j)” to represent the similarity between models “i” and “j,” computed using a specific similarity metric. The hard pruning condition can be expressed as follows: “If  $\text{sim}(i, j) > \text{threshold}$ ,” then model “j” is removed from the ensemble.

**Example:**

Consider an ensemble comprising three neural networks (A, B, and C) for image classification. If the cosine similarity between neural networks A and B exceeds a predefined threshold of 0.9, one of the two similar networks (for example, B) is removed from the ensemble.

- 2. **Soft Pruning:** Soft pruning is also known as model weighting. The soft pruning technique retains all models in the ensemble. It allocates lower weights to similar models. This leads to diminishing the impact of similar models while keeping them in the ensemble (Ding et al., 2021).

Consider “ $w(i)$ ” representing the weight assigned to model “ $i$ .” The soft pruning condition can be expressed as: “If  $sim(i,j) > threshold$ ,” then set “ $w(j) = w(j) \times factor$ ,” where the factor is typically a value between 0 and 1.

**Example:**

Consider an ensemble of five regression models: M1, M2, M3, M4, and M5, for predicting housing prices. If the Pearson correlation between models M2 and M4 is high (e.g., 0.95), you may decrease the weight of M4 to 80% of its original weight by setting factor = 0.8.

- 3. **Threshold-based Pruning:** -- Threshold-based pruning involves setting a threshold for the similarity metric. Models exceeding this threshold are considered for pruning. This technique provides flexibility to control the degree of pruning (Azarian et al., 2021).

Consider “ $T$ ” to be the threshold for similarity. Consider “ $P(i, j)$ ” to be a binary variable indicating whether model “ $i$ ” and model “ $j$ ” are pruned (1 for pruned, 0 for retained). The threshold-based pruning condition can be expressed as:

$\text{If } sim(i, j) > T, \text{ then set } P(i, j) = 1$ <p>(Equation 5. Extracted idea from (Azarian et al., 2021))</p>
---

**Example:**

In an ensemble of four clustering algorithms: C1, C2, C3, and C4, you set a threshold of 0.85 for the Jaccard similarity. If the Jaccard similarity between C3 and C4 exceeds the threshold, you would mark them for potential pruning (e.g.,  $P(C3, C4) = 1$ ).

After pruning, the remaining models are integrated to make the final prediction. Integration methods can vary and may include weighted averaging, voting, or more advanced techniques like stacking. The weights allocated to models regularly represent their degree of diversity or confidence in their predictions.

Here, the main advantages of model pruning by diversity are listed below, including “improved generalization,” “reduced redundancy,” “interpretability,” and “robustness.”

1. **Improved Generalization:** Pruning by diversity helps ensemble models. They suggest diverse views and ways to resolve problems. This variety can make the model better at learning many data patterns and types and making strong predictions (X. Chen et al., 2021).
2. **Reduced Redundancy:** Having too many redundant models, that do the same thing, can make it more expensive to run without improving much. Pruning by diversity can help eliminate these redundant models, making the group work better and saving time and money (Augasta & Kathirvalavakumar, 2013).
3. **Interpretability:** Keeping only diverse models can make the ensemble easier to understand. When the models are different from each other, it is simpler to see what each one does and how they help (Augasta & Kathirvalavakumar, 2013).
4. **Robustness:** When ensembles are pruned by diversity, they typically handle outliers and changes in data better. If some models are very sensitive to certain data points, having others that are not as sensitive can keep predictions accurate.

Below, the main disadvantages of model pruning by diversity are listed, including “selecting the right diversity metric,” “threshold selection,” and “balancing diversity and performance.”

1. **Selecting the Right Diversity Metric:** Deciding on the right diversity metric can be tricky because it must really show how models are different. The metric you pick depends on what problem you are solving and what the models are like.
2. **Threshold Selection:** Figuring out the best threshold for pruning models based on their differences can be challenging. That may need various methods such as experimentation, cross-validation, or domain expertise.

3. **Balancing Diversity and Performance:** Pruning should find a balance between keeping things diverse and keeping predictions accurate. If you prune too much, you might lose important information.

Model pruning by diversity is a fancy technique that makes ensemble models better by focusing on how different and useful each model is. It is really helpful when you have a bunch of different base models and when you care a lot about understanding the models and saving time.

Below, two different subsections mention the main discussions and limitations of “Ensemble Learning” and “Ensemble Pruning” in separate subsections. Each subsection’s first paragraph presents the discussions, and the next paragraph indicates the limitations.

### **3.4.3 Ensemble Learning**

Ensemble learning is beneficial in image classification for several reasons. By combining the predictions of various models, ensemble methods regularly gain higher accuracy. This is mainly valuable in image classification, where diverse patterns and features are required to be captured for enhanced overall performance. Ensembles prove their robustness in managing variations and noise in image data. This effectively leads to mitigating the impact of outliers and inconsistencies within the dataset. Additionally, ensemble methods assist in reducing the bias that individual models inherently possess. If a single model shows bias towards specific image characteristics, other models within the ensemble can compensate. This results in more balanced predictions. Moreover, ensembles suggest a suitable level of interpretability. By looking at how each model in the ensemble contributes to the final decision, users or stakeholders can understand what factors are affecting the image classifications.

However, ensemble methods also present certain limitations. They can be computationally expensive, especially with large datasets and complex base models. This complexity usually means it takes longer to train and make predictions. Furthermore, having many models in an ensemble uses up more memory. This can be difficult in places where there is not a lot of memory available, which is a problem in resource-constrained settings. Creating and improving ensemble models requires skill and knowledge. This involves picking the right base models, adjusting settings called hyperparameters, and coming up with strategies that work specifically for ensembles. This process might be time-consuming. Moreover, ensembles need regular



maintenance and updates, mainly when new data becomes available. Keeping an ensemble that can change with the data is tough work. It adds another challenge to the ensemble learning process, which is already complex and ongoing.

#### **3.4.4 Ensemble Pruning**

Ensemble pruning methods suggest diverse advantages. These advantages are often obvious when applied to ensembles in image classification. Pruning has the potential to decrease the size and complexity of ensembles. This develops their efficiency for real-time or embedded applications. This is mostly important in scenarios where computational resources are inadequate. It allows for streamlined processing without compromising accuracy. The pruning process shortens ensemble models. It makes them easier to interpret. This is appreciated when discovering the reasons behind classification decisions is essential. Also, pruning can help mitigate overfitting by concentrating on the most relevant models within the ensemble. By ranking these models, ensemble pruning enables better generalization when classifying new, unseen data, thus enhancing the total robustness of the classification system.

However, ensemble pruning may present challenges. Aggressive pruning strategies may lead to the loss of valuable information. That may potentially diminish the ensemble's ability to accurately classify data. Removing models that could contribute under specific conditions might unpleasantly affect classification accuracy, especially in complex or dynamic environments. Also, picking the right pruning thresholds can be a challenging task. The ideal thresholds are often related to the precise characteristics of the problem at hand. They require domain expertise or experimentation for actual implementation. In addition, pruning methods may not operate fine in scenarios characterized by extremely dynamic data distributions. In those cases, where data patterns change over time, traditional pruning techniques may attempt to maintain classification accuracy. Dynamic ensemble adaptation methods might be better suited for ensuring consistent performance as the data landscape shifts.

### **3.5 Conclusion**

This study has explored the multifaceted domain of ensemble learning. This chapter specifically looks at ensemble pruning. This method proposes to make ensemble models work better without making them worse at predicting things. Throughout this investigation, various

ensemble learning techniques were studied. These include bagging, boosting, stacking, and gradient boosting. Each method influences the variety of multiple learning models. This helps develop predictive accuracy and robustness beyond what separate models can achieve. Also, ensemble pruning has been offered as a key strategy. It helps report the complexities and resource supplies of traditional ensemble methods.

Ensemble pruning shortens ensemble systems and is an efficient approach. Pruning techniques validate that only the most effective models are retained by carefully selecting a subset of base models. Commonly employed techniques for this purpose include dynamic model selection and diversity-based pruning. On one hand, this selection not only diminishes the computational load and memory supplies. On the other hand, removing redundant or less operative models, also possibly enhances the ensemble's performance.

This chapter shows how helpful ensemble methods are in solving tough prediction difficulties. They are more appropriate practices in several different areas where predicting things is hard. These techniques deliver a powerful toolkit for tackling the challenges of modern data-driven spaces. This is particularly correct when they are combined with effective pruning strategies. They suggest a balanced approach to reaching high accuracy and robustness. This is while managing the computational costs related to large-scale data analysis.

## Chapter 4: Methodology Design (MAGNAT and MAGNUT)

### 4.1 Introduction

This chapter conducted a study on the methodologies of MAGNAT and MAGNUT in advanced maritime management systems. This chapter explains the proposed solutions based on using ensemble learning approaches. While ensemble learning is being developed, several issues arise. High computational resources and the negative contribution of the models are addressed as two main challenges. In the proposed solution, an accurate selection of the ensemble learning models has been developed. This selection focuses on those that bring positive contributions and eliminating those that do not. The discussion also covers how the author has designed strategies for pruning non-contributory elements to improve efficiency. Finally, this chapter overview sets the groundwork for future research directions. Additionally, the proposed methods can be applied in existing systems and highlight possible areas for reform.

### 4.2 Explanation of Research Questions

Following the research questions available in Section 1 under Subsection 1.2, which are available on page 15, this section provides a more detailed and clear explanation about the research questions in the following sentences.

Researchers are focusing on using AI, especially DL, in maritime management systems. DL trying to address the different challenges in this field. But still, the efficacy of different architectures varies, and some models work better than others with specific data.

To come up with the uncertainties in model behavior, researchers have explored ensemble learning. Ensemble learning by developing multiple models whose outputs are aggregated can enhance overall performance. Nevertheless, two prominent challenges remain unaddressed in ensemble learning:

1. **Computational Resource Intensity:** Ensemble models demand significant time and memory resources during the inference phase. This is because they have to run all models, which slows things down.

2. **Negative Contribution of Models:** Another challenge is existing models that have a negative impact within the ensemble which can influence the overall learning process.

In response to these challenges, this chapter proposes intelligent strategies. The main aim of this proposed intelligence is resolving two following issues: aimed at resolving the two following issues:

- **Model Selection:** Given a bunch of models that are made for the particular maritime task (classification in our context), the aim is to define the one that works best. This needs to see which of the models contribute positively and negatively to the learning process.
- **Performance Optimization:** Once the best models are defined, the subsequent challenge is how to use them to get the best overall performance. The focus is on developing efficient strategies to influence effectively.

The vision is to present innovative strategies for selecting the best model or models from the ensemble. This needs to address the main challenges such as computational challenges and also navigate the intricacies of model contributions. By doing so the overall effectiveness would be enhanced.

### 4.3 Data Preprocessing

Data preprocessing consists of a series of steps. The main purpose of these steps is to improve the quality of raw data before being fed into an algorithm for further processing. In the perspective of marine remote sensing images, effective feature extraction is critical. Specifically, to correctly identify marine targets (Q. Chen et al., 2018). Preprocessing includes several methods and techniques, dependent on the specific application or dataset. Nonetheless, the common preprocessing steps include resizing the image, normalizing pixels, and converting the colorful images into grayscale (Meena & Velmurugan, 2023). These steps improve the standardization of the data and make it more suitable for further analysis (Saponara & Elhanashi, 2022). The input data for image classification is a set of images. In this research, the input data is related to maritime scenarios.

The data preprocessing steps, as mentioned above, including “Resizing Images,” “Normalization,” and “Image Conversion to Grayscale,” are discussed below:

### 4.3.1 Resizing Images

Resizing involves rescaling the width and height of an image to a specific size. That makes ensuring the input images are uniform. This consistency is crucial in neural networks to prevent variations in input sizes during model training. This consistency is really important in neural networks. During model training, this helps prevent variations in input sizes. Also, to maintain consistency and computational efficiency in model design, resizing images is important. According to the model's input size requirements, keeping the aspect ratio and resizing seem to be the main considerations (Al-Barazanchi et al., 2018; Hashemi, 2020). Figure 30 depicts a visual representation of the resizing process.



Figure 30. Illustration of Resizing Process

### 4.3.2 Normalization

One of the vital steps is normalizing the pixel value for faster model convergence during training. By scaling pixel values to a common range, such as  $[0, 1]$ , stability is assured and features with larger scales would be prevented. Channel-wise normalization is applied independently to each color channel in red, green and blue (RGB) images, and standardization involves subtracting the mean and dividing by the standard deviation (z-score normalization) (Tipu et al., 2024). The code in Figure 31 is a Python function that normalizes and preprocesses data for a machine. As shown in the figure this function takes in several parameters where  $x$  is input data,  $y$  is the targets belongs to the input data. Height and width are the dimensions to which each input sample will be reshaped. Number\_channels are the number of channels in the input data. The num\_classes are the number of corresponding classes in the target data. The input data  $x$  is change over to a NumPy array (Van Der Walt et al., 2011). With the reshape command we reshape our image to the specific size (Joseph et al., 2021). we covert it to the “float32” to ensure that the division operation in normalization is carried out in floating-point arithmetic (Jacob et al., 2018). We divide each pixel value by 255 to bring all the pixel values into the range  $[0,1]$ , to make it normalized. The last part is converting one hot encoding of the target data by using Keras’s utility function (Chollet, 2015). Keras is an open-source software library that provides a Python interface for ANNs. One-hot encoding transforms the integer class labels into a binary matrix representation (Chollet, 2015; Joseph et al., 2021).

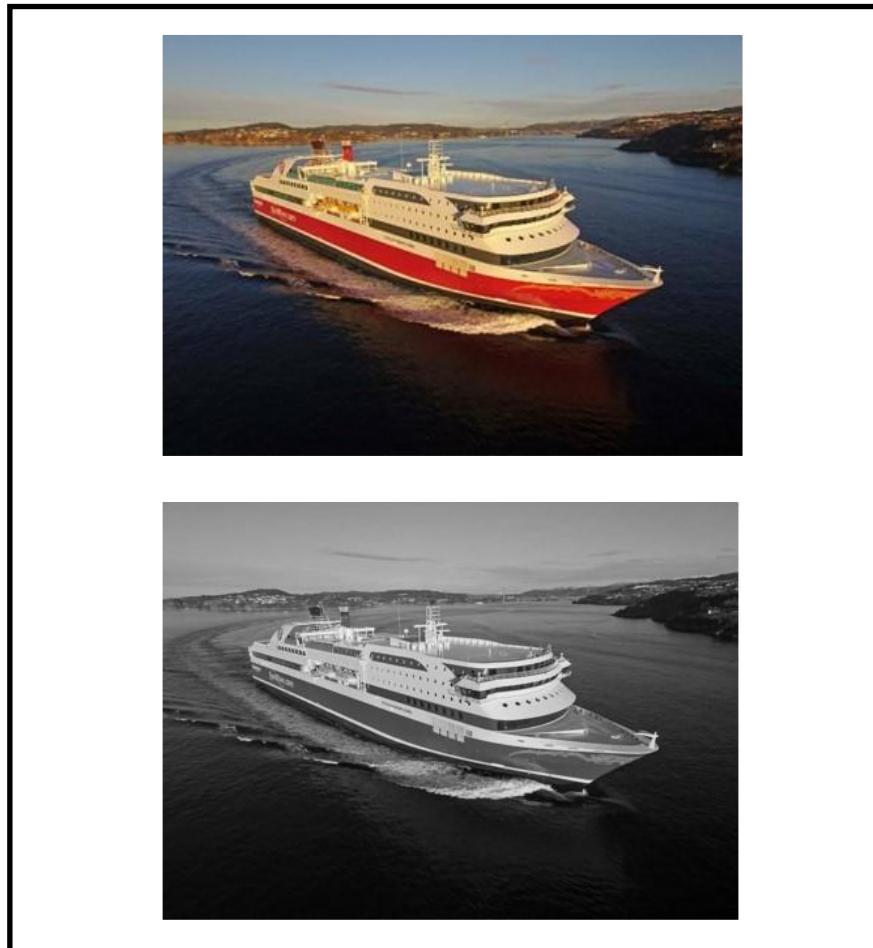
```
26 def normalize_data(x, y, height, width, number_channels, num_classes):
27     x_array = np.array(x)
28     x_array = x_array.reshape(x_array.shape[0], height, width, number_channels)
29     x_array = x_array.astype('float32')
30     x_array = x_array / 255
31     y = keras.utils.to_categorical(y, num_classes)
32     return x_array, y
```

Figure 31. Python code of the normalization process

### 4.3.3 Image Conversion to Grayscale

Converting colorful images to grayscale simplifies the learning task and reduces computational requirements. Grayscale images do not have color information. This makes them computationally more effective for training. However, this process may result in the loss of color-

related features that could be beneficial for some complicated maritime tasks (Saravanan, 2010). The decision to use grayscale or color images depends on each specific maritime scenario according to the complexity of the task. Figure 32 describes an example RGB image with its corresponding grey scale.



*Figure 32. RGB image with its greyscale*

#### **4.4 Single-based Solution**

After preprocessing, the images are ready to be injected into the network. CNNs are a type of DL model commonly used for various tasks, such as image classification. They consist of different layers that enable them to efficiently address such problems by extracting features and classifying images. The effectiveness of feature extraction depends on the architecture of each CNN model. In this section, we have developed five single simple models based on different CNN

architectures. In the subsequent descriptions, we will detail each model's architecture. For Model One, although it has a significantly large number of weights, to avoid redundancy, the weight calculations will not be repeated for the other four models.

## 1. First model

The first model, as shown in Figure 33, consists of various layers including two convolutional layers, and a max-pooling layer. That is subsequently flattened from two dimensions into one, followed by two FCLs. Figure 34 presents the Python code for Model One. The model starts with two convolutional layers. Each applies filters to capture features from the images. ReLU activation is used for introducing non-linearity (Agarap, 2018). The model then uses a max-pooling layer to reduce the spatial dimensions of the feature maps. It helps simplify the network's complexity. A flattening step follows, converting the two-dimensional feature maps into a one-dimensional vector. By doing so, it makes ready for the subsequent dense layers. High-level reasoning within the network is handled by two dense layers. The first is used as a FCL with the RELU activation and the second one uses softmax to categorize images into classes (Liu et al., 2016). The learning process of the model is managed by the categorical crossentropy loss function (Z. Zhang & Sabuncu, 2018). The next parameter is an optimizer which adadelta is used (Zeiler, 2012). The primary focus is on maximizing the accuracy metric. In the following, the weights for the first model are calculated.

The first CNN model's weights are  $18\ 720 + 524\ 288 \times (W-4) \times (H-4) + 256 \times n$  to be optimized, where W and H are the resolution of the input image, and n is the number of classes. For proofing the number of weights of the first CNN model, the below computation was done:

- First Convolution Layer: The number of weights of the first convolution layer is  $32 \times 3 \times 3$  since we have 32 filters, each of which contains  $3 \times 3$  weights to be optimized. After the first convolution layer, 32 feature maps are generated each of which has  $(W-2) \times (H-2)$  pixels.
- Second Convolution Layer: The number of weights of the second convolution layer is  $32 \times 64 \times 3 \times 3$  since we have 32 feature maps, and 64 filters, each of which



contains  $3 \times 3$  weights to be optimized. After the second convolution layer,  $64 \times 32$  feature maps are generated each of which has  $(W-4) \times (H-4)$  pixels.

- Maxpooling Layer: There is no weight to be optimized in this layer.
- Flatten Layer: The  $64 \times 32$  feature maps created by the second convolution layer are transformed into a set of neurons. Since each feature map contains  $(W-4) \times (H-4)$  pixels, the number of neurons generated is  $64 \times 32 \times (W-4) \times (H-4)$ .
- First FCL:  $64 \times 32 \times (W-4) \times (H-4)$  neurons already generated by the flatten layer will be connected to 256 neurons. Therefore, the number of weights in the first FCL is  $64 \times 32 \times (W-4) \times (H-4) \times 256$  weights to be optimized.
- Second FCL: 256 neurons will be connected to  $n$  neurons. Therefore, the number of neurons in the second FCL is  $256 \times n$  weights to be optimized.

Consequently, the number of weights in the first CNN model is:

$$(32 \times 3 \times 3) + (64 \times 32 \times 3 \times 3) + (64 \times 32 \times (W-4) \times (H-4) \times 256) + (256 \times n) = 18720 + 524 288 \times (W-4) \times (H-4) + 256 \times n \text{ weights to be optimized.}$$

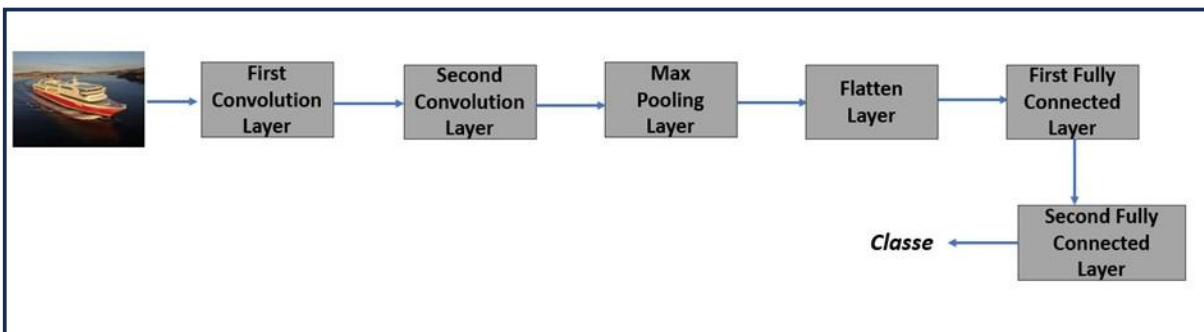


Figure 33. Architecture Design of the first CNN model

```

9 def CNN_first_model(num_classes, input_shape):
10     model = Sequential()
11     model.add(Conv2D(filters=32,
12                     kernel_size=(3, 3),
13                     activation='relu',
14                     input_shape=input_shape
15                     ))
16
17     model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
18     model.add(MaxPooling2D(pool_size=(2, 2)))
19     model.add(Flatten())
20     model.add(Dense(units=256, activation='relu'))
21     model.add(Dense(num_classes, activation='softmax'))
22     model.compile(loss=keras.losses.categorical_crossentropy,
23                 optimizer=keras.optimizers.Adadelta(),
24                 metrics=['accuracy'])
25
26     return model

```

Figure 34. Source Code of the first CNN model

## 2. Second Model

As shown in Figure 35, the Python code describes the CNN architecture. It begins with a max pooling layer applied directly to the input for initial spatial reduction. It follows a pattern of alternating between convolutional layers with 64 filters of 3x3 size using ReLU activation for feature extraction and max pooling layers for downsampling (Agarap, 2018). After passing through these layers, the data is flattened into a single vector, which is then processed by a dense layer with 64 units and ReLU activation before reaching the final dense layer that determines the class probabilities for the given number of classes using softmax activation (Liu et al., 2016). The model is compiled with categorical crossentropy to measure loss, an Adadelta optimizer to adjust weights during learning, and tracks accuracy as the performance metric (Zeiler, 2012; Z. Zhang & Sabuncu, 2018). This architecture is structured to efficiently handle image data, progressively reducing dimensionality while capturing and refining features essential for classification.

```

30 def CNN_second_model(num_classes, input_shape):
31     model = Sequential()
32     model.add(MaxPooling2D(pool_size=(2, 2), input_shape=input_shape))
33     model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
34     model.add(MaxPooling2D((2, 2)))
35     model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
36     model.add(Flatten())
37     model.add(Dense(units=64, activation='relu'))
38     model.add(Dense(num_classes))
39     model.compile(loss=keras.losses.categorical_crossentropy,
40                 optimizer=keras.optimizers.Adadelta(),
41                 metrics=['accuracy'])
42     return model

```

Figure 35. Source Code of the second CNN model

### 3. Third Model

The third model is a DL model which is coded in Figure 36. It put together several convolutional layers that progressively double the number of 3x3 filters from 32 up to 1024. Each layer enhances the model's ability to recognize complex patterns through ReLU activation (Agarap, 2018). Following the extraction of detailed features, a single max pooling operation condenses the data, which is then linearised by a flatten operation. Subsequently, a 256-unit dense layer upgrades the data further. Finally, the output layer maps these refined features to the number of classes using a softmax function (Liu et al., 2016). The model is compiled with categorical crossentropy to measure loss. It uses an Adadelta optimizer to adjust weights during learning. It also tracks accuracy as the performance metric (Zeiler, 2012; Z. Zhang & Sabuncu, 2018). The targeting accuracy as the measure of its predictive power is used. Unlike previous models, this one emphasizes depth and feature scaling, potentially capturing more intricate image details.

```

45 def CNN_third_model(num_classes, input_shape):
46     model = Sequential()
47     model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
48                     input_shape=input_shape))
49     model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
50     model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
51     model.add(Conv2D(filters=256, kernel_size=(3, 3), activation='relu'))
52     model.add(Conv2D(filters=512, kernel_size=(3, 3), activation='relu'))
53     model.add(Conv2D(filters=1024, kernel_size=(3, 3), activation='relu'))
54     model.add(MaxPooling2D(pool_size=(2, 2)))
55     model.add(Flatten())
56     model.add(Dense(units=256, activation='relu'))
57     model.add(Dense(num_classes, activation='softmax'))
58     model.compile(loss=keras.losses.categorical_crossentropy,
59                 optimizer=keras.optimizers.Adadelta(),
60                 metrics=['accuracy'])
61     return model

```

Figure 36. Source Code of the third CNN model

#### 4. Fourth Model

Model four is a CNN architected in which the Python code is written in Figure 36. Initiating with a 32-filter convolutional layer, it applies ReLU to introduce non-linear processing right at the outset (Agarap, 2018). Unique to this model is a series of four max pooling layers with increasing window sizes, from 1x1 to 4x4, each progressively diminishing the output dimensions in a distinctive manner. Post-pooling, the flattened data passes through two dense layers, with 128 and 256 nodes respectively, each continuing the use of ReLU activation to further refine the data. The sequence culminates in a softmax-activated layer that maps the output to the number of classes (Liu et al., 2016). Compiled to minimize categorical crossentropy loss and optimize with Adadelta, the model sets its sights on optimizing for accuracy, marking its readiness to classify images with a nuanced understanding of their features (Zeiler, 2012; Z. Zhang & Sabuncu, 2018).

```

64 def CNN_fourth_model(num_classes, input_shape):
65     model = Sequential()
66     model.add(Conv2D(filters=32, kernel_size=(3, 3),
67                     activation='relu',
68                     input_shape=input_shape))
69     model.add(MaxPooling2D(pool_size=(1, 1)))
70     model.add(MaxPooling2D(pool_size=(2, 2)))
71     model.add(MaxPooling2D(pool_size=(3, 3)))
72     model.add(MaxPooling2D(pool_size=(4, 4)))
73     model.add(Flatten())
74     model.add(Dense(units=128, activation='relu'))
75     model.add(Dense(units=256, activation='relu'))
76     model.add(Dense(num_classes, activation='softmax'))
77     model.compile(loss=keras.losses.categorical_crossentropy,
78                 optimizer=keras.optimizers.Adadelta(),
79                 metrics=['accuracy'])
80     return model

```

Figure 37. Source Code of the fourth CNN model

## 5. Fifth Model

As shown in Figure 37, model five is a CNN model which consists of different layers. initial layer that applies 32 distinct 3x3 filters to the input images using ReLU activation for added depth in feature detection (Agarap, 2018). What follows is a quartet of max pooling layers with progressively larger windows, each methodically reducing the feature map size to streamline the data. The model then transitions this streamlined data through a flattening process, making it ready for a series of three dense layers. These dense layers increase in capacity, from 128 to 512 neurons, each layer applying ReLU to ensure complex feature interactions are captured. Culminating in a softmax layer, the model assigns probabilities to various class outcomes (Liu et al., 2016). The entire network is tuned with a categorical crossentropy loss function and the Adadelta optimizer, all the while homing in on accuracy as the key performance indicator (Zeiler, 2012; Z. Zhang & Sabuncu, 2018).

```

83 def CNN_fifth_model(num_classes, input_shape):
84     model = Sequential()
85     model.add(Conv2D(filters=32, kernel_size=(3, 3),
86                     activation='relu',
87                     input_shape=input_shape))
88     model.add(MaxPooling2D(pool_size=(1, 1)))
89     model.add(MaxPooling2D(pool_size=(2, 2)))
90     model.add(MaxPooling2D(pool_size=(3, 3)))
91     model.add(MaxPooling2D(pool_size=(4, 4)))
92     model.add(Flatten())
93     model.add(Dense(units=128, activation='relu'))
94     model.add(Dense(units=256, activation='relu'))
95     model.add(Dense(units=512, activation='relu'))
96     model.add(Dense(num_classes, activation='softmax'))
97     model.compile(loss=keras.losses.categorical_crossentropy,
98                 optimizer=keras.optimizers.Adadelta(),
99                 metrics=['accuracy'])
100    return model

```

Figure 38. Source Code of the fifth CNN model

#### 4.5 Maritime mAnaGement eNsemble leArning sysTem (MAGNAT)

A comprehensive depiction of the Maritime mAnaGement eNsemble leArning sysTem (MAGNAT) is shown in **Error! Reference source not found.** This is an innovative system designed for maritime management. To initiate this process, the maritime authority collects historical ship operation images. These images form the basis of a robust ship image database. The database provides the training ground for five CNN models, as detailed earlier. These models are trained by using the ship image database. This can improve the model's ability to recognize and analyze maritime patterns.

Within the aggregation process, CNN models work together. The local output of each model comes together to create a single global output. Then, a drone is used across the port. Diverse ship images from multiple angles can be captured with the drone. Using the pre-trained ensemble model, these images undertake dedicated analysis.

Within a detailed report, the analytical findings are compiled. This report is quickly sent to a nominated analyzer. The analyzer is really important. It processes this information and creates

situation reports. If any abnormal maritime behaviors are identified, alerts are generated. These alerts inform the relevant authorities.

In the following, the detailed components of the developed aggregation function are explored. This illuminates the methodology employed to merge the local outputs from each model. This not only improves the overall efficacy of the MAGNAT system. It also ensures a proactive response to potential abnormalities in maritime activities.

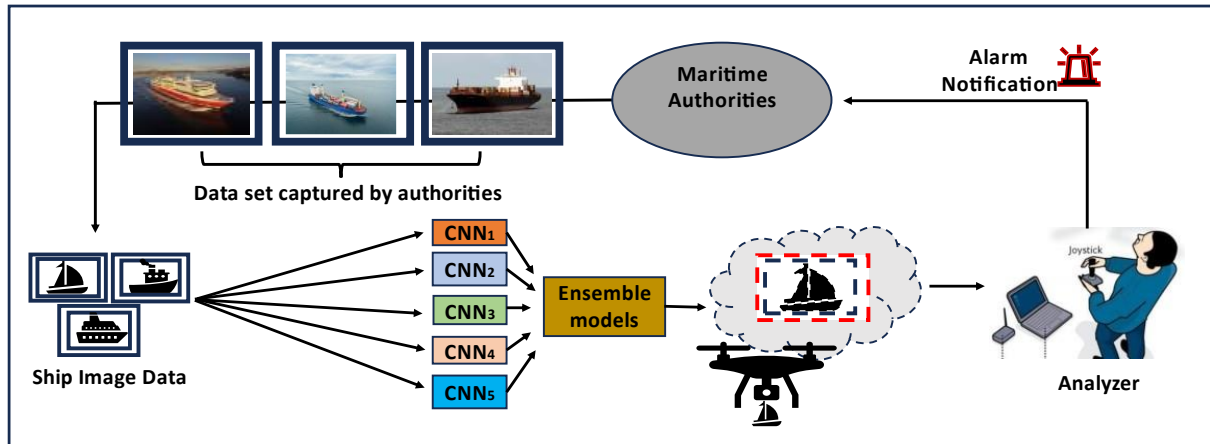


Figure 39. Ensemble model Framework

#### 4.5.1 Ensemble Aggregation

Each CNN model from the five models already described will independently process the input ship image. It captures unique features and patterns present in the image. The probability distribution output by each model is normalized to make comparable outputs, ensuring each model's contribution is proportionate. Weight factors are assigned to normalized probabilities based on historical performance or reliability. In this context, we will give more influence on models with higher accuracy or reliability in the past, improving overall ensemble performance. then we calculate the weighted average of normalized probabilities for each class across all models to aggregate predictions, considering the relative importance of each model, producing a more robust decision. We will then establish a threshold value for the final decision to control decision-making, ensuring only confident predictions contribute to the final classification. Afterward, we select the class with the highest weighted average probability (or surpassing the threshold) as the final decision. This allows to provide a clear and definitive output based on the ensemble's collective prediction. Optionally calculates a confidence score by summing weighted

average probabilities across all classes. We will also consider periodically updating weights assigned to each model based on recent performance. This allows the ensemble to adapt to changes in models' effectiveness over time, ensuring continued optimal performance.

#### 4.6 Maritime mAnaGement eNsemble prUning sysTem (MAGNUT)

A complete overview of Maritime mAnaGement eNsemble prUning sysTem (MAGNUT) is depicted on Figure 40. It is an innovative system designed for effective maritime management. Similar to MAGNAT, the process starts with the maritime authority collecting and archiving historical ship operation images. As mentioned earlier, his database serves as the training platform for five CNN models. These models get better at detecting and analyzing maritime patterns through training with the ship image database. The collaboration of these CNN models is achieved through an ensemble pruning process, where the local output of each model is combined to produce a cohesive global output. The configuration of each model is saved to a knowledge base containing model specifications. The ensemble pruner uses the information in the knowledge to select the most relevant models in this context, we should keep the most general models and remove the redundant ones. In the following, we will provide a detailed explanation of the MAGNUT framework, as illustrated in Figure 40.

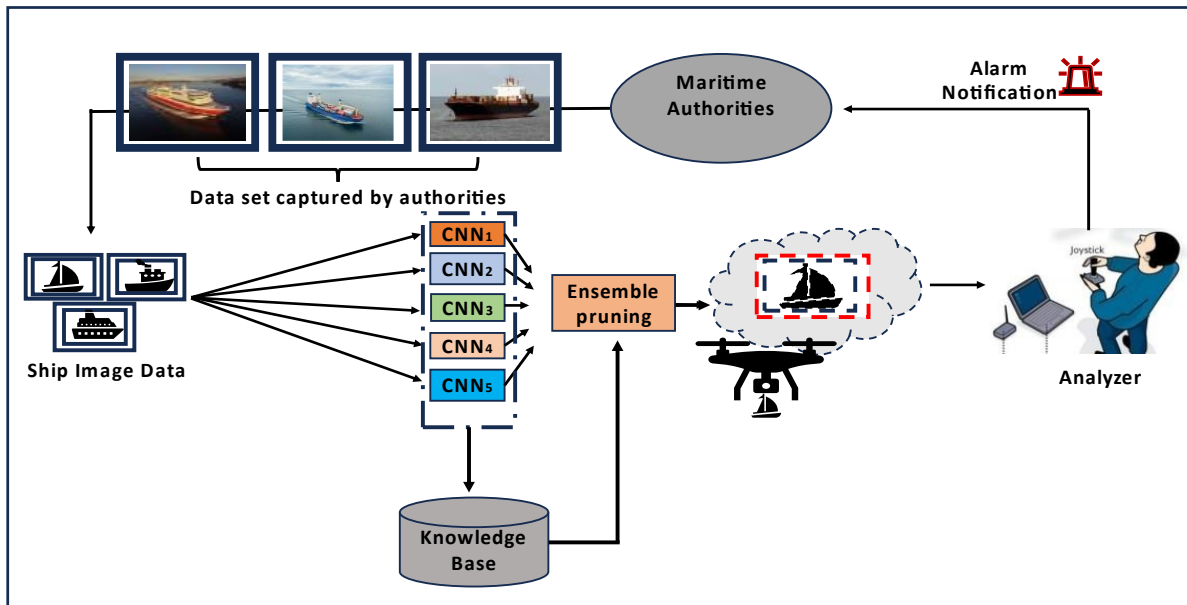


Figure 40. Ensemble Pruning Framework

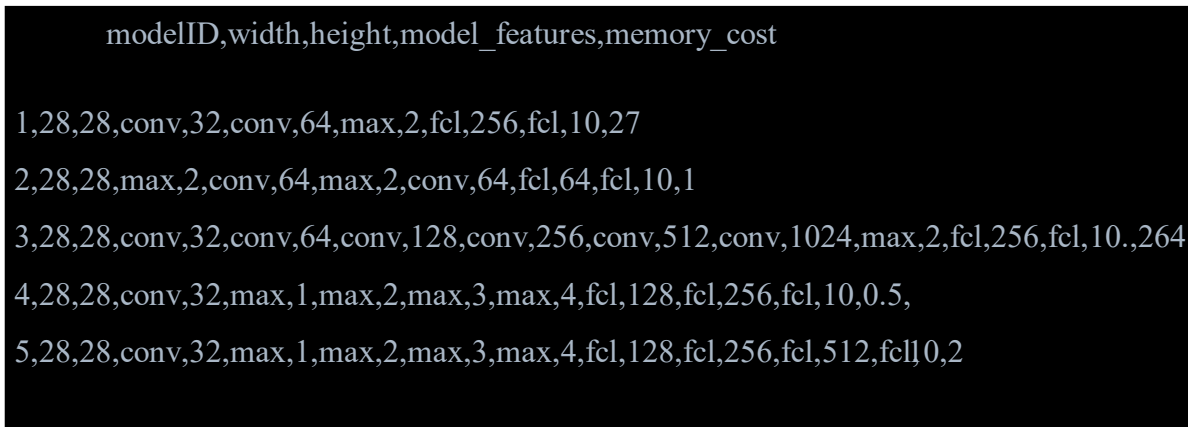


### 4.6.1 Knowledge Base Creation

In this subsection, the characteristics of each model in a knowledge base will be saved. These characteristics include the convolution layers with their filters, the maxpooling layers with their strides, and the FCLs with their connected neurons. For instance, the first model consists of two convolution layers with 32, and 64 filters, respectively, one maxpooling layer with (2x2) stride, and two FCLs with 256, and n neurons, respectively. With 28 x 28 image resolution as input, and n set to 10, the knowledge base should contain the following row describing the characteristics of the first model:

1, 28, 28, conv, 32, conv, 64, max, 2, fcl, 256, fcl, 10, 27

1 is used to describe that we are talking about the first model, conv for the convolution layer, max for maxpooling layer, and fcl for the FCL. The last item describes the model cost in terms of memory size. The knowledge base for the five models is illustrated in Figure 41.



modelID	width	height	model_features				memory_cost														
1	28	28	conv	32	conv	64	max	2	fcl	256	fcl	10	27								
2	28	28	max	2	conv	64	max	2	conv	64	fcl	64	fcl	10	1						
3	28	28	conv	32	conv	64	conv	128	conv	256	conv	512	conv	1024	max	2	fcl	256	fcl	10	264
4	28	28	conv	32	max	1	max	2	max	3	max	4	fcl	128	fcl	256	fcl	10	0.5		
5	28	28	conv	32	max	1	max	2	max	3	max	4	fcl	128	fcl	256	fcl	512	fcl	0	2

Figure 41. Knowledge Base Created for the Five Models

### 4.6.2 Intelligent Aggregation:

This subsection aims to introduce a novel approach for intelligent aggregation of the five CNN models previously designed. Unlike existing solutions, MAGNUT proposes a set of non-redundant models within the ensemble for intelligent aggregation. The proposed intelligent aggregation method leverages a knowledge-driven pruning process as indicated in Figure 40. This process aims to discover a minimal subset of non-redundant models. Then the voting mechanism to determine the final output is applied. This approach enhances the efficiency and effectiveness

of ensemble learning for CNN models. Further details of this method will be explained in the following subsections, including “model selection and redundancy elimination,” “greedy search algorithm and model search tree,” “minimal subset determination,” and “voting mechanism and final output.” An illustrative example will also be presented below.

#### **a) Model Selection And Redundancy Elimination**

The set of five CNN models is denoted as “ $M = \{\text{CNN1}, \text{CNN2}, \text{CNN3}, \text{CNN4}, \text{CNN5}\}$ .” A subset, “ $M^*$ ” is created to identify non-redundant models. It comprises representative models from each unique configuration in “ $M$ .” This process involves leveraging the established knowledge base to prune irrelevant models. Specifically, the specifications of all models in “ $M$ ” are explored, and “ $M^*$ ” is defined as the minimal subset that maximizes the inclusion of specified layers with identical parameter characteristics. For example, if the specified layer is a convolutional layer, “ $M^*$ ” will be determined by the minimal subset containing convolutional layers with the same number of filters. The objective is to pinpoint this subset  $M^*$  that captures the essence of the ensemble.

#### **b) Greedy Search Algorithm And Model Search Tree**

The described approach employs a greedy search algorithm to construct a model search tree. This algorithm iteratively explores a knowledge base to gather specifications for all models in the set “ $M$ .” During the tree-building process, models with identical specifications to others but fewer instances of specific layers are eliminated, streamlining the search space. The algorithm operates iteratively, considering all possible subsets of models in “ $M$ .” This strategy aims to efficiently navigate the space of model configurations, identifying optimal or satisfactory solutions by iterative refining and exploring different combinations of models (de França, 2018).

#### **c) Minimal Subset Determination**

Following the application of the greedy search algorithm, the result is a collection of potential subsets, each denoted as a candidate “ $M^*$ .” The objective is to identify the minimal subset, designated as “ $M^*$ ,” from these candidates. The selection of “ $M^*$ ” is determined by considering the subset with the fewest number of models while still retaining the maximum number of specified layers that share identical parameters. In essence, the algorithm aims to find the most compact subset, “ $M^*$ ,” that encapsulates the essential layer specifications common

across multiple models. The criteria for selection involve minimizing the number of individual models in the subset while maximizing the inclusion of layers that have identical parameters. This approach is likely geared towards promoting simplicity and efficiency in the resulting subset, focusing on the most representative and efficient model configurations for the given task.

#### d) **Voting Mechanism And Final Output**

Upon identification of the minimal subset “M\*,” a subsequent step involves applying a voting mechanism, reminiscent of the MAGNAT approach. This mechanism serves the purpose of determining the ultimate output. In this context, each model within the minimal subset actively contributes to the final decision. The aggregation process that follows is designed to enhance both the robustness and accuracy of the outcome. The voting mechanism involves soliciting input from each model in the minimal subset, allowing them to express their opinion or predictions regarding the task at hand. This collective decision-making process helps mitigate the impact of potential abnormal model behavior or models that might exhibit less reliability in certain scenarios. By aggregating the contributions from multiple models, the approach aims to achieve a more robust and accurate final output. This aggregation process is crucial for leveraging the diverse insights provided by each model in the minimal subset. It enhances the overall reliability and performance of the final decision, contributing to a more comprehensive and dependable outcome in the context of the given task or problem.

- **Illustrative Example**

In **Error! Reference source not found.**, provides a detailed portrayal of the intelligent aggregation process. It sheds light on the intensive exploration of four models and the deliberate consideration of convolutional layers during the pruning procedure. This illustrative example improves comprehension of the complexities involved in our pruning methodology. The process begins by exploring individual models as potential candidates. Subsequently, models are iteratively added to the candidate pool using a recursive strategy. For instance, as shown in **Error! Reference source not found.**, Model 2 (M2) has 64 feature maps in its convolution layer, whereas Model 1 (M1) has 32 feature maps from a convolutional layer and an additional 64 feature maps from another convolution layer. This shows that “M1,” in terms of its convolution layer, is not similar to “M2,” and thus both need to be kept. However, when comparing Model 1 (M1) and Model 3 (M3), we observe that M1 has 32 and 64 feature maps from convolution layers, while

“M3” includes 32 and 64, as well as 128, 256, 512, and 1024 feature maps from its convolution layers. This suggests that in terms of convolutional features, “M1” is included within “M3” because the 32 and 64 feature maps present in “M1” also exist in “M3.” Therefore, we can remove “M1.” This procedure is replicated for all possible combinations of models, ensuring the exclusion of redundant candidates. The resulting model candidates are then evaluated based on their total cost based on memory usage. The calculated costs initiate a ranking process, organizing the model candidates based on their respective costs. The outcome of this ranking manifests as follows:

- M2 and M3 both share a cost of 19.5
- M1 and M2 both share a cost of 20.5
- M2a and M4 both share a cost of 26.9

Instead of exploring all  $2^4 = 16$  model candidates, this research focuses on just three non-redundant ones. This reduction strategy greatly simplifies the learning process. These three candidates are explored in detail to identify the optimal subset based on accuracy while observing memory size constraints. This intelligent selection process ensures an efficient model aggregation that meets performance criteria.

```

modelID,width,height,model_features,memory_cost
1,28,28,conv,32,conv,64,max,2,fcl,256,fcl10,7.1
2,28,28,max,2,conv,64,max,2,conv,64,fcl,10,13.4
3,28,28,conv,32,conv,64,conv,128,conv,256,conv,512,conv,1024,max,2,fcl10, 6.1
4,28,28,conv,32,max,1,max,2,max,3,max,4,fcl,128,fcl,256,fcl10,13.5

```

Figure 42. Knowledge base creation for the search of tree exploration

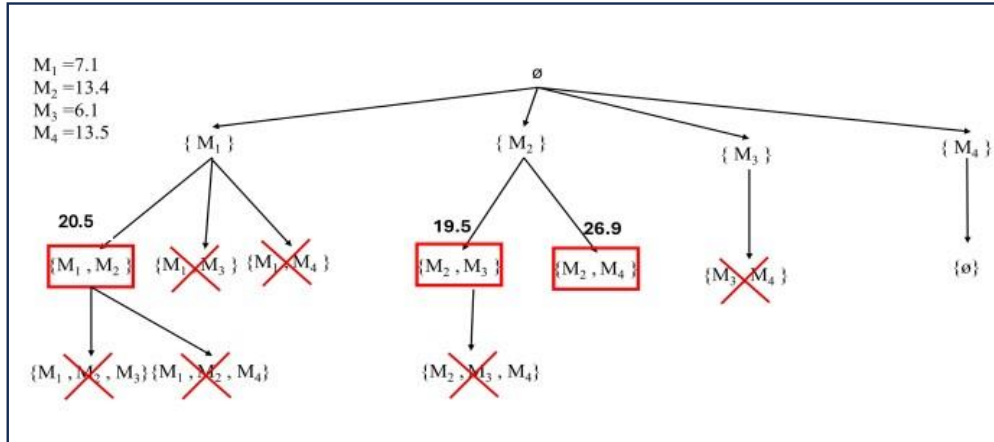


Figure 43. Example of search of tree exploration

## 4.7 Challenges And Discussions

The designed MAGNAT and MAGNUT systems will bring several benefits to maritime management applications. MAGNAT and MAGNUT will promote innovation in autonomous surveillance technology by investing in research and development to enhance the capabilities of surveillance drones and integrate them into maritime infrastructure that can contribute to sustainable development. MAGNAT and MAGNUT can enhance maritime security, promote the rule of law at sea, and facilitate the prosecution of illegal activities.

It can also help in protecting coastal communities by ensuring the safety of ships entering and leaving ports. These designed systems will help in a transformative era in maritime operations. It promises substantial advantages, including expanded coverage, real-time data acquisition, and a notable reduction in human risk. However, this innovative approach also introduces a distinct set of safety and security concerns that necessitate careful consideration and the development of effective mitigation strategies. Safety-wise, the risk of collisions, be it due to technical problems, navigation errors, or unexpected obstacles, comes up as a serious issue.

Adverse weather conditions can impact flight stability and sensor performance. On the security front, the interconnected nature of autonomous drones exposes them to cybersecurity vulnerabilities and potential data breaches, especially as they handle sensitive information. Unauthorized access or interference by malicious actors is another concern, as is safeguarding the physical security of these drones. Navigating these safety and security challenges is essential to

harness the full potential of the designed system while ensuring the safety and confidentiality of maritime operations.

Even MAGNUT provides an intelligent strategy to prune the models of the MAGNAT system, determining the specific model features to be stored in the knowledge base for the MAGNUT system is a nuanced task that lacks straightforwardness. Taking the convolution layer case as an example, the consideration of the number of filters alone may not suffice for a comprehensive exploration of the model. It prompts the question of whether more granular features, such as the kernel size or activation function, are indispensable for thorough exploration. To make informed decisions about which model features are relevant to retain in the knowledge base, a careful and detailed analysis becomes important. It necessitates an in-depth investigation into the intricacies of various features associated with the model. For instance, understanding the impact of factors beyond just the number of filters, such as the specific characteristics of each filter like kernel size and activation function, becomes crucial. The complexity of model architectures requires thoughtful consideration of not only the broad characteristics but also the fine-grained details that contribute to the model's behaviour. Hence, a comprehensive exploration of model features is essential to discern and subsequently store the relevant information in the knowledge base of the MAGNUT system. This approach ensures that the knowledge base captures the nuanced details critical for optimizing the system's performance and decision-making processes.

#### **4.8 Conclusion**

In this chapter, the MAGNAT and MAGNUT frameworks are proposed as advanced solutions for maritime management system. The first solution of this research, MAGNAT, addresses challenges in ensemble learning by enhancing model selection and performance optimization. MAGNAT utilizes ensemble learning to propose intelligent aggregation by assigning weight factors based on the historical performance of the models. With this strategy, models that positively contribute can assign more weight, influencing the overall performance through aggregation. MAGNUT focuses on pruning non-contributory elements of the ensemble models, thereby enhancing model performance, improving computational efficiency, and enabling real-time scalability. These strategies highlight the potential of AI and DL in

modernizing maritime management, emphasizing the importance of smart and adaptable systems. In the last part, each challenge of each proposed strategy was discussed.

# Chapter 5: Experiments

## 5.1 Introduction

This chapter explores the performance evaluation of proposed strategies like MAGNAT and MAGNUT. It examines their effectiveness with specialized datasets such as fish and ship classification. The first part provides comprehensive information about our datasets. Furthermore, the chapter also goes deeper into model visualization using Netron as a tool for this purpose. It is not just a technical presentation but also it has insights about why we visualize DL models as a way of understanding and optimizing them better. The heart of this chapter focuses on evaluating model performance that includes a detailed assessment of proposed strategies. Different challenges that the author faced due to the dataset are explained as well. In the further results part of this chapter, five advanced models of architecture are proposed to add robustness to the experimentation. By demonstrating the feasibility of employing an advanced model, the research shows the potential applications of this methodology across various model architectures. To visualize the situation more realistically, after evaluating each advanced model on existing datasets, the summary is conducted.

## 5.2 Dataset Description

To demonstrate the success of proposed strategies, such as MAGNAT and MAGNUT, applicable across various domains, two datasets are selected: jellyfish and ships. The jellyfish dataset demonstrates the possibility of using this idea in marine science. The ship dataset is selected to show that this concept can be applied in maritime technology as well.

### 5.2.1 Jellyfish Dataset

A jellyfish dataset is selected to classify different types of fish. This dataset is extracted from the Kaggle website, which consists of nine classes (Marine Animal Images, n.d.). This dataset has been divided into two subsets: training and testing datasets. The training dataset has been used for training our models, and the testing dataset was useful for evaluating our models. The training dataset consists of 621 images, and the testing dataset includes 185 images (Marine Animal Images, n.d.). Figure 44 represents the jellyfish classes. These classes are as follows: Fish, Goldfish, Harbor Seal, Jellyfish, Lobster, Oyster, Sea Turtle, Squid, and Starfish. Each class



contributes to the overall richness and complexity of the dataset, ensuring that the models can distinguish between different fish categories.



Figure 44. Dataset illustration on different fish classes taken from (Marine Animal Images, n.d.)

### 5.2.2 Ship Dataset

Another dataset, the ship dataset, is chosen from the Kaggle website to validate the results with an additional dataset (JAIN, 2019; Kaggle, n.d.). As shown in Figure 45, the dataset includes five different classes of ships: “Cargo ship,” “Military ship,” “Carrier ship,” “Cruise ship,” and “Tanker ship.” It is downloaded from the Kaggle website and contains 6252 images in the training set and 2680 images in the test set (JAIN, 2019).



Figure 45. Ship dataset

### 5.3 Model Visualizations

Netron tool is used to visualize and explore how neural networks are made up (Roeder, n.d.). Netron is a user-friendly open-source alternative for displaying DL models (Figure 46). This tool functions with different frameworks such as TensorFlow, Keras, and PyTorch (Chollet, 2015; Paszke et al., 2019). With this interactive interface of Netron, we can study the layers, connections, and parameters that make up a neural network leading to a better understanding of its operations. Additionally, Netron is accessible from various platforms which makes it flexible enough to help both researchers and practitioners in the ML field. Among other issues, Netron distinguishes itself by having a wide variety of users across different operating systems like Windows, macOS, or Linux. It also has an added advantage on local deployment where sensitive models can be visualized offline without access to the internet or exposure to external servers. After installation, Netron is available offline making it very convenient for individuals who have slow internet connections. Lastly, Netron provides detailed descriptions for each layer of a neural network containing information about input shapes/output shapes including parameter details plus the number of parameters for every layer. This profound exploration enables one to understand more about the setup and operation mechanism behind the model as well as why it behaves in a certain manner. In our pipeline for saving models after training them, we use “h5”. In Figure 47, architectures of the five proposed models were visualized using Netron.



Figure 46. Neutron Software (Roeder, n.d.)

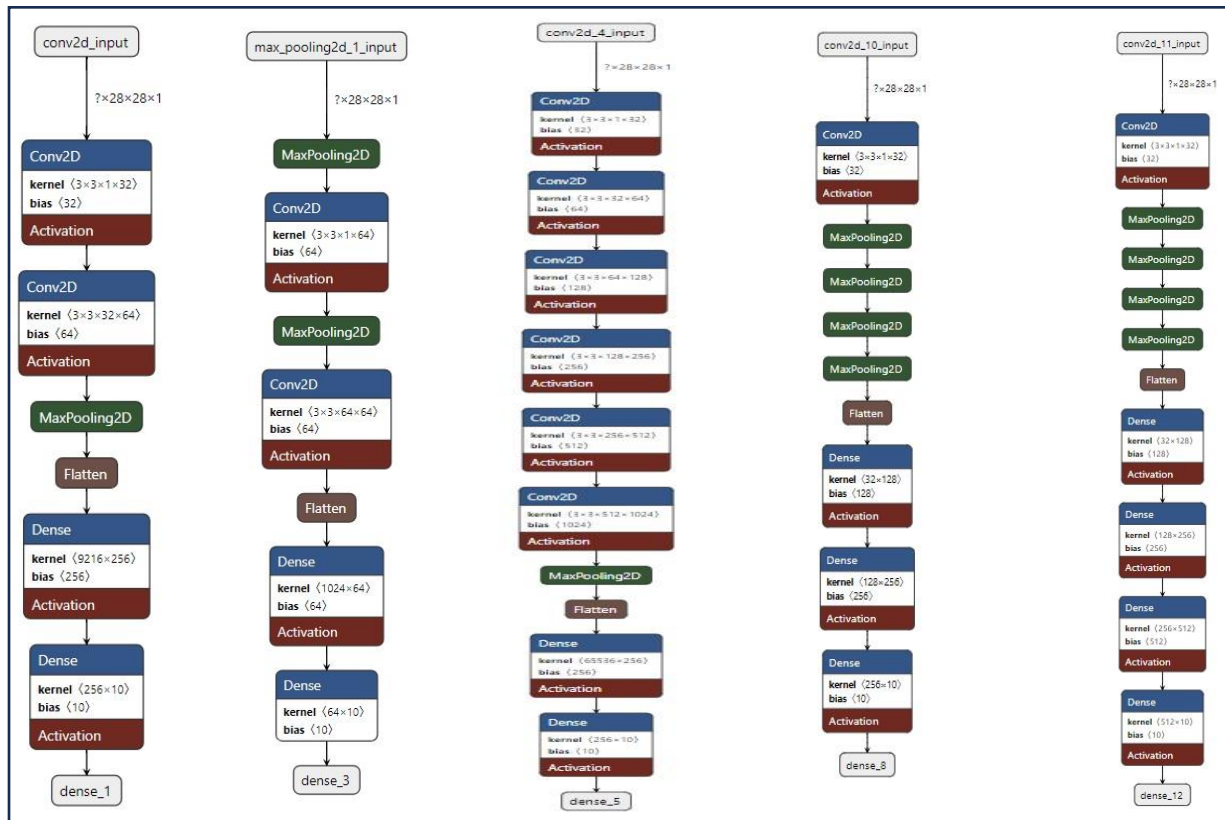


Figure 47. Model visualization using Neutron (from the left to right, Model-1, Model-2, Model-3, Model-4, Model-5)

## 5.4 Performance Evaluation

In this section, the jellyfish dataset was evaluated by our proposed models to demonstrate how these models can be applied in maritime science. After obtaining results from this dataset, due to the contextual problem of the dataset, we not only replaced the five proposed models with the VGG16 model to evaluate the issue of low accuracy in the model's architecture, but we also replaced our dataset with the MNIST dataset and ship dataset. Through this strategy, we validated

our claim regarding the challenging nature of the jellyfish dataset, and we also demonstrated how these strategies can be applied to other marine applications such as maritime technology.

The results in this section are presented in tables. Each table includes the accuracy of individual models after 50 epochs of training. It also shows each model's accuracy with unseen images in the testing dataset. Additionally, tables show the memory cost consumed by the models. In the last 2 columns of the tables except for the VGG16 result table, the result of the proposed strategies such as MAGNAT and MAGNUT is attached.

#### 5.4.1 Evaluation of the models on the Jellyfish dataset.

After experimenting with various image sizes from the jellyfish dataset, the image size of 128x128 was chosen due to its higher accuracy compared to other sizes (Table 1). The overall accuracy of models using an image size of 28x28 is 25%. For 32x32, it is 48%, for 64x64, it is 65%, and for 128x128, it is 72%, the highest among all tested sizes.

*Table 1. Accuracy performance with different shape sizes of jellyfish images*

Shape size	Accuracy
(28, 28)	25.0
(32, 32)	48.0
(64, 64)	65.0
(128, 128)	72.0

Five model architectures were proposed, and the jellyfish dataset was used to train 5 models. However, upon testing with the unseen test dataset, the achieved accuracies were found to be unsatisfactory. For instance, in Table 1, Model 1 was trained with an accuracy of 53%, and the accuracy achieved in tests with the test dataset was 36%. Model 2 learned with an accuracy of 14% from the dataset, and the test accuracy was 19%. Model 3, due to its architecture's high computational complexity, was challenging and could not be computed on a conventional computer. The model has been used in this research to demonstrate how the complexity of a model's architecture can impact outcomes and to emphasize the importance of designing models that have not only lower computational complexity but also the ability to achieve good accuracy. During training, Model 4 was able to reach an accuracy of 23% but managed only 13% accuracy in the tests. Model 5 attained an accuracy of 24% during training and similarly could only achieve 13% accuracy in the tests. The ensemble method resulted in an accuracy of 26%. This accuracy

is expected due to the negative contribution of models that had lower accuracy; they decreased the ensemble learning accuracy. For pruning with the proposed strategy, the first step was to keep the more general models according to their architectural design and prune the redundant ones. Four subsets remained: (M2, M4), (M2, M5), (M1, M2), and (M2, M3). Then, the memory cost for each subset was calculated, and the models were prioritized based on it (Table 2). After calculation, (M2, M4) with a size of 41 MB had the lowest memory cost and became our first priority. Next, (M2, M5) with a size of 43 MB became our second priority. The third priority was (M1, M2) with a size of 778 MB. Due to the high memory cost of model 3, the fourth one was impossible to calculate. The overall low accuracies across models and strategies suggested potential issues with model architectures or dataset suitability.

Table 2. Accuracy of models on jellyfish dataset with epoch 50 with 128x128 image size, and 132 batch size

Model/Accuracy	Model_1	Model_2	Model_3	Model_4	Model_5	Ensemble Learning	Ensemble Pruning			
							M2,M4	M2,M5	M1,M2	M2,M3
<b>Train- Accuracy</b>	53%	14%	-	23%	24%	-	-	-	-	-
<b>Test Accuracy</b>	36%	19%	-	13%	13%	26%	11%	13%	32.4%	-
<b>Memory_cost(MB)</b>	738	40	-	1	3	782	41,	43	778	-

- **Replacing models with VGG16 model**

To find the reason for low accuracy, the models were first examined. The 5 designed models were replaced with the VGG16 model, which is one of the popular models in image classification (Luu & Anh, 2023). As shown in Table 2, the results were not satisfactory. According to the table, the accuracy of the model in training was 53%, and the test accuracy was 36%. The results remained low even with the VGG16 model. Thus, it was understood that the problem was not due to the architecture of our models.

Table 3. Accuracy of the VGG16 Model on the Jellyfish Dataset (Epoch: 50, Image Size: 224x224, Batch Size: 132)

Models/accuracy	VGG16
<b>Train accuracy</b>	53%
<b>Test accuracy</b>	36%

- **Replacing jellyfish dataset with MNIST dataset**

To validate our dataset, the MNIST dataset was replaced with the jellyfish dataset. This dataset is widely used for image classification (Cohen et al., 2017). MNIST dataset consists of

60,000 examples in its training set and 10,000 in its test set, all containing handwritten digits (Cohen et al., 2017). Table 4 shows models designed that are trained on the MNIST dataset. Model 1 demonstrated the highest accuracy, achieving 94% accuracy with a memory cost of 27 MB for both training and test sets. Conversely, model 4 showed the lowest accuracy across both sets, achieving only 28% accuracy in the test set, with a memory consumption of 0.5 MB. Model 3 did not succeed in training on both the jellyfish and MNIST datasets because it had a high memory cost which was not able to be computed by a normal computer. Keeping this model is a good example to show how the memory cost and model architecture are important. Despite its lower accuracy, the ensemble's overall accuracy of 47% exceeds that of individual models other than model 1. The presence of high-accuracy models like Model 1 in the ensemble had a positive impact, though it's countered by the presence of models with lower accuracy.

In ensemble pruning, following the MAGNUT strategy, four subsets remained as non-redundant and general subsets in terms of architectural design: (M2, M4), (M2, M5), (M1, M2), and (M2, M3). The subset (M2, M4) has the lowest memory cost and was selected as our first priority. (M2, M5) ranks second, with a memory cost of 27%. (M1, M2) follows as the third priority, with a memory cost of 28 MB and an accuracy of 74%, the highest after Model 1. Due to its high memory cost, (M2, M3) could not be calculated. Despite its slight difference in memory cost, (M1, M2) prove to be more robust and non-redundant, aligning with our strategy.

As a result, we got good accuracy compared to the jellyfish dataset which can support our claim about a challenging dataset that jellyfish have. This dataset has different contextual details. As shown in Figure 44, as an example, the image that shows the Harbor seal and the image that shows the picture of fish do not have the same such as clarity, background elements, lighting, or even angle. This result could be a good result for showing how much the dataset can affect training models.

Table 4. Accuracy Comparison of Five Models on the MNIST Dataset (Fixed Parameters: Epoch: 100, Batch Size: 32, Image Size: 28x28)

Model/Accuracy	Model_1	Model_2	Model_3	Model_4	Model_5	Ensemble Learning	Ensemble Pruning			
							M2,M4	M2,M5	M1,M2	M2,M3
<b>Train- Accuracy</b>	95%	11%	-	27%	33%	-	-	-	-	-
<b>Test Accuracy</b>	94%	11%	-	28%	31%	47%	25%	27%	74%	-
<b>Memory_cost(MB)</b>	27	1	-	0.5	2	294	1.5	3	28	-

- **Illustration of test from MNIST dataset**

To illustrate the quality of models, we conducted a test using a random selection of digits from the MNIST dataset. As shown in **Error! Reference source not found.**, there is an illustration of one of the model's predictions (in this example we use model 1) for a random digit. For example, in the image with the illustration of the number 4, the actual label is 4, and the predicted model is also 4, indicating that the predicted model is classified correctly. However, for the image depicting the number 8, the actual label is 8, but the model incorrectly identifies it as 5. In this instance, our model did not correctly recognize the number. Similar analyses are conducted for other randomly selected digits to assess the model's accuracy.

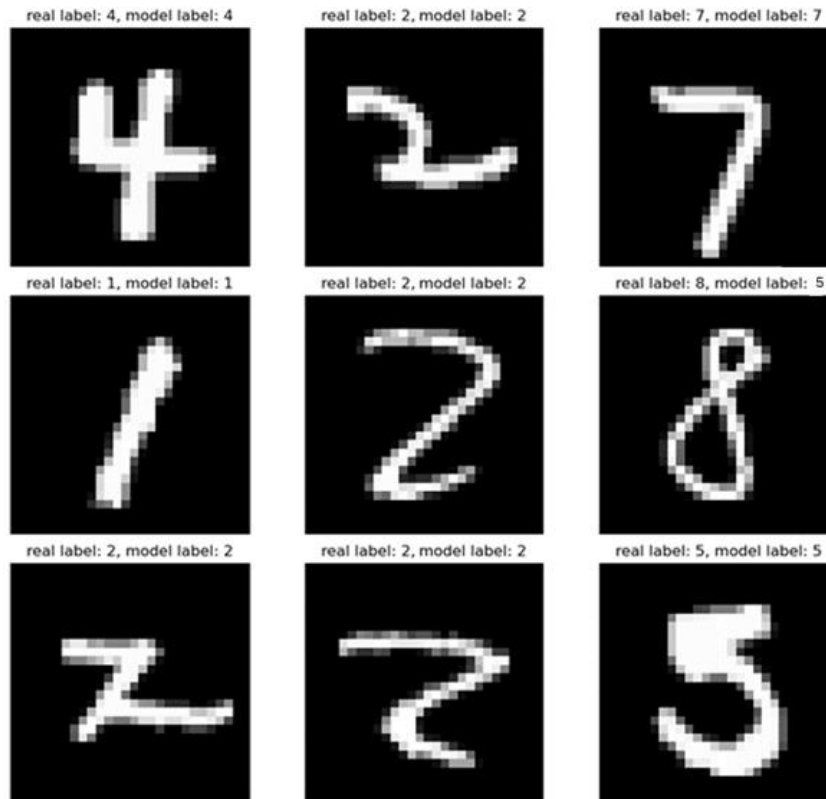


Figure 48. Illustration of test in MNIST dataset

#### 5.4.2 Evaluation of the models on the ship dataset

Not only did we evaluate our models on the MNIST dataset to confirm its validity as a dummy dataset, but we also tested them on the ship dataset (Cohen et al., 2017; JAIN, 2019). Table, illustrates that among the single models, Model 1 achieved the highest accuracy, not only

in the testing set at 37% but also in the training set at 34%. Conversely, Model 5 exhibited the lowest accuracy, in the training set at 17% and in the testing set at 11%.

Even though using ensemble learning was better than using just one model, its accuracy was still not great because each model’s accuracy was low but, in the pruning, the problem of memory cost would be solved. In the pruning strategy, instead of using all models, just non-redundant models would be chosen. The non-redundant subsets were defined based on their architecture and the memory costs were calculated. The first priority, due to low memory cost, would be (M2, M4) with a memory cost of 41 MB and an accuracy of 37%. The second priority is (M2, M5) with the same accuracy and a memory size of 43 MB. The third priority achieved 39% accuracy with a memory cost of 778. However, the accuracy of pruning could not increase significantly because the accuracy of each single model was low and within the same range. Even after pruning out negatively contributing models, the overall accuracy could not reach higher levels due to the similar accuracy range of the remaining models. Overall, the result verifies our strategy due to lower memory cost usage and fewer models in the MAGNUT.

Table 5. Accuracy of the models with the ship dataset

Model/Accuracy	Model_1	Model_2	Model_3	Model_4	Model_5	Ensemble Learning	Ensemble Pruning			
							M2,M4	M2,M5	M1,M2	M2,M3
<b>Train- Accuracy</b>	34%	31%	-	22%	17%	-	-	-	-	-
<b>Test Accuracy</b>	37%	30%	-	20%	11%	39%	37%	37%	39%	-
<b>Memory_cost(MB)</b>	738	40	-	1	3	294	41	43	778	-

## 5.5 Further Results

### 5.5.1 Advanced Model Architecture

We designed advanced CNN model architectures to show the adaptability of our strategy on other strong models. Through designing models, pre-trained networks and custom configurations were used. The aim of designing the models was to enhance performance, especially for memory-constrained devices like drones. Pre-trained models were trained on a general dataset like ImageNet and further training on the specific dataset (Deng et al., 2009; Russakovsky et al., 2015). This strategy can improve model performance. ImageNet is a large-scale visual database utilized for training DL models in computer vision tasks (Deng et al., 2009).

#### 1. Advanced Model 1



Model 1 used MobileNet as the base model due to its high memory efficiency. It integrated a pre-trained MobileNet from ImageNet. The top layer was excluded to adapt to the custom classification (Deng et al., 2009; Howard et al., 2017). Additional layers include a convolutional layer, GAP, and a dense output layer conducted to our specific number of classes. It was designed to be compact enough for simple devices. For compiling the model, “Adam” as an optimizer was used which is suitable for training deep neural networks (Kingma & Ba, 2017). “Categorical\_crossentropy” was chosen as a loss function which is common in multi-classification tasks (Z. Zhang & Sabuncu, 2018). “Sigmoid” activation was used in the convolutional layers to make the output between 0 and 1 (S. Sharma et al., 2017). The “Softmax” function was used in the output layer of the classification model (Liu et al., 2016). It converts raw scores into probabilities which helps to define the predictions.

## 2. **Advanced Model 2**

Model 2 is a simplified variant of the ResNet architecture. It was designed to be smaller and more memory efficient. It used residual connections to prevent vanishing gradients (He et al., 2016). It featured multiple convolutional layers with L1L2 regularization to prevent overfitting (Bidaran & Sharifian, 2021). Batch normalization was applied to maintain effective learning rates throughout training, followed by “ReLU” as an activation function (Agarap, 2018). The “Softmax” function was used in the output layer of the classification model (Liu et al., 2016). For compiling, the same structure was used for all 5 models. This design aimed to reduce its memory usage and computational requirements, making it suitable for drones.

## 3. **Advanced Model 3**

Model 3 utilizes VGG19, a well-known architecture (Simonyan & Zisserman, 2014). We fine-tuned the last five layers while freezing the earlier layers. This approach used the pre-trained weights on ImageNet. It enables the model to benefit from previously learned features. Moreover, it helps in adapting to the new classification task. The same activation function was used for the output layer of the classification model. However, the learning weight was not set by default, and it was set to 0.001. This model consumed more memory compared to the previous models because of the sensitivity in feature extraction.

#### 4. **Advanced Model 4**

Model 4, designed without fine-tuning any pre-trained network, was built from scratch and includes several layers of convolution with batch normalization (Ioffe & Szegedy, 2015). The activation function was “RELU” and the same function was used in the output layer of the classification model (Agarap, 2018). The compiler of this model was the same as the model\_3 which means the learning rate is set to 0.001. This model focused on learning features directly from the training dataset, making it fully customizable and adaptable to specific features of the dataset.

#### 5. **Advanced Model 5**

Model 5 uses ResNet50 for its base, leveraging the strong feature extraction capabilities of ResNet50 with partial fine-tuning (He et al., 2016). The last five layers are trainable, which allows the model to adjust more extensively to the new dataset while still benefiting from ImageNet. The “Softmax” function is used in the output layer of the classification model (Liu et al., 2016). For compiling the model, the same function is used except for the learning weight that is set to 0.0001. This model consumes more memory. As shown in Figure 49, For visualization of all proposed models, the Netron website which is an open source is used (Roeder, n.d.).

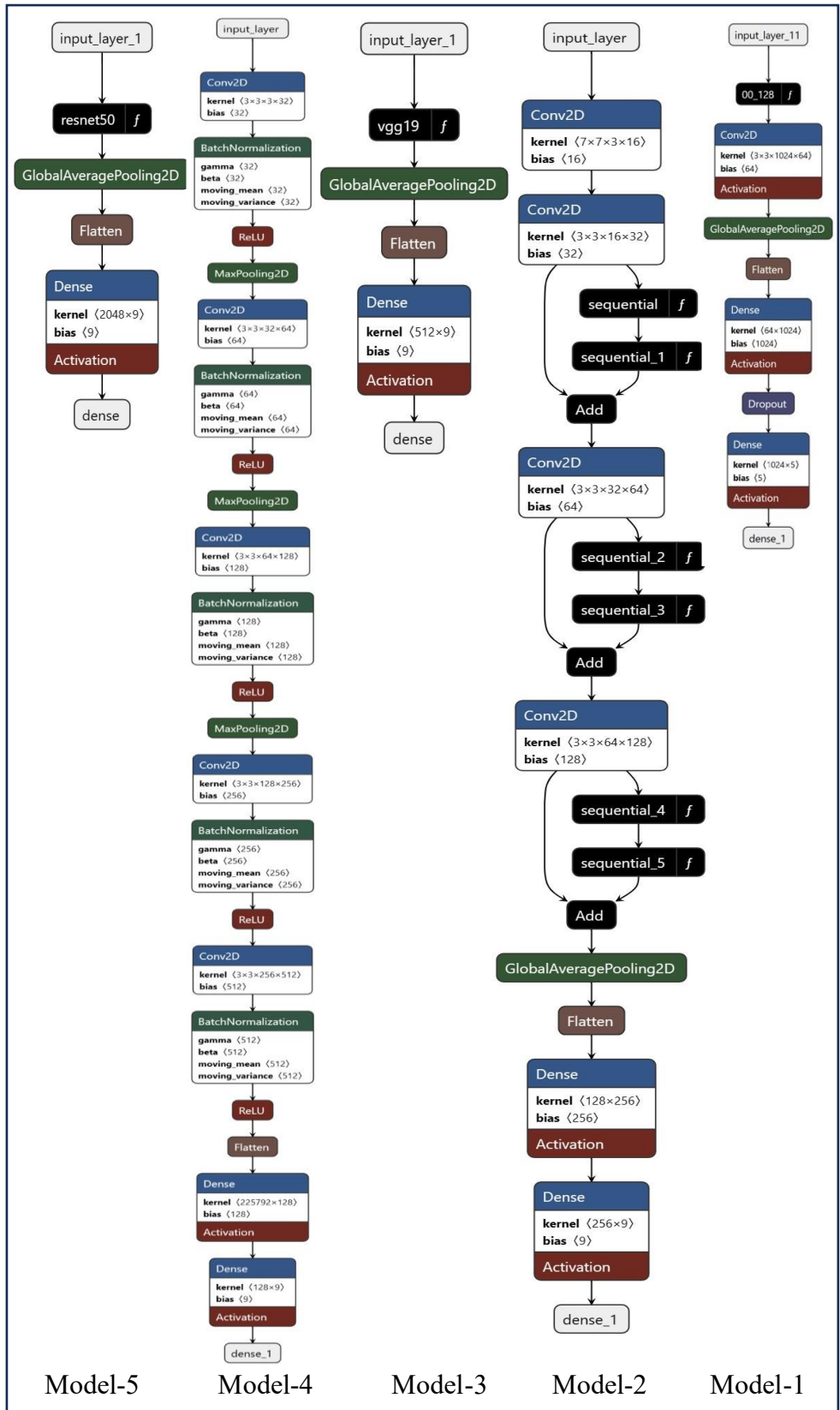


Figure 49. Advanced models architecture, drawn by Netron (Roeder, n.d.)

### 5.5.2 Result Of Using Advanced Models On The Jellyfish Dataset

The jellyfish dataset was used to train five advanced models. Table , demonstrates their effective performance. Using the previous architecture achieved 36% accuracy in the test set. However, the advanced version gained remarkable improvement, reaching 99% accuracy. For Model 2, the previous architecture achieved 19% accuracy, while the new architecture improved it to 27%. Model 3 encountered challenges with computation due to high demands in the previous architecture. However, the advanced model classed test images with 10% accuracy and trained itself with 39% accuracy. Model 4, was trained with the previous architecture and achieved 23% accuracy, however, the advanced model 4 reached a training accuracy of 41% and maintained the same testing accuracy of 13%. Model 5, with the previous architecture, achieved accuracies in the train and 13% in the test. Conversely, the advanced model 5 provides considerable enhancement. It achieves a training accuracy of 71% and a testing accuracy of 15%. The ensemble results also showed improvement compared to the previous models, with an accuracy of 37%. The accuracy of pruning was not calculated due to the high number of layers and complexity of architecture in each model.

Table 6. Comparison of advanced and old model accuracy on jellyfish dataset.

Model/Accuracy	Model_1	Model_2	Model_3	Model_4	Model_5	Ensemble Learning	Ensemble Pruning
Training accuracy of previous model's architecture	53%	14%	-	23%	24%	-	-
Training accuracy of advanced model's architecture	99%	68%	39%	41%	71%	-	-
Testing accuracy of the previous model's architecture	36%	19%	-	13%	13%	26%	-
Testing the Accuracy of Advanced Model's architecture	99%	27%	10%	13%	15%	37%	-
Memory Cost of advanced models architecture (MB)	41	6	152	357	101	657	-

### 5.5.3 Result Of Using Advanced Models On The Ship Dataset

The same ship dataset was chosen to train our advanced models, and the results, as shown in Table , improved significantly. The architecture of the previous Model 1 had an accuracy of 34% in training and 37% in testing. However, with the advanced Model 1, the training accuracy reached 100%, and the testing accuracy increased to 82%. Model 2 improved from 31% to 77% in training and 30% to 11% in testing with the advanced architecture. Model 3 achieved 100% training and 71% testing accuracy with the advanced architecture. The training accuracy of Model

4 increased to 33% with the advanced model, but testing accuracy decreased to 15%. Model 5 with the advanced architecture showed improvements in both training and testing accuracy compared to the old model. Its training accuracy reached 66%. In testing, it achieved 37% accuracy. The advanced models in the ensemble also outperformed the previous ensemble models. The ensemble of advanced models achieved an accuracy of 44%, exceeding the accuracy of the previous ensemble models, which was 39%.

*Table 7. Comparison of advanced and old model accuracy on ship dataset*

Model/Accuracy	Model_1	Model_2	Model_3	Model_4	Model_5	Ensemble Learning	Ensemble Pruning
<b>Training Accuracy of old model's architecture</b>	34%	31%	-	22%	17%	-	-
<b>Training Accuracy of advanced model's architecture</b>	<b>100%</b>	<b>77%</b>	<b>100%</b>	<b>33%</b>	<b>66%</b>	<b>39%</b>	-
<b>Testing the Accuracy of the old model's architecture</b>	37%	30%	-	20%	11%	-	-
<b>Testing Accuracy of advanced model's architecture</b>	<b>90%</b>	<b>11%</b>	<b>71%</b>	<b>15%</b>	<b>37%</b>	<b>44%</b>	-
<b>Memory Cost of advanced models architecture (MB)</b>	13.39	10	36.01	116.2	4.06	171.65	-

## 5.6 Conclusion

In this chapter, an analysis of experiments was conducted to evaluate proposed strategies such as MAGNAT and MAGNUT within the maritime management framework. It begins with the selection of two datasets, Jellyfish and Ship from Kaggle. The jellyfish dataset had a significant challenge, that allowed utilizing the VGG16 model and the MNIST dataset to define the problem. After extensive evaluation, it became evident that the datasets were challenging. The results of our models with the MNIST dataset were satisfactory and proved the effectiveness of our strategies. However, the ship dataset did not present such challenges and served to validate our strategy. After training on the ship dataset, the research proposed five advanced CNN models to improve efficiency. With these advanced models, both datasets were used and demonstrated the possibility of using proposals on diverse architecture.

## **Chapter 6: Conclusion and Future Perspective**

As detailed below, this section aims to first present the summary and conclusion of the master's thesis. It will then address the research questions introduced in Section 1 and subsection 1.2, available on page 15, entitled "Research Questions." Finally, the related publications of this thesis study will be presented.

### **6.1 Master Thesis Summary And Conclusions**

Using AI in maritime management plays an important role in marine and maritime safety and security issues. Ports are the main gateways for international trade. There are billions of dollars of goods handled in ports. That makes the port vulnerable to illegal activities such as smuggling, piracy, and terrorism. Safety and environmental protection are the main concerns, given the presence of heavy machinery, hazardous materials, and significant human activity, all of which can endanger the area. Security protocols help prevent accidents. AI, especially DL is one solution for enhancing the ports' security.

The thesis focuses on using DL to classify different maritime objects. Maritime objects such as ships vary in size and shape, making them difficult to distinguish. Ensemble learning is a powerful ML technique, that can improve performance and accuracy. In ensemble learning, local predictions are combined to make a meta-model. However, using ensemble learning in maritime management brings different challenges. Two main challenges in ensemble learning are the consumption of significant computational time and the memory resources required. This is because multiple models need to be loaded and executed individually. Also, the performance of models is not the same and some models may have a negative impact on the overall learning process especially when models are aggregated. In this thesis, the author proposes intelligent strategies focused on model selection and performance optimization to address these challenges. MAGNAT is one of the main proposed strategies in this thesis that uses ensemble learning for maritime management. With its specialized algorithm, this strategy tries to decrease the effect of models that contribute negatively and enhance the effect of positive models by assigning weight factors based on their historical performance. In the MAGNAT strategy, collecting datasets from maritime authorities and preprocessing the data is the first step. Preprocessing, especially in CNN models is a significant step. Each CNN model is trained based on prepared data and makes their

local predictions. In the aggregation process, a weight factor is consumed to do the global prediction. Ensemble learning with an explained strategy is more accurate compared to the prediction of every single model. In weighted voting, the influence of each model on the overall global prediction is based on its performance over time. Models with better performance receive higher weight probabilities and would have more influence. Using a threshold value for the final prediction can ensure that only confident predictions participate in the final classification. Each model's weight based on its performance is updated periodically. To improve the overall performance of this strategy, MAGNUT as an intelligence aggregation is proposed. It differs only in the aggregation method of the previously explained CNN models. Unlike existing solutions, MAGNUT aggregation focuses on a set of non-redundant models within the ensemble. This intelligent aggregation of the MAGNUT method utilizes knowledge-driven pruning to define a minimum subset of non-redundant models and then the voting mechanism for the final output is applied. With this approach, the efficiency of the ensemble learning for the CNN models is improved.

These two innovative techniques, MAGNAT and MAGNUT, are not just theoretical. Various experiments have been conducted to evaluate their performance using Python programming codes. The experimentation began by choosing two datasets from the Kaggle website, such as the jellyfish dataset related to marine science, and the ship dataset relevant to maritime technology. The reason for choosing these two datasets was to demonstrate that the proposed strategies can be applied in different domains such as marine science and maritime technology. The Jellyfish dataset consists of nine classes of fish, and the ship dataset consists of five classes of ships. Both datasets consist of training and testing samples. This split helps us train our models with the training dataset and test models' accuracy with the test dataset. For model visualization, Netron as a user-friendly tool for DL models was used. This helps to better understand the components of each model architecture. First, CNN models were trained using the jellyfish dataset. Different image sizes were tested, and the size of 28 x 28 pixels was selected due to its high accuracy. The training was set to 50 epochs with a batch size of 132. However, the accuracy of correctly classified images in the test from individual models and ensemble learning with pruning was not satisfactory. The reason could be due to model robustness or an inappropriate dataset. Models were replaced with the popular VGG16 model. Even with the VGG16 model, significant improvement in accuracy was not achieved. The next step was

checking the dataset by replacing it with another dataset. A popular dataset in image classification which is called MNIST was used to replace with the jellyfish dataset. Images were fixed at 28x28 pixels with a batch size set to 128 and a fixed epoch of 100. The results were satisfactory. MAGNAT achieved 47% accuracy with a memory cost of 294 MB. MANUT strategy solved not only the problem of memory cost but also reduced the number of the models of the ensemble. Instead of having a large number of models, three non-redundant subsets were defined in MAGNUT. Each subset based on their memory cost was prioritized. The first subset with a memory cost of 1.5 was the first priority. The second subset with the memory cost of 3 MB was the second priority and the third one with the memory cost of 28 MB got the higher accuracy with 74%. The reason for the low accuracy with the jellyfish dataset was determined to be the dataset itself. The dataset had contextual issues such as clarity and consistency in lighting. It was evident that harbor seals represented a different species of animal depicted in varied contexts. The models were also trained on the ship dataset, demonstrating the effectiveness of our strategy on maritime technology. The ensemble achieved an accuracy of 39%, indicating the success of the MAGNAT strategy. For MAGNUT, instead of using a large number of models, we defined three subsets that are both general and non-redundant. We, then calculated the memory costs for each subset, prioritizing the one with the lowest memory requirement. The first subset had a memory cost of 41 and an accuracy of 37%. The second subset had a memory cost of 43 and an accuracy of 37%, while the last one had a memory cost of 778 and an accuracy of 39%. After evaluating each single model on different datasets, we realized that the models' architecture needs improvement. Therefore, we proposed 5 advanced model architectures. Three of the models used fine-tuning techniques, allowing them to be trained not only on general datasets such as ImageNet but also on specific datasets such as jellyfish or ship datasets. The other two models were designed from scratch. Various techniques were employed on each model's architecture to enhance the accuracy of each model. Five advanced models were trained and tested using the same datasets such as the jellyfish dataset and the ship dataset. The results with the advanced model were conducted to show how these proposed strategies can be used in a variety of architectures

The author learned about the significance and methodologies of employing ensemble learning and pruning techniques in maritime management systems. The thesis emphasizes the optimization of maritime management through advanced DL architectures, particularly ensemble learning, to enhance the performance and accuracy of models in maritime operations. It



introduced a novel framework that combines ensemble learning with innovative pruning techniques to efficiently manage computational resources while achieving high precision in tasks like ship classification and abnormal detection. The research evaluates various DL architectures, showing their applicability across diverse maritime datasets and optimizing ensemble models through strategic pruning for better decision support and operational optimization in maritime applications.

The proposed solution in this research is the use of AI-driven solutions, such as surveillance drones equipped with advanced detection algorithms, to enhance port security. These drones aim to recognize different classes of ships with high accuracy and send their reports to an analyzer, which, based on the level of threats, decides whether to forward them to maritime authorities. The future work of this research encompasses a wide array of ideas. The two novel solutions proposed, MAGNAT and MAGNUT, could be applied to popular models. This means that instead of using our five CNN models, we could utilize well-known models such as VGG16 or DenseNet. Another possibility involves applying MAGNAT and MAGNUT to other datasets for other applications. Top of Form We did not consider different weather conditions in our scenario, which exist in real-world scenarios. Capturing different pictures from various angles in poor weather conditions could pose a challenge for drones, especially in Norway, where summer and winter differ a lot. On one hand, capturing images from different angles could be difficult for the drone if the weather is windy or rainy. On the other hand, analyzing and recognizing unclear pictures could be challenging for algorithms to accurately classify images with high accuracy. The classification of different marine objects was done but one of the interesting future works could be detection and classification together.

## **6.2 Answering Research Questions**

For the first research question (RQ1) on addressing demanding resource requirements and substantial computational time in ensemble learning:

Section 4 describes a novel ensemble learning system, “MAGNAT” and “MAGNUT,” which are designed to address the issue of demanding resource requirements in ensemble learning. Specifically, MAGNUT focuses on ensemble pruning to reduce computational and memory demands by selecting and combining the most effective models, ensuring that only those

contributing significantly to performance are utilized. This strategy effectively addresses the problem of computational overhead during the inference phase of ensemble learning.

For the second research question (RQ2) regarding handling detrimental effects of specific models in ensemble learning to enhance overall performance:

Sections 5 and 6 discuss the implementation and results of the proposed systems (MAGNAT and MAGNUT) that utilize ensemble pruning and intelligent aggregation techniques. These sections highlight that by carefully selecting models that contribute positively and pruning those that do not, the ensemble's overall effectiveness is enhanced. This not only improves the performance but also ensures that cooperative dynamics within the ensemble contribute constructively to the learning outcomes, avoiding any potential detrimental effects from less effective individual models.

The thesis employs innovative approaches to pruning and intelligent model aggregation to enhance both computational efficiency and cooperative dynamics in ensemble learning systems. It demonstrates significant improvements in maritime image classification tasks, showcasing applications such as identifying fish in marine and maritime science and ships in maritime technology. These techniques could potentially be extended to other business domains requiring image classification and innovative techniques.

### **6.3 Related Scientific Publications**

As of today, May 14, 2024, one scientific conference paper has been published in the “12th World Conference on Information Systems and Technologies (WorldCIST’24),” which is related to the MAGNUT proposal, one of the main contributions of this paper (Mesgaribarzi, 2024). Additionally, other publications are being considered for future publication related to this master's thesis study and will be cited accordingly in the future. It is also worth noting that some figures and results of this master's thesis study have already been published in the aforementioned publication.

## References

- Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv Preprint arXiv:1803.08375*.
- Al-Barazanchi, H., Verma, A., & Wang, S. X. (2018). Intelligent plankton image classification with deep learning. *International Journal of Computational Vision and Robotics*, 8(6), 561–571.
- Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1–6.  
<https://doi.org/10.1109/ICEngTechnol.2017.8308186>
- Anwar, A. (2022, January 22). *Difference between AlexNet, VGGNet, ResNet and Inception*. Medium.  
<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96>
- Aouadi, S., Torfeh, T., Arunachalam, Y., Paloor, S., Riyas, M., Hammoud, R., & Al-Hammadi, N. (2023). Investigation of radiomics and deep convolutional neural networks approaches for glioma grading. *Biomedical Physics & Engineering Express*, 9(3), 035020.
- Augasta, M. G., & Kathirvalavakumar, T. (2013). Pruning algorithms of neural networks—A comparative study. *Central European Journal of Computer Science*, 3(3), 105–115.  
<https://doi.org/10.2478/s13537-013-0109-x>
- Azarian, K., Bhalgat, Y., Lee, J., & Blankevoort, T. (2021). *Learned Threshold Pruning* (arXiv:2003.00075). arXiv. <https://doi.org/10.48550/arXiv.2003.00075>
- Bangar, S. (2022, July 5). Resnet Architecture Explained. *Medium*.  
<https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d>
- Basha, S. S., Dubey, S. R., Pulabaigari, V., & Mukherjee, S. (2020). Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 378, 112–119.

- Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Similarity Matching in Computer Vision and Multimedia*, 110(3), 346–359.  
<https://doi.org/10.1016/j.cviu.2007.09.014>
- Bidaran, A., & Sharifian, S. (2021). *Designing an AI-assisted toolbox for fitness activity recognition based on deep CNN* (p. 34). <https://doi.org/10.1109/IKT54664.2021.9685153>
- Bir, P. (2019). *Image Classification with K Nearest Neighbours*. <https://medium.com/swlh/image-classification-with-k-nearest-neighbours-51b3a289280>
- Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123–140.
- Chauhan, N. K., & Singh, K. (2018a). A review on conventional machine learning vs deep learning. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 347–352.
- Chauhan, N. K., & Singh, K. (2018b). A review on conventional machine learning vs deep learning. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 347–352.
- Chauhan, N. K., & Singh, K. (2018c). A review on conventional machine learning vs deep learning. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 347–352.
- Chávez, C. A. G., Brynolf, S., Despeisse, M., Johansson, B., Rönnbäck, A. Ö., Rösler, J., & Stahre, J. (2024). Advancing sustainability through digital servitization: An exploratory study in the maritime shipping industry. *Journal of Cleaner Production*, 436, 140401.
- Chen, Q., Huang, M., Wang, H., Zhang, Y., Feng, W., Wang, X., Wu, D., & Bhatti, U. A. (2018). A feature preprocessing framework of remote sensing image for marine targets recognition. *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, 1–5.

- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785–794.
- Chen, X., Mao, J., & Xie, J. (2021). Comparison Analysis for Pruning Algorithms of Neural Networks. *2021 2nd International Conference on Computer Engineering and Intelligent Control (ICCEIC)*, 50–56.  
<https://doi.org/10.1109/ICCEIC54227.2021.00018>
- Chin, C. S., & Venkateshkumar, M. (2022). Artificial Intelligence for Maritime Transport. *IEEE Transportation Electrification Community (TEC) eNewsletter*. <https://eprints.ncl.ac.uk>
- Chollet, F. (2015). *Keras*. <https://github.com/fchollet/keras>
- Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). *EMNIST: An extension of MNIST to handwritten letters* (arXiv:1702.05373). arXiv. <https://doi.org/10.48550/arXiv.1702.05373>
- Cortes, C., & Vapnik, V. (1995). *Support-Vector Networks* | SpringerLink.  
<https://doi.org/10.1023/A:1022627411411>
- Crammer, K., & Singer, Y. (2001). *On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines*. <https://api.semanticscholar.org/CorpusID:10151608>
- Cunningham, P., & Delany, S. J. (2021). K-Nearest neighbour classifiers-A Tutorial. *ACM Computing Surveys (CSUR)*, 54(6), 1–25.
- Damastuti, N., Siti Aisjah, A., & Masroeri, A. A. (2019). Classification of Ship-Based Automatic Identification Systems Using K-Nearest Neighbors. *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 331–335.  
<https://doi.org/10.1109/ISEMANTIC.2019.8884328>
- de França, F. O. (2018). A greedy search tree heuristic for symbolic regression. *Information Sciences*, 442, 18–32.

- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Li, F.-F. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition* (p. 255).  
<https://doi.org/10.1109/CVPR.2009.5206848>
- Di Ciaccio, A., & Giorgi, G. (2016). Deep learning for supervised classification. *Rivista Italiana Di Economia, Demografia e Statistica, Vol. LXVIV*.
- Ding, G., Zhang, S., Jia, Z., Zhong, J., & Han, J. (2021). Where to Prune: Using LSTM to Guide Data-Dependent Soft Pruning. *IEEE Transactions on Image Processing, 30*, 293–304.  
<https://doi.org/10.1109/TIP.2020.3035028>
- Dong, X., Yu, Z., Cao, W., Shi, Y., & Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science, 14*, 241–258.
- Farabet, C., Couprie, C., Najman, L., & LeCun, Y. (2012). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(8), 1915–1929.
- Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. *Icml, 96*, 148–156.
- Fronzetti, N. (2019). *Predictive Neural Network Applications for Insurance Processes*.
- Fu, B., Wang, Z., Pan, R., Xu, G., & Dolog, P. (2013). An Integrated Pruning Criterion for Ensemble Learning Based on Classification Accuracy and Diversity. In L. Uden, F. Herrera, J. Bajo Pérez, & J. M. Corchado Rodríguez (Eds.), *7th International Conference on Knowledge Management in Organizations: Service and Cloud Computing* (Vol. 172, pp. 47–58). Springer Berlin Heidelberg.  
[https://doi.org/10.1007/978-3-642-30867-3\\_5](https://doi.org/10.1007/978-3-642-30867-3_5)
- Gale, S., Vestheim, S., Gravdahl, J. T., Fjerdingen, S., & Schjolberg, I. (2013). RBF network pruning techniques for adaptive learning controllers. *9th International Workshop on Robot Motion and Control, 246–251*. <https://doi.org/10.1109/RoMoCo.2013.6614616>
- Gallego, A.-J., Pertusa, A., & Gil, P. (2018). Automatic ship classification from optical aerial images with convolutional neural networks. *Remote Sensing, 10*(4), 511.

- Guillen, M. D., Aparicio, J., & Esteve, M. (2023). Gradient tree boosting and the estimation of production frontiers. *Expert Systems with Applications*, 214, 119134.
- Gülsoylu, E., Koch, P., Yildiz, M., Constapel, M., & Kelm, A. P. (2024). Image and AIS Data Fusion Technique for Maritime Computer Vision Applications. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 859–868.
- Gupta, R., & Jha, N. (2020). Real-Time Continuous Sign Language Classification using Ensemble of Windows. *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 73–78. <https://doi.org/10.1109/ICACCS48705.2020.9074319>
- Hagiwara, M. (1993). Removal of hidden units and weights for back propagation networks. *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, 1, 351–354 vol.1. <https://doi.org/10.1109/IJCNN.1993.713929>
- Hashemi, M. (2020). Web page classification: A survey of perspectives, gaps, and future directions. *Multimedia Tools and Applications*, 79(17), 11921–11945. <https://doi.org/10.1007/s11042-019-08373-8>
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv Preprint arXiv:1207.0580*.
- Hoffmann, J., Navarro, O., Kastner, F., Janßen, B., & Hubner, M. (2017). *A Survey on CNN and RNN Implementations*.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications* (arXiv:1704.04861). arXiv. <http://arxiv.org/abs/1704.04861>

- Hristov, A., Nisheva-Pavlova, M., & Dimov, D. (2019). *Filters in Convolutional Neural Networks as Independent Detectors of Visual Concepts* (p. 117). <https://doi.org/10.1145/3345252.3345294>
- Huang, G., Liu, Z., & Weinberger, K. (2016). *Densely Connected Convolutional Networks*. 12.
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 448–456.
- Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2704–2713.
- Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Deep features for text spotting. *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part IV 13*, 512–528.
- JAIN, A. (2019). *Game of Deep Learning: Ship datasets*.  
<https://www.kaggle.com/datasets/arpitjain007/game-of-deep-learning-ship-datasets/data>
- Joseph, F. J. J., Nonsiri, S., & Monsakul, A. (2021). Keras and TensorFlow: A hands-on experience. *Advanced Deep Learning for Engineers and Scientists: A Practical Approach*, 85–111.
- Kaggle. (n.d.). Kaggle: Your Home for Data Science. Retrieved May 11, 2024, from <https://www.kaggle.com/>
- Karami, E., Prasad, S., & Shehata, M. (2017). Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. *arXiv Preprint arXiv:1710.02726*.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30.
- Khruschev, S. S., Plyusnina, T. Yu., Antal, T. K., Pogosyan, S. I., Riznichenko, G. Yu., & Rubin, A. B. (2022). Machine learning methods for assessing photosynthetic activity: Environmental monitoring



applications. *Biophysical Reviews*, 14(4), 821–842. [https://doi.org/10.1007/s12551-022-00982-](https://doi.org/10.1007/s12551-022-00982-2)

2

Kingma, D. P., & Ba, J. (2017). *Adam: A Method for Stochastic Optimization* (arXiv:1412.6980). arXiv.

<https://doi.org/10.48550/arXiv.1412.6980>

Kohavi, R. (1998). Glossary of terms. *Machine Learning*, 30, 271–274.

Konstantinov, A. V., & Utkin, L. V. (2021). Interpretable machine learning with an ensemble of gradient boosting machines. *Knowledge-Based Systems*, 222, 106993.

Kumar, Y., Gupta, S., & Singh, W. (2022). A novel deep transfer learning models for recognition of birds sounds in different environment. *Soft Computing*, 26(3), 1003–1023.

<https://doi.org/10.1007/s00500-021-06640-1>

Kuo, C.-C. J. (2016). Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41, 406–413.

<https://doi.org/10.1016/j.jvcir.2016.11.003>

Kuruvayil, S., & Palaniswamy, S. (2022). Emotion recognition from facial images with simultaneous occlusion, pose and illumination variations using meta-learning. *Journal of King Saud University - Computer and Information Sciences*, 34(9), 7271–7282.

<https://doi.org/10.1016/j.jksuci.2021.06.012>

L. Samuel, A. (1959). *Some Studies in Machine Learning Using the Game of Checkers* | *IBM Journals & Magazine* | . <https://ieeexplore-ieee-org.ezproxy2.usn.no/abstract/document/5392560>

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), Article 7553.

<https://doi.org/10.1038/nature14539>

Leonidas, L. A., & Jie, Y. (2021). Ship classification based on improved convolutional neural network architecture for intelligent transport systems. *Information*, 12(8), 302.

- L.Hosch, W. (2023, September 23). *Machine learning | Data Science, Algorithms & Automation | Britannica*. <https://www.britannica.com/technology/machine-learning>
- LI, B., XIE, X., WEI, X., & TANG, W. (2021). Ship detection and classification from optical remote sensing images: A survey. *Chinese Journal of Aeronautics*, 34(3), 145–163.  
<https://doi.org/10.1016/j.cja.2020.09.022>
- Li, F.-F., Karpathy, A., & Johnson, J. (2015). Convolutional neural networks for visual recognition.  
*Available in Http://Cs231n. Github. Io/Convolutional-Networks*.
- Liu, W., Wen, Y., Yu, Z., & Yang, M. (2016). Large-margin softmax loss for convolutional neural networks.  
*arXiv Preprint arXiv:1612.02295*.
- Lorencin, I., Meštrić, H., & Car, Z. (2021). Intelligent Automation System for Vessels Recognition: Comparison of SIFT and SURF Methods. *Tehnički Vjesnik*, 28(4), 1221–1226.
- Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 91–110. <https://doi.org/10.1023/B:VISI.0000029664.99615.94>
- Luu, N. D., & Anh, V. T. (2023). Application of Swin Transformer Model to Retrieve and Classify Endoscopic Images. *International Conference on Intelligent Systems and Data Science*, 161–173.
- M. Geetha, & Neena Aloysius. (2017). *A review on deep convolutional neural networks | IEEE Conference Publication | IEEE Xplore*. <https://ieeexplore-ieee-org.ezproxy2.usn.no/document/8286426>
- Mana, S. C., & Sasipraba, T. (2021). Research on Cosine Similarity and Pearson Correlation Based Recommendation Models. *Journal of Physics: Conference Series*, 1770(1), 012014.  
<https://doi.org/10.1088/1742-6596/1770/1/012014>
- Marine Animal Images*. (n.d.). Retrieved May 13, 2024, from  
<https://www.kaggle.com/datasets/mikoajfish99/marine-animal-images/data>

- Meena, L., & Velmurugan, T. (2023). Optimizing Facial Expression Recognition through Effective Preprocessing Techniques. *Journal of Computer and Communications*, 11(12), 86–101.
- Mesgaribarzi, N. (2024). MAGNAT: Maritime Management Ensemble Learning System. In Á. Rocha, H. Adeli, G. Dzemyda, F. Moreira, & A. Poniszewska-Marańda (Eds.), *Good Practices and New Perspectives in Information Systems and Technologies* (pp. 3–12). Springer Nature Switzerland. [https://doi.org/10.1007/978-3-031-60218-4\\_1](https://doi.org/10.1007/978-3-031-60218-4_1)
- Mohammed, A., & Kora, R. (2023). A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University - Computer and Information Sciences*, 35(2), 757–774. <https://doi.org/10.1016/j.jksuci.2023.01.014>
- Munim, Z. H., Dushenko, M., Jimenez, V. J., Shakil, M. H., & Imset, M. (2020). Big data and artificial intelligence in the maritime industry: A bibliometric review and future research directions. *Maritime Policy & Management*, 47(5), 577–597.
- Munoz, A. (2014). Machine learning and optimization. *Courant Institute of Mathematical Sciences*, 1–2.
- Munoz, A. (2017). *Machine Learning and Optimization*. [https://www.cims.nyu.edu/~munoz/files/ml\\_optimization.pdf](https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf)
- Nguyen, A.-D., Choi, S., Kim, W., Ahn, S., Kim, J., & Lee, S. (2019). Distribution padding in convolutional neural networks. *2019 IEEE International Conference on Image Processing (ICIP)*, 4275–4279.
- Nithin, D. K., & Sivakumar, P. B. (2015). Generic feature learning in computer vision. *Procedia Computer Science*, 58, 202–209.
- O’Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv Preprint arXiv:1511.08458*.
- Panchal, V., Sankla, H., Sharma, P., Neema, V., Panchal, A., & Chouhan, S. S. (2023). FPGA implementation of proposed number plate localization algorithm based on YOLOv2 (You Only

- Look Once). *Microsystem Technologies*, 29(10), 1501–1513. <https://doi.org/10.1007/s00542-023-05506-w>
- Partalas, I., Tsoumakas, G., Hatzikos, E. V., & Vlahavas, I. (2008). Greedy regression ensemble selection: Theory and an application to water quality prediction. *Information Sciences*, 178(20), 3867–3879.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., ... Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library* (arXiv:1912.01703). arXiv. <https://doi.org/10.48550/arXiv.1912.01703>
- Peters, D. (2017). Recognising multidimensional Euclidean preferences. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).
- Pradhan, N., Sagar, S., & Jagadesh, T. (2024). Advance Convolutional Network Architecture for MRI Data Investigation for Alzheimer's Disease Early Diagnosis. *SN Computer Science*, 5(1), 167. <https://doi.org/10.1007/s42979-023-02560-z>
- Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, 31.
- Qamar, R., & Zardari, B. (2023). Artificial Neural Networks: An Overview. *Mesopotamian Journal of Computer Science*, 2023, 130–139. <https://doi.org/10.58496/MJCSC/2023/015>
- Ramachandran, R., Rajeev, D. C., Krishnan, S. G., & Subathra, P. (2015). Deep learning an overview. *International Journal of Applied Engineering Research*, 10(10), 25433–25448.
- Reimers, C., & Requena-Mesa, C. (2020). Deep learning—an opportunity and a challenge for geo-and astrophysics. In *Knowledge discovery in big data from astronomy and earth observation* (pp. 251–265). Elsevier.

- Ren, Y., Gu, Z., Pan, L., & Liu, C. (2020). The Class Overlap Model for System Log Anomaly Detection Based on Ensemble Learning. *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, 369–374.
- Ren, Y., Yang, J., Zhang, Q., & Guo, Z. (2019). Multi-feature fusion with convolutional neural network for ship classification in optical images. *Applied Sciences*, *9*(20), 4209.
- Rincy, T. N., & Gupta, R. (2020). Ensemble learning techniques and its efficiency in machine learning: A survey. *2nd International Conference on Data, Engineering and Applications (IDEA)*, 1–6.
- Roeder, L. (n.d.). *Netron*. Retrieved May 11, 2024, from <https://netron.app/>
- Rokach, L. (2019). *Ensemble learning: Pattern classification using ensemble methods*. World Scientific.
- Rokach, L. (2010). *Pattern classification using ensemble methods*. World Scientific.
- Rousseau, F., Drumetz, L., & Fablet, R. (2020). Residual networks as flows of diffeomorphisms. *Journal of Mathematical Imaging and Vision*, *62*, 365–375.
- Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision* (p. 2571). <https://doi.org/10.1109/ICCV.2011.6126544>
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). *ImageNet Large Scale Visual Recognition Challenge* (arXiv:1409.0575). arXiv. <https://doi.org/10.48550/arXiv.1409.0575>
- Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, *8*(4), e1249. <https://doi.org/10.1002/widm.1249>
- Salem, M. H., Li, Y., Liu, Z., & AbdelTawab, A. M. (2023). A Transfer Learning and Optimized CNN Based Maritime Vessel Classification System. *Applied Sciences*, *13*(3), 1912.
- Saponara, S., & Elhanashi, A. (2022). Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance. In S. Saponara & A. De Gloria (Eds.), *Applications in Electronics*

- Pervading Industry, Environment and Society* (pp. 10–17). Springer International Publishing.  
[https://doi.org/10.1007/978-3-030-95498-7\\_2](https://doi.org/10.1007/978-3-030-95498-7_2)
- Saravanan, C. (2010). Color Image to Grayscale Image Conversion. *2010 Second International Conference on Computer Engineering and Applications, 2*, 196–199.  
<https://doi.org/10.1109/ICCEA.2010.192>
- Saxena, S. (2021, March 19). Introduction to The Architecture of Alexnet. *Analytics Vidhya*.  
<https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/>
- Schapire, R. E. (2013). Explaining adaboost. In *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik* (pp. 37–52). Springer.
- Sert, Z. (2020, December 27). ESA(Convolutional Neural Networks). *Medium*.  
<https://zeysert.medium.com/esa-evri%C5%9Fimsel-sinir-a%C4%9Flar%C4%B1-87d9bd986579>
- Setiowati, S., Zulfanahri, Franita, E. L., & Ardiyanto, I. (2017). A review of optimization method in face recognition: Comparison deep learning and non-deep learning methods. *2017 9th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 1–6.  
<https://doi.org/10.1109/ICITEED.2017.8250484>
- Sharma, P. (2023). *CNN vs ANN for Image Classification*. <https://www.tutorialspoint.com/cnn-vs-ann-for-image-classification>
- Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. *Towards Data Sci*, 6(12), 310–316.
- Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404, 132306.
- Shi, Q., Li, W., Tao, R., Sun, X., & Gao, L. (2019). Ship classification based on multifeature ensemble with convolutional neural network. *Remote Sensing*, 11(4), 419.

- Shiruru, K. (2016). AN INTRODUCTION TO ARTIFICIAL NEURAL NETWORK. *International Journal of Advance Research and Innovative Ideas in Education*, 1, 27–30.
- Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv Preprint arXiv:1409.1556*.
- Siswantining, T., & Parlindungan, R. (2021). Covid-19 classification using X-Ray imaging with ensemble learning. *Journal of Physics: Conference Series*, 1722, 012072. <https://doi.org/10.1088/1742-6596/1722/1/012072>
- Smyth, P., & Wolpert, D. (1997). Stacked density estimation. *Advances in Neural Information Processing Systems*, 10.
- Solmaz, B., Gundogdu, E., Yucesoy, V., & Koc, A. (2017). Generic and attribute-specific deep representations for maritime vessels. *IPSI Transactions on Computer Vision and Applications*, 9(1), 22. <https://doi.org/10.1186/s41074-017-0033-4>
- Subasi, A. (2019). *Feature Extraction*. <https://www-sciencedirect-com.ezproxy2.usn.no/topics/biochemistry-genetics-and-molecular-biology/feature-extraction>
- Tipu, R. K., Batra, V., Suman, Panchal, V. R., & Pandya, K. S. (2024). Predictive modelling of surface chloride concentration in marine concrete structures: A comparative analysis of machine learning approaches. *Asian Journal of Civil Engineering*, 25(2), 1443–1465.
- Tiu, E., Huang, Y., Ng, J. L., Aldahoul, N., Ali Najah Ahmed, A.-M., & Elshafie, A. (2022). An evaluation of various data pre-processing techniques with machine learning models for water level prediction. *Natural Hazards*, 110. <https://doi.org/10.1007/s11069-021-04939-8>
- Valerio, L., Nardini, F. M., Passarella, A., & Perego, R. (2022). Dynamic hard pruning of Neural Networks at the edge of the internet. *Journal of Network and Computer Applications*, 200, 103330. <https://doi.org/10.1016/j.jnca.2021.103330>

- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22–30.  
<https://doi.org/10.1109/MCSE.2011.37>
- Varshney, P. (2020). *AlexNet Architecture: A Complete Guide*.  
<https://kaggle.com/code/blurredmachine/alexnet-architecture-a-complete-guide>
- Viken Grini, S. (2019). *Object Detection in Maritime Environments* [Norwegian University of Science and Technology]. [https://folk.ntnu.no/edmundfo/msc2019-2020/grini\\_simen\\_msc\\_reduced.pdf](https://folk.ntnu.no/edmundfo/msc2019-2020/grini_simen_msc_reduced.pdf)
- Wang, Y., Yang, L., Song, X., Chen, Q., & Yan, Z. (2021). A Multi-Feature Ensemble Learning Classification Method for Ship Classification with Space-Based AIS Data. *Applied Sciences*, 11(21), 10336.
- Wu, H., Liu, Q., Liu, X., Zhang, Y., & Yang, Z. (2022). An edge-assisted cloud framework using a residual concatenate FCN approach to beam correction in the internet of weather radars. *World Wide Web*, 25(5), 1923–1949. <https://doi.org/10.1007/s11280-021-00988-y>
- Wu, Y.-C., Lee, Y.-S., & Yang, J.-C. (2008). Robust and efficient multiclass SVM models for phrase pattern recognition. *Pattern Recognition*, 41(9), 2874–2889.  
<https://doi.org/10.1016/j.patcog.2008.02.010>
- Yan, Z., Song, X., Yang, L., & Wang, Y. (2022). Ship Classification in Synthetic Aperture Radar Images Based on Multiple Classifiers Ensemble Learning and Automatic Identification System Data Transfer Learning. *Remote Sensing*, 14(21), 5288.
- Zaniolo, L., & Marques, O. (2020). On the use of variable stride in convolutional neural networks. *Multimedia Tools and Applications*, 79(19), 13581–13598. <https://doi.org/10.1007/s11042-019-08385-4>
- Zeiler, M. D. (2012). Adadelata: An adaptive learning rate method. *arXiv Preprint arXiv:1212.5701*.
- Zhang, C., Benz, P., Argaw, D. M., Lee, S., Kim, J., Rameau, F., Bazin, J.-C., & Kweon, I. S. (2021). ResNet or DenseNet? Introducing Dense Shortcuts to ResNet. *2021 IEEE Winter Conference on*



- Applications of Computer Vision (WACV)*, 3549–3558.  
<https://doi.org/10.1109/WACV48630.2021.00359>
- Zhang, H., & Fan, Y. (2020). Two implementation methods of handwritten numbers recognition. *IOP Conference Series: Materials Science and Engineering*, 768, 072054.  
<https://doi.org/10.1088/1757-899X/768/7/072054>
- Zhang, M., Yang, Y., Pei, H., & Liu, W. (n.d.). *Target Planning for UAV Merchant Ship Recognition Based on KNN Nearest Neighbor Algorithm*.
- Zhang, Z. (2016). Introduction to machine learning: K-nearest neighbors. *Annals of Translational Medicine*, 4(11), Article 11. <https://doi.org/10.21037/atm.2016.03.37>
- Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in Neural Information Processing Systems*, 31.
- Zhao, R., Wang, J., Zheng, X., Wen, J., Rao, L., & Zhao, J. (2020). Maritime Visible Image Classification Based on Double Transfer Method. *IEEE Access*, 8, 166335–166346.  
<https://doi.org/10.1109/ACCESS.2020.3022883>
- Zhou, Z.-H., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1), 239–263. [https://doi.org/10.1016/S0004-3702\(02\)00190-X](https://doi.org/10.1016/S0004-3702(02)00190-X)
- Ziv, Y., Goldberger, J., & Raviv, T. R. (2021). Stochastic weight pruning and the role of regularization in shaping network structure. *Neurocomputing*, 462, 555–567.
- Zuchniak, K. (2023). *Multi-teacher knowledge distillation as an effective method for compressing ensembles of neural networks*.

