"Automated Biofouling Detection and Classification on Ship Hulls"

Department:	**Department of Management Science & Technology**
Program:	**MSc Business Analytics (Part-Time)**
Course:	**Machine Learning & Content Analytics (3rd trimester 2024)**
Students:	**Koromilas Dimitrios (p2822316)**
	Lampos Nikolaos (p2822317)
	Moraitis Anastasios (p2822330)
	Stamatis Ioannis (p2822331)

Table of Contents

Introduction 4
Our project 4
Biofouling Treatment 7
The Challenge 8
Our Vision 10
Methodology 11
Data Collection 12
Dataset Overview 13
Data Processing/Annotation/Normalization 15
Folder-structured Image Dataset 16
Algorithms, Development Environment and Machine Learning Frameworks 17
Development Environment 17
Software 17
Hardware 17
Machine Learning Frameworks 18
Model Architecture, Experiments and Configuration 19
Model Architecture and setup – Convolutional Neural Networks (CNNs) 19
Activation Functions 19
Examples of CNN – Pretrained Models 20
Pretrained Models – Fine Tuning and their practical implementation 20
ResNet50 20
Experiment setup – Fine Tuning and Results 22
Model selection, Final Training and Fine Tuning 23
Final Training 23
Trying different CNN – EfficientNet70 23
Confusion Matrices 24
Train using Validation Set 24
Train using 100% of the training data 25
Fine Tuning Training 25
EfficientNet70 training 26
Results & Quantitative Analysis 27
Discussion, Comments and Future Work 28
Members/Roles 29
Time Plan 30

References 31
Appendix...33

Introduction

We propose a method for marine biofouling detection on vessels' hull, using Machine Learning (ML) algorithm for recognizing vessel biofouling and classification to the appropriate class of fouling. Biofouling, which occurs when marine organisms accumulate on the hull, significantly impacts vessel performance and fuel efficiency. Our current model is trained on real dataset of vessel hulls, being capable to identify the most sever types of fouling present, with classifications being Clean Hull, Algae, Barnacles and Tubeworms. This approach aims to automate the detection process, removing the manual inspection and reporting of fouling.

Building on that, our vision is to further develop this classification model and integrate this into a Remotely Operated Vehicle (R.O.V.) system and into the company's data entry procedures for underwater inspections and cleaning records. This will reduce human error and ensure reliable datasets for biofouling analysis.

Our project

The shipping industry possesses a very crucial role in delivering goods and commodities around the world on a nonstop basis, serving as the backbone of global trade and driving economic growth (Figure 1). Shipping companies remain highly important in the contemporary global economy since it is estimated that around 80% of all goods are carried by sea.

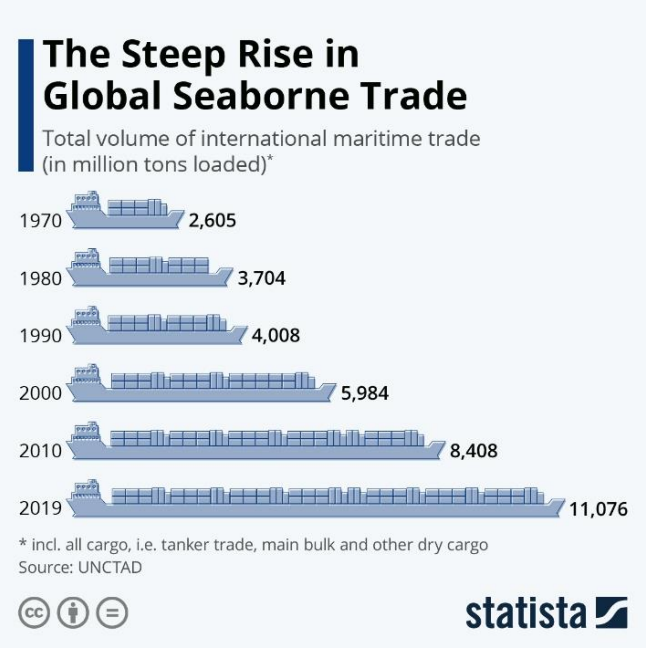

***Figure 1**: The rise of seaborne trade*

However, the maritime industry is also responsible for around 2.9% of global emissions caused by human activities, a significant portion when considering the global scale of shipping. In 2018 only, the global shipping emissions represented 1,076 million tons of CO2, while at EU level, the same industry was responsible for 3% - 4% of the EU's total CO2 emissions, amounting to over 124 million tons of CO2 in 2021 only. Projections have shown that these emissions could increase up to 130% of 2008 emissions by 2050. Therefore, the International Maritime Organization (I.M.O.), being the main regulatory body of the Maritime industry, committed to new targets for GHG emissions reductions and introduced several regulations to compel shipping companies to improve their energy efficiency, thereby lowering fuel consumption and emissions. Key measures include the Energy Efficiency Design Index (EEDI) for new ships, the Energy Efficiency Existing Ship Index (EEXI) for existing vessels, and the Carbon Intensity Indicator (CII), which tracks a ship's annual operational efficiency. As a result, shipping companies are investing substantial time and capital to maintain vessel performance and reduce operational costs.

One of the key challenges in achieving these goals is biofouling, where marine organisms like barnacles, algae, and tubeworms accumulate on the ship's hull (Figure 2). The submerged side of hull, its appendages, and the propeller, tend over time to accumulate marine microorganisms, such as barnacles, tubeworms and algae, when the ship is in motion or stationery. Especially the occurrence of biofouling on the hull and propeller can severely impact the overall performance of a ship, by increasing hydrodynamic drag. Technically, the added resistance caused by the biofouling, increases the propulsive power (in kW) required by the vessel to maintain its speed, which is critical for remaining competitive in the market. As the main engine must work harder to overcome the added resistance caused by biofouling, this power increase leads to higher fuel consumption and consequently more CO2 emission being released in the atmosphere. This directly impacts both operational costs, environmental footprint, and marketability, and may also affect the reputation of the shipping company among the clients (charterers).

To put into perspective the negative effect of biofouling on the performance, fuel consumption and emissions, consider a vessel operating at utilization rate of 70% - 80% per year (approximately 255 to 292 days) with a specific fuel consumption of 180 g/kWh at 40% Maximum Continuous Rating (MCR) of around 9,000 kW. Under such conditions, the extra fuel consumed due to fouling could average 1,080 tones more than normal consumption, which would be equivalent to approximately 600k USD extra operational costs due to overconsumption only.

These 1,080 tons of fuel burnt correspond to approximately 3,363 tons of CO2 emitted in the atmosphere.

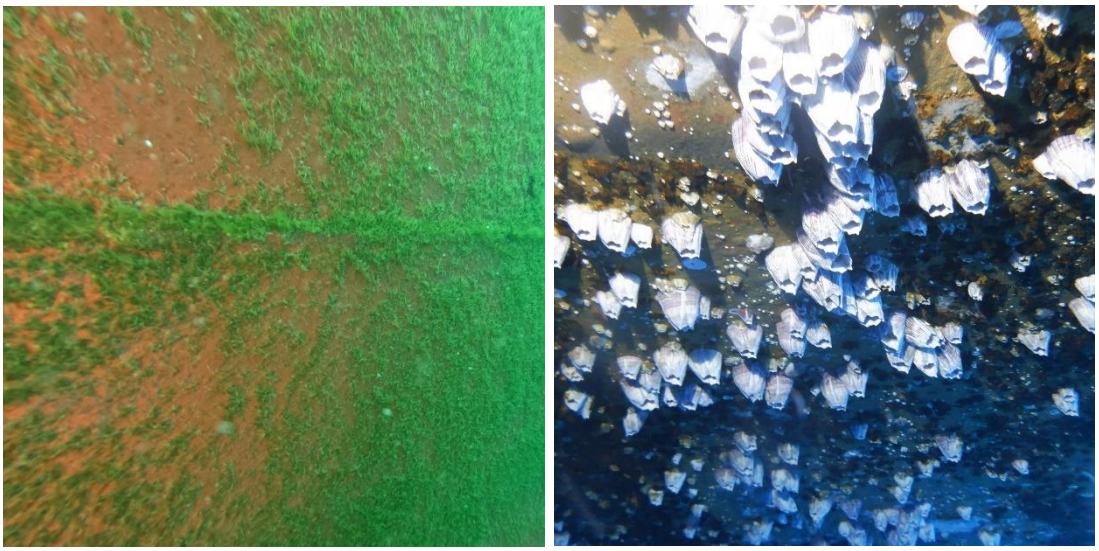

***Figure 2**: Algae (left) and Barnacles (right) biofouling formation on vessel's hull*

Adding to the above, recently the EU has presented a new set of regional regulations, the EU's Emissions Trading System (EU ETS), targeting the greenhouse gas (GHG) emissions from vessels[1] arriving at and departing from EU ports[2]. Therefore, accumulating further cost and adding to the direct economic impact of biofouling to the shipping companies (mainly due to overconsumption), EU has created a system creating a taxation scheme on reported emissions, with the goal of fully taxing emissions by 2027[3]. Although these regulations currently apply only within the EU, which remains a key player in the global markets, there is a broader shift toward greener policies in the shipping industry worldwide. So going forward, it is likely that a similar taxation scheme will expand in more regions.

---

[1] *The EU ETS will apply to all cargo and passenger ships of 5,000 gross tonnage (GT) and above. In 2027, the EU will decide on whether to expand the scope of the ETS to include offshore ships of 5,000 GT and above, and general cargo ships between 400 GT and 5,000 GT.*

[2] *It addresses emissions from maritime transport in the following ways: a) 100% of emissions on voyages departing from and arriving at a port under the jurisdiction of an EU member state, b)50% of emissions on voyages departing from a country outside the EU and arriving at an EU port and 50% of emissions departing from an EU port and arriving in a port located in a non-EU country.*

[3] *The EU ETS for shipping will be phased in starting from 2024. This means that in 2025, shipping companies will have to pay for 40% of their emissions reported in 2024, in 2026, this will increase to 70% of the emissions reported in 2025, from 2027 shipping companies will have to pay for 100% of their emissions*

Biofouling Treatment

As highlighted above, hull biofouling plays an important role in energy consumption and contributes to the rise of vessel emissions. The global marine industry invests significant billions of euros annually in addressing fouling, using a variety of invasive cleaning techniques.

The measures shipping companies use for tackling the biofouling development on their fleets are frequent and timely cleanings of the hull, its appendages and the propeller, and investing in good antifouling paints.

Biofouling cleanings are typically initiated when a drop in performance metrics indicate increased fuel consumption by the main engine, signaling that cleaning may be required. When this happens, shipping companies either schedule an urgent dry-dock or arrange for underwater inspections and cleanings through diving companies. Alternatively and depending on the extent of the drop in performance, ships equipped with specialized ROVs use their cleaning robots to assess the hull's condition with cameras and then proceed with cleaning. Especially in the underwater cleanings, visual data (e.g. pictures from the hull before cleaning and after cleaning) are sent to the company's relevant department, which analyzes the images and the feedback from the diver or the ROV operator, and documents details like the fouling type, its severity, its location on the hull and the cleaning actions taken (Figure 3).

***Figure 3**: An ROV cleaning the vertical parts of a vessel (left) and a diver polishing a propeller (right)*

The second measure, anti-fouling paints, are designed to target specific types of biofoulings (Figure 4). Shipping companies actively test and compare these paints by applying small patches on the hull and monitoring fouling growth. Once performance metrics indicate that a cleaning is required, divers assigned for such job are instructed to first capture the condition of these anti-fouling "test patches", and then proceed with cleaning of the hull. The photos and feedback from the diver are once again sent to the company's relevant department, which is responsible for its analysis and correct documentation, manually updating the company's platform.

***Figure 4**: Different layers of hull painting to tackle biofouling*

The Challenge

In all cases, the need for divers to capture the condition, as well as the requirement for personnel to analyze and document the event in the company's network, presents important challenges associated with the current hull fouling management procedures.

The first problem concerns the decision-making process for need of cleaning, which heavily relies on data gathered through rough qualitative assessments and the subjective judgment of individuals, rather than a structured approach using quantitative criteria. This leaves a significant margin for errors, that either result because of human mistake or conflicting financial interests of the involved parties.

The second problem concerns poor record-keeping practices and the accumulation of inaccurate datasets, which hinder companies from conducting accurate future analyses and drawing meaningful conclusions for various

strategies that can be used to combat fouling and help in optimizing its procedures.

This problem is broken into two pillars:

1) Initial Diver Feedback:

   Divers are responsible for the capturing of the pictures of biofouling while also provide critical feedback through reports, including information such as the location of the fouling on the hull, the type of fouling, the severity or extent to the corresponding region or location of the hull. Since this initial feedback is subject to human error, it increases the risk of misidentifications or misjudgments. In that case, the dataset is exposed to risk of distortion from the very beginning, impacting the reliability of future analyses.

2) Data Entry and Processing:

   Even if the diver's feedback is accurate, the data still need to be processed by the company's relevant department. The information such as fouling type, extent of fouling and location are usually inserted into the company's platform and database. This introduces another layer of risk for human error, that time in data entry. This also compromises the quality of the dataset, making it impossible to perform any type of meaningful analyses on the fouling, such as relation between the fouling type's development and regions, routes, sea water temperature, etc. hindering that way decision making at the management level regarding the biofouling treatment approaches.

Considering all the above, it is evident that hull fouling management is becoming even more crucial for shipping companies and that there is a pressing need for innovative ways that can improve the efficiency, reliability, and scalability of detecting and managing biofouling. The above issues are very common and widespread in the industry and addressing them is essential to maintain operational effectiveness and meet evolving environmental standards.

Our Vision

Considering the above challenges we have developed an automated procedure designed to detect hull contamination with comparable or higher accuracy and reliability than human evaluation. The system classifies hull conditions into four categories; clean hull, algae, barnacles and tubeworms. That way we can determine both the presence and specific type of fouling that is dominant on the hulls surface. In other words, it can distinguish between cases that require cleaning and those where the hull is already clean and does not need immediate attention.

Our solution employs a machine learning algorithm trained on underwater images of ship hulls, which are either clean or exhibit various types of hard fouling, such as algae, tubeworms and barnacles. This model classifies each newly received image into one of the four categories (clean, algae, tubeworms, barnacles) by detecting the most dominant type of fouling present on the hull. The images we used for our model were retrieved from a proprietary dataset provided by a Greek shipping company, with access granted to a subset of their database for confidentiality reasons. The name of the company cannot be disclosed.

Our vision is that this classification model will serve as a backbone for future model that can be integrated into both ROV systems and the company's data entry and quality control procedures for underwater inspections and cleaning records. This will reduce human error and ensure accurate, liable datasets for biofouling analyses. The ROV will be easily deployed by crew aboard each ship, with the aim to periodically inspect the vessel's hull and providing real-time data directly into the database of the company. The data can either log immediately or pass through a secondary protocol or quality control model to ensure accuracy. With the ability to accurately detect and classify fouling, it will supply company researchers with solid data to create insights and add value to the company.

For this to be further improved, the foundation built by the current model could be expanded. This includes recognizing the severity of fouling, in addition to its type, identifying the location (hull or propeller), and detecting multiple fouling types in a single image. These expansions will further enhance decision-making processes, as discussed in the "Future Work" section.

Furthermore, the ROV's smaller size and lower deployment cost compared to existing ROV solutions, will allow more frequent inspections, providing more data at the disposal of the company which can be used to optimize its operational and regulatory costs and increase its revenues by securing more profitable contracts, without risking breaches of performance clauses.

Methodology

Our methodology is centered on utilizing data and machine/deep learning techniques to classify images of a ship's hull to one of four categories. The objective is to enhance the quality of the decisions a maritime company makes about the cleaning of biofouling on the hull of its ships, which currently relies heavily on human judgment, and improve data record keeping. To achieve this, we developed a deep learning model capable of classifying hull images into four of the following distinct classes, thereby providing a more objective and accurate assessment of the most common biofouling conditions:

a) clean hull

b) algae fouling

c) barnacles fouling

d) tubeworms fouling

The reason we selected the classification between these 4 different classes, although more fouling types do exist, is mainly attributed to three factors:

1) The dominance of these fouling types in the dataset: acorn barnacles were found in the original dataset of hull and propeller cleanings 186 times, algae 116 times, tubeworms 43 times, gooseneck 23 times and all rest below 10 times, as our statistical analysis showed (Figure 5).
2) The effect of hard fouling on the performance of the vessel: Out of the classes obtained through the dataset, the tubeworms and barnacles fall under the Hard Fouling class. Hard fouling (tubeworms and barnacles) significantly increases the frictional and viscous resistance, leading to considerable rise in fuel consumption and decrease in propulsion efficiency that cannot be ignored when evaluating the performance of the vessel. Therefore, their inclusion is of profound significance to ensure accurate detection from the model.
3) The size of the dataset: Were we to have a bigger dataset at hand, it would make sense to include more fouling types that are of minor significance compared to the ones selected, but existent, nonetheless.

Thus, the model was built on these four classes: one class for the clean hull, and three for hull needing cleaning due to algae, barnacles, or tubeworm.

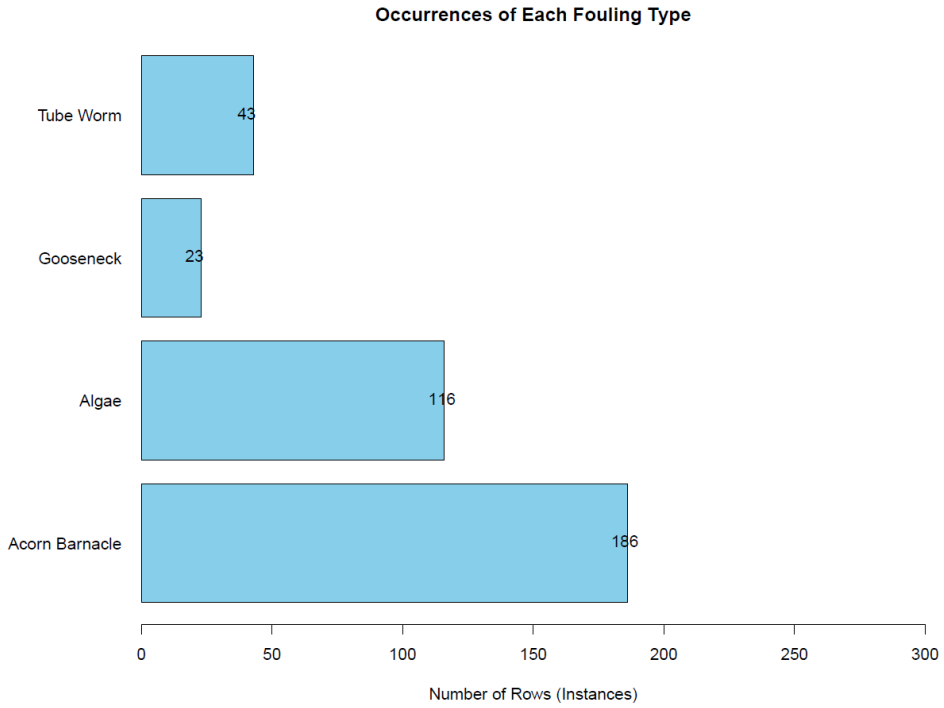

***Figure 5**: Occurrence of every major type of fouling in the original dataset*

Data Collection

Given the highly specialized nature of the problem we aimed to address, we encountered challenges in sourcing a dataset that contains images of ship hulls that were either clean or contaminated with specific microorganisms. Although we identified several datasets which were previously used in research, such as datasets from the Australian Department of Agriculture, Water and the Environment (DAWE), the New Zealand Ministry for Primary Industries (MPI), and the California State Lands Commission (CSLC), we could not obtain access, as these datasets required special permissions.

Eventually, we were able to obtain a dataset for training and evaluating our model through one of our team members who is currently working on the research department of a maritime company. This company maintains a database of images of their ships before and after cleaning and we were granted access to a limited set of these images due to confidentiality concerns.

Initially, we collected a total of 700 colored images from 50 ships in their fleet, representing various fouling conditions observed over the years. After individually examining each image for resolution and clarity, we narrowed the dataset down to 430 images suitable for our analysis, which were divided into four categories. The classification of the whole dataset was performed by us, inspecting each picture and determining its class.

Dataset Overview

The final dataset consists of 430 images of ship hulls, taken either at close or at distance, depicting either clean surfaces or surfaces covered with one of the three types of hard fouling. The resolution of pictures varied across the dataset. Each image was manually classified into its respective category.

Upon closer examination, the dataset is well-balanced across the four distinct classes, with the following distribution per category (Table 1): a) 111 pictures of clean hulls, b)110 pictures with algae, c) 112 pictures with barnacles and d) 97 pictures with tubeworms.

Each class's images were organized into segregated folders to be utilized by our code as the basis for classifying them, as discussed below in “Data Processing” section. This structured approach ensured accurate categorization for model training and evaluation.

Table 1: Distribution of each class in the dataset

	Class	**# of images**
Not Need Cleaning	CLEAN_HULL	111
Hard Fouling	ALGAE	110
	BARNACLES	112
	TUBEWORMS	97
Total		**430**

We chose to focus on these three types of fouling because they are the most observed in real-world scenarios and can spread extensively across a ship’s hull. Their presence significantly increases hydrodynamic drag, leading to higher fuel consumption and increased operational and regulatory costs.

Below follows a brief description for each fouling type and picture samples per class (Figure 6).

a. Algae: Algae are simple, plant-like organisms that often form a slippery layer on the ship’s hull. Because of the simplicity of the plant, rapid growth occurs and further cleaning becomes necessary. Since all algae require light for growth they are not generally found on the flat bottoms of ships. Although usually softer than other fouling types, algae can still significantly increase hydrodynamic drag and fuel consumption.

b. Barnacles: The most encountered fouling animals. Barnacles are hard-shelled marine crustaceans that attach themselves firmly to the hull. They form dense, calcareous layers that create substantial resistance.

c. Tubeworms: Tubeworms are marine organisms that form tubular, calcified structures on surfaces. They can spread extensively on the hull, creating rough textures. Settlement is heavier in warmer waters.

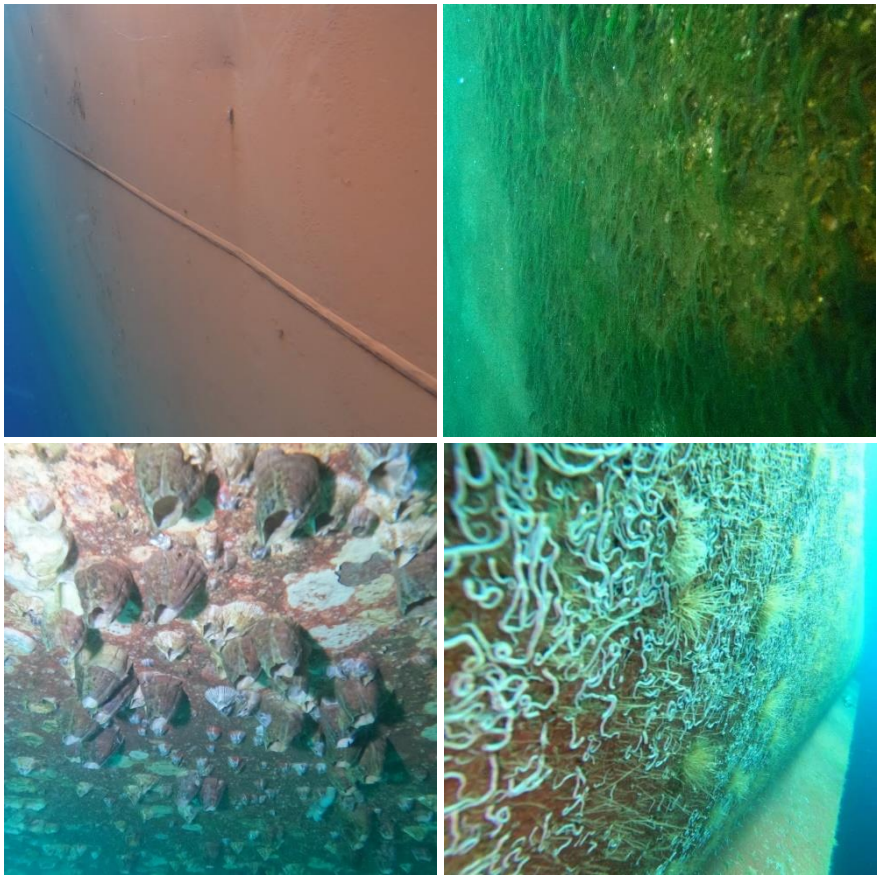

***Figure 6**: A clean hull (top left), algae fouling (top right), barnacles fouling (bottom left), tubeworms fouling (bottom right)*

Data Processing/Annotation/Normalization

At first, due to the images coming from a diverse dataset of photos taken from different divers and on different ships, we had to do some pre-processing to ensure uniformity for the model. There was a multitude of aspect ratios and it was necessary to resize the images to a square shape. In addition to that, the image quality was diverse, with images coming in dimensions 1024x768, 4000x3000, etc. It is known, though, that CNNs require images of the same size for batch processing.

So, there were many ways we could deal with these discrepancies.

- Direct resizing, to simultaneously resize the image by changing the pixels quantity as needed and changing the aspect ratio, leading though to a distorted image. This was considered not a good idea.

- Resizing the image without distortion and using cropping or padding to "fill in" the gaps. This method was deemed more appropriate to create a uniform dataset of square shaped images of a specific resolution. And due to the images already being of a bigger-than-needed size, cropping was enough.

Therefore, we manually cropped the images to a square shape, keeping the target of the image in center view as much as possible, without losing important information. This was accomplished using conventional software for basic photo editing.

Then, the images were ready to be further resized to the needed resolution, as demanded by the model to be used. Resizing was accomplished during import of the images in our code.

Additionally, the images needed to be normalized, so that the pixel values match the distribution expected by the pre-trained model that is used as a basis for fine-tuning. Expected pixel values should be in the scale of 0 and 1 (instead of 0 and 255), so a function is used in our code that does this transformation by means of a simple Keras Rescaling layer.

Finally, data augmentation was an important step in pre-processing. Due to our dataset having a relatively small size, given that the target is to be adequately trained to be able to generalize on unseen data, augmentation was catalytic, because:

- It dynamically "creates" new images from the existing training dataset in each pass, so that each time the model is trained on a slightly

different image. This is done during training, without altering the original dataset.

- This way overfitting is prevented, because it hinders the model from memorizing specific images and can generalize better.
- Simple variations (image rotation, flipping, etc.) can mirror actual variations that may exist in the data, for example due to the way a photo was taken. So, data augmentation can help simulate a larger dataset than the actual one that is used in training.

During training, certain data augmentation techniques were used simultaneously, and more specifically horizontal and/or vertical image flipping, random rotation by up to 20% of a full rotation, and random image zoom by up to 20%. Thus, the model can be trained in similar, but not identical, images in each pass, leading to robustness and the ability to generalize better.

Folder-structured Image Dataset

The Image data for training are organized in a folder-structured dataset. This means that labeling of the respective class on the images is done inherently in the pre-processing step by using the image_dataset_from_directory function of the Keras python library, which is a very convenient way of achieving this. By having the dataset split into folders by class, under a shared directory, the algorithm automatically assigns labels to the images, according to the folder in which each one resides. Thus, it is not needed to manually or otherwise label the data, as was considered preliminarily to be done in dedicated software such as Label Studio.

So, to summarize the pre-processing pipeline, after having manually cropped the images to a square shape, they are imported resized and are automatically labeled. Then, before being fed in the model for training, they are normalized and during training the model is trained on slightly different images due to data augmentation techniques that are utilized: flipping, rotation and zooming.

To create a successful training pipeline, the dataset should be split in different parts: (train, validation and test splits), maintaining a balanced and unbiased representation of for every class in each split:

- During training, the model is readjusting its “weights” acc. to the train split.
- After each epoch, the model’s accuracy is checked on the validation split.
- After reaching a point of adequate training, the model’s accuracy is then checked on the previously unseen test split.

The accuracy of the model on the unseen data should closely resemble the one on the train data. This way robustness of the model is promoted and its ability to generalize is ensured as much as possible.

Algorithms, Development Environment and Machine Learning Frameworks

Development Environment

The whole developing and training procedure took place on a *personal computer* running the below software and hardware implementations:

Software

- **(Host)OS**: *Windows 11 23H2, OS Build: 22631.4169*
- **VM Guest**: *Ubuntu Linux 24.04.1 LTS* (Kernel version: 5.15.153.1-2) as a virtual machine via *Windows Subsystem for Linux (WSL version: 2.2.4.0)*.
- **Graphics**: *CUDA version: 12.3 – cuDNN version: 8.6*
- **Programming Language:** *Python version: 3.12.5*
- **Python Virtual Environment:** *Anaconda version: 24.7.1*
- **Notebook:** *Jupyter Core version: 5.7.2 – Jupyter Server version: 2.14.1*

Hardware

- **CPU**: *i7-14700 @ 5.3Ghz*
- **RAM**: *64G DDR5 @ 5600Mhz*
- **GPU**: NVIDIA *RTX 3060 12G*
- **HDD**: *2T NVMe PCIe Gen3 x4 M.2 SSD*

The initial set up involved the installation of an *Ubuntu Linux* Virtual Machine using *Windows Subsystem for Linux* (WSL). WSL offers brilliant NVIDIA GPU Passthrough capabilities, which means that we can pass an entire NVIDIA GPU PCIe device into a Virtual Machine, resulting in, a near-native, GPU computing acceleration performance.

Next step is about setting up a *Python* development Virtual Environment using *Anaconda* platform. By using a virtual environment set up, which acts like a sandbox, we can isolate our development set up from the rest of the hosting system.

Machine Learning Frameworks

After setting up the *Linux WSL Virtual Machine* and the *Anaconda Python Virtual Environment* platform, we proceed with the installation and setup of all the appropriate Machine Learning Frameworks for Deep Learning using Artificial Neural Networks. Below, we see the ML Frameworks setup:

- **ANN:** *Tensorflow version: 2.17.0* – Tensorflow is the preferred choice for implementing and training Artificial Neural Networks. Initially developed by Google as an open-source machine learning framework, Tensorflow encompasses a large world-wide support community. Main advantages of Tensorflow are the cross-platform compatibility,
- **Tensorflow Wrapper:** *Keras version: 3.5.0* – Keras is high-level neural network API acting as a wrapper (on top) of lower-level frameworks such as Tensorflow. Main advantages of using *Keras* are the simplicity of the code, the user-friendliness and the cross-platform compatibility. It supports a vast amount of NN architectures implementing Sequential models and Function API.

Model Architecture, Experiments and Configuration

Model Architecture and setup – Convolutional Neural Networks (CNNs)

Since our task is related to computer vision, *Convolutional Neural Networks (CNN)* are the appropriate neural network architecture to implement. A CNN is the type of deep learning algorithm mainly used for image recognition. It consists of multiple layers, such as convolutional layers, pooling layers and fully connected layers.

- **Convolutional Layers:** Main aim of a convolutional layer involves decomposing a passing image to a feature map, named activation map, with shape *Number of Inputs X Feature Map Height X Feature Map Width X Feature Map*. More specifically, a convolutional layer applies operations to input images, using filters named as kernels to be able to detect image features such as edges, textures and a variety of patterns.
- **Pooling Layers:** This type of layers is responsible for reducing the data dimensions by combining the outputs of neuron groups at one layer into one neuron of the feature map. Particularly, pooling layers down sample the spatial dimensions of the input, thus, reducing the computational complexity and the huge number of network parameters.
- **Fully Connected Layers:** The Fully Connected Layers connect every neuron in one layer to every neuron of the next one.

Activation Functions

Just like any other NN architecture, a CNN applies the, so-called, *Activation Functions* to the output result of each neuron. One of the most used *Activation Functions* is *ReLU*. *ReLU* is a non-linear activation function that introduces non-linearity to the model. Specifically, *ReLU* returns zero value, in case neuron output is less than zero, else, it returns the actual neuron output.

Examples of CNN – Pretrained Models

There are different types of CNN, such as, *LeNet, AlexNet, ResNet, GoogleNet, etc*. All the previous examples are CNN model architectures that have been trained on a huger amount of image data of numerous different classes.

Pretrained Models – Fine Tuning and their practical implementation

Pretrained models, such as *ResNet50*, have been trained on vast amounts of data classified to numerous different categories. Thus, such models have already learned valuable number of features and patterns of the data. By using a pretrained model as a base model, removing the top layers and connecting custom *Dense* and *Output* layers, we can leverage the knowledge and vastly improve the model performance on our specific task.

Implementation and fine tuning of a pretrained model involves the below steps:

- **Pretrained model selection:** This is a crucial part, since, in involves selection an appropriate pretrained model that best fits and matches the nature of our task. For example, for an image classification model, to select a model with similar features to the task.
- **Base model adjustments:** Modifications to the base model need to be applied. In most cases, the next step is removing the top layers of the pretrained model and adding custom Dense and Output layers suitable for our task.
- **Layer Freezing or Unfreezing:** For reducing complexity, it's essential to freeze some or all the layers of the pretrained model, thus, the *weights* of the corresponding *neurons* shall not get updated. That is, only the *weights* of the neurons of the custom fully connected layers are to be updated allowing the final model to adapt to the new data.
- **Fine Tuning:** Fine Tuning means, that after freezing the base model layers and let only the custom layers to be trained, it's time to unfreeze the pretrained model layers and apply new train using a lower *learning rate* value.

ResNet50

ResNet50 is a CNN architecture belonging to the *ResNet* family. It initially developed by researchers at Microsoft Research Asia and is known for its depth

and efficiency in image classification. Based on the number of its internal layers, ResNet architectures varies from ResNet18 up to ResNet50.

Main *advantages* of *ResNet50* are:

- **Accuracy:** Various benchmarks show that ResNets are able to achieve higher accuracy than conventional neural networks. It can capture patterns in complex dataset, and it is trained on more that 14 million images belonging to 1000 different classes.
- **Better generalization:** Due to its structure, this model learn more general structures in the data and do not focus on specific features of the dataset.
- **Residual Connections:** ResNet50 skips connections allowing gradients to flow more easily through the network. Thus, it does not suffer from the *vanishing gradient* problem.
- **Support and Adoption:** ResNet50 is widely adopted and supported by all big machine learning frameworks, such as, Tensorflow, PyTorch and Keras.

Main disadvantages:

- **Increased complexity:** The skip connections lead to a higher complexity of the model, thus, higher computing requirements.
- **Overfitting:** ResNet50 is prone to overfitting, as long as, small datasets are involved.

Experiment setup – Fine Tuning and Results

Our model setup involves the usage of the *ResNet50* pretrained image classification model. Since our task is about classifying hard ship hull fouling images that belong to 4 separate categories (classes), we must remove the top layers of the *ResNet50* network, freeze the already trained internal layers and add our custom fully connected layers: 1 *Dense* layer plus the *Output* layer containing 4 output neurons.

The experiment is separated to the below steps:

- **Define a ResNet50:** The very first step is about defining the *ResNet50* base model by removing its top layers and giving the appropriate parameters of the input image width and height.
- **Freeze layers:** Next step is freezing the base model pretrained layers, thus, not letting the corresponding neurons *weights* to be updated.
- **Define a function for Model creation:** The function *new_MODEL()* defines and creates a new model applying the base model and the custom fully connected layers
- **Define a function for repeated model training:** The function *exec_Model()* takes as inputs the number of epochs, the train dataset and the test dataset. According to the given number of epochs, it trains repeatedly, for 5 times, five different models. Each one of the 5 training trials produces an individual test evaluation accuracy value. The function, finally, returns a list containing the 5 different accuracies of each train trial.
- **Define a function for trial training:** The function *trial_exec_Models()* calls the function *exec_Model()* and prints out the average accuracy of each trial for the given number of epochs.
- **Trials:** The experiment executes trials for 10, 20, 35, 50 and 100 epochs.

The Table 2, below, shows the average test accuracy values of each trial for the given number of epochs:

Table 2: Test Accuracy values for each epoch trial

Epochs	10	20	35	50	100
Test Accuracy	*0.50*	*0.56*	*0.60*	*0.62*	*0.63*

Model selection, Final Training and Fine Tuning

From *table.1*, model training configuration involving 50 and 100 epochs seems to achieve the highest accuracy on unseen testing data. We choose the 100 epochs setup.

Final Training

Final training involves the below steps:

- We choose the 100 epochs setup, and we proceed with the model training for 100 epochs. By using validation dataset to monitor training progress, we can be sure whether training suffers from overfitting or, even worse, it cannot learn, at all.
- We evaluate the model training on unseen testing dataset and produce the *Confusion Matrix*, as well as the *Precision* and *Recall* values of each one of the 4 classes.
- We train again, this time using the whole training dataset, and we evaluate on the unseen testing dataset printing the *Confusion Matrix* and the *Precision – Recall* values of each class.
- Final step, is the ***Final Tuning***, where we unfreeze the internal layers of the *ResNet50* and by applying lower *learning rate* we train again.

Trying different CNN – EfficientNet70

Last, but not least, we try to train the model using as base model the *EfficientNet70*. In Table 3 below, we the cumulative results of the corresponding accuracies, Confusion Matrix, Recall – Precision values of each one of the training trials.

Table 3: Testing accuracy values for each training trial

Model	**With Validation Set**	**Full train data**	**Fine Tuning**	**EfficientNet70**
Test Accuracy	*0.67*	*0.71*	*0.71*	*0.30*

Confusion Matrices

Train using Validation Set

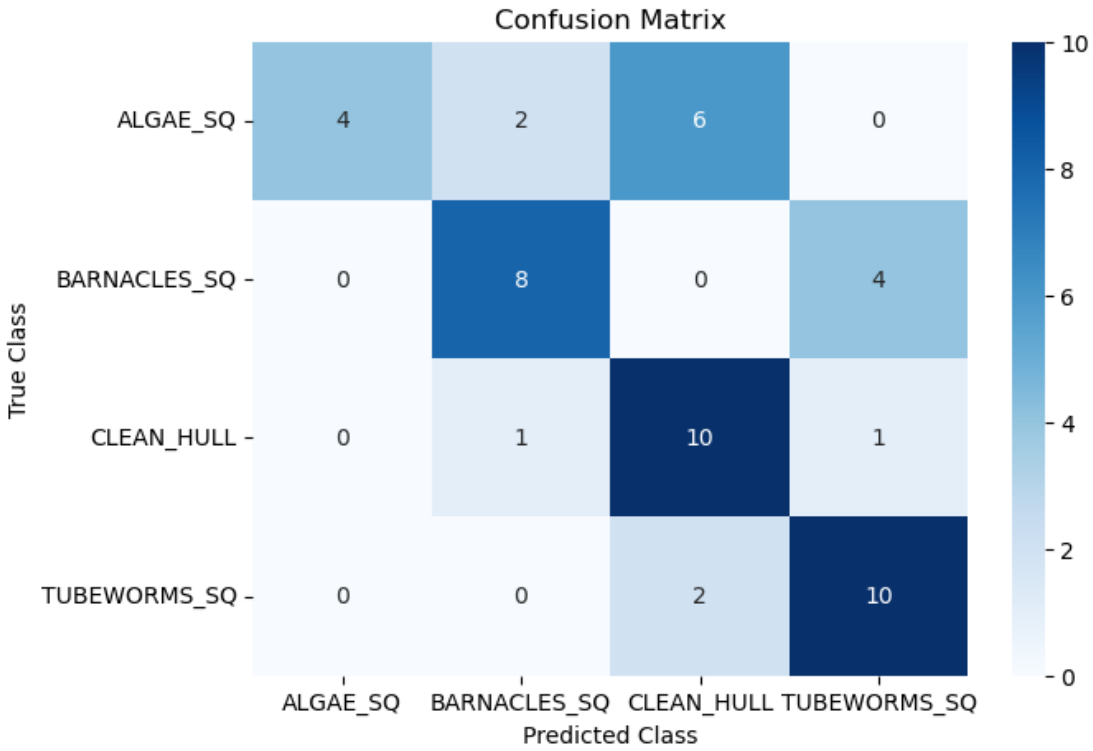

Figure 7: *Confusion matrix using validation set*

Train using 100% of the training data

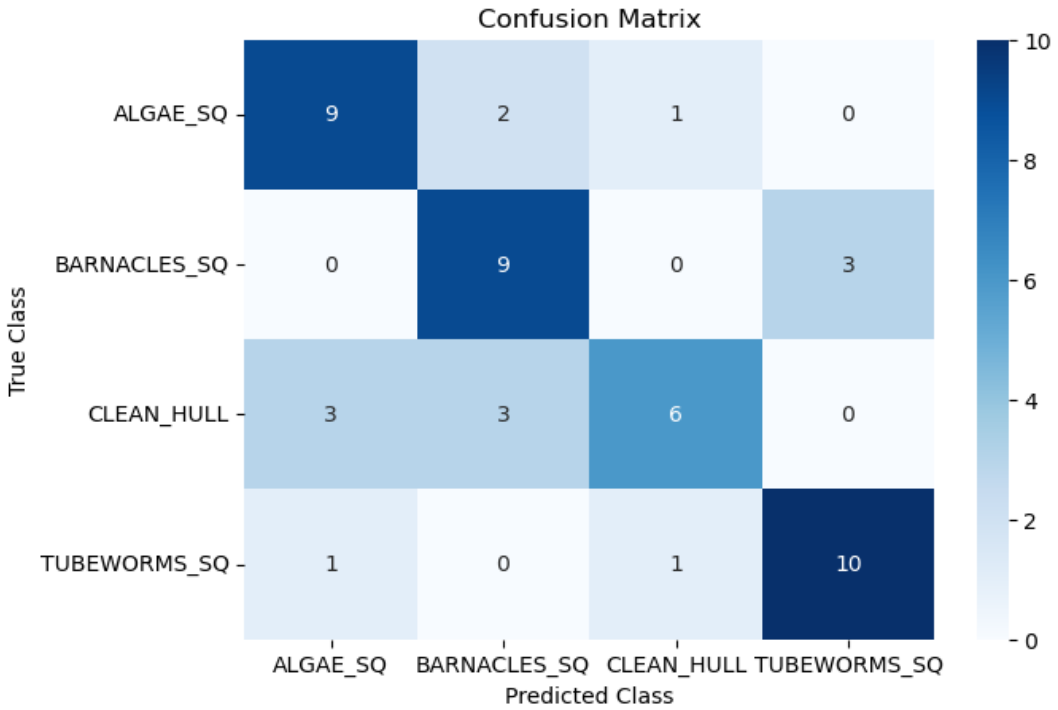

***Figure 8**: Confusion matrix using 100% of the training data*

Fine Tuning Training

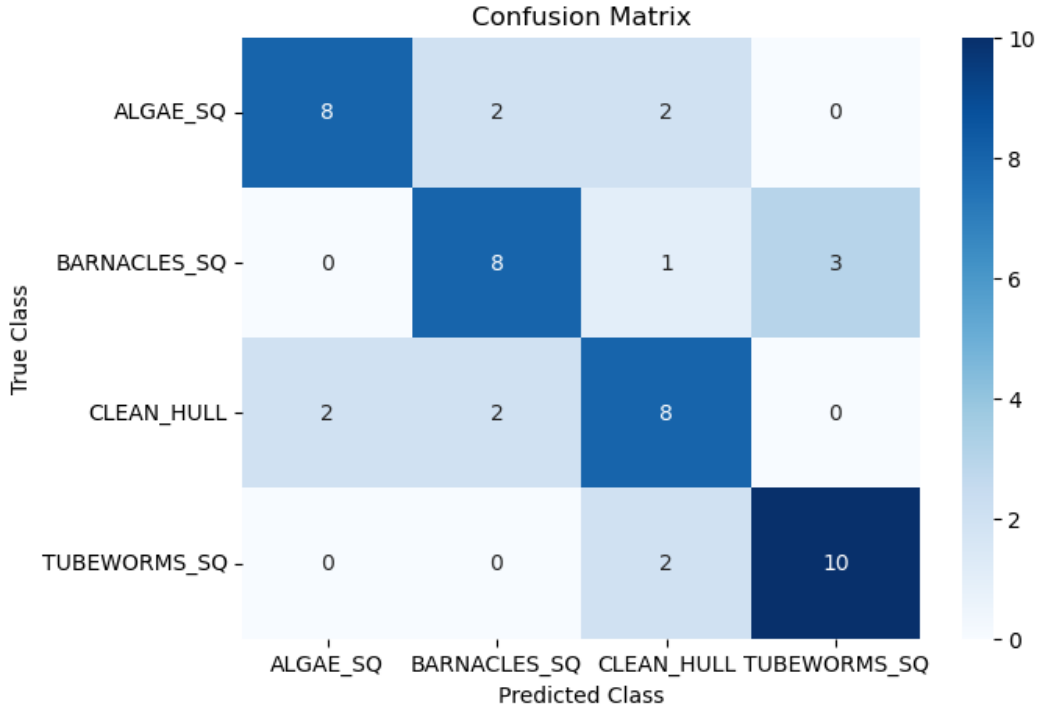

***Figure 9**: Confusion matrix after fine tuning training set*

EfficientNet70 training

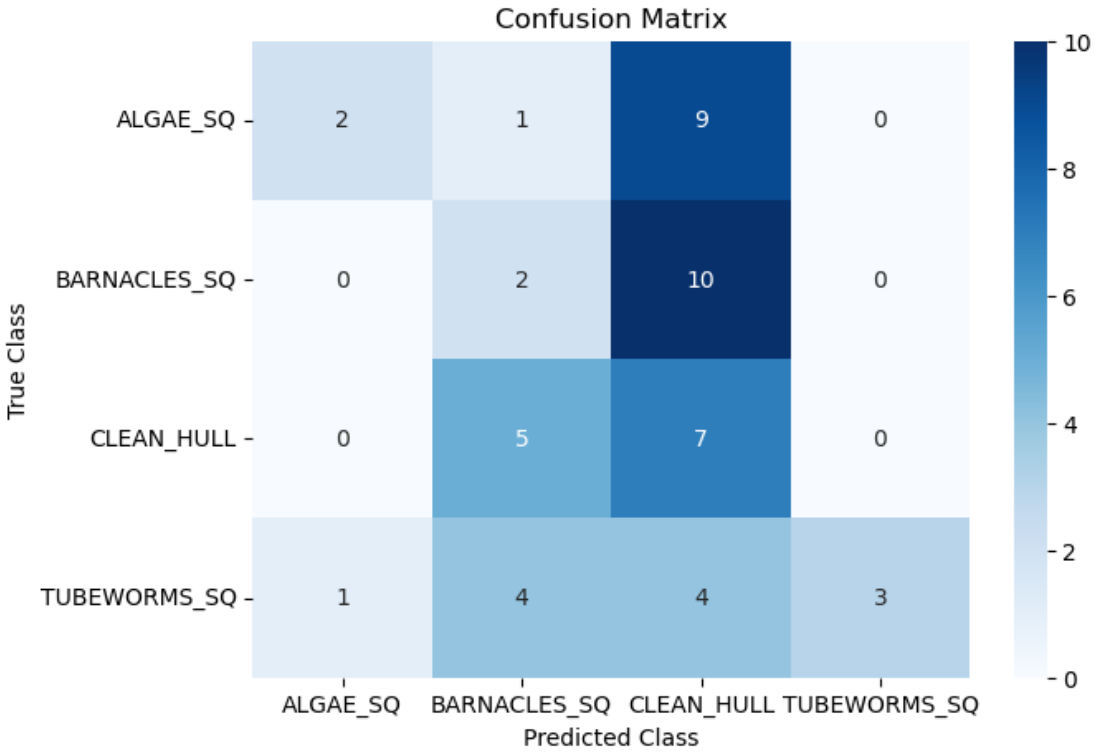

***Figure 10**: Confusion matrix of EfficientNet70 model*

Results & Quantitative Analysis

After executing all the appropriate training trials, we can conclude to the below statements:

- **Training on 80%** of the training dataset by using validation set for monitoring the training procedure, we achieved a moderate accuracy value by evaluating the model on our testing dataset. Half of the classes achieved a *Recall* value greater than 80% meaning that the model detected them accurately. However, ALGAE_SQ – CLEAN_HULL fouling categories failed to be classified properly as the model could not discern between them. That is, ALGAE_SQ images classified as CLEAN_HULL, and vice-versa.
- **Training the 100%** of the training dataset, model achieved a higher enough accuracy value on testing data. Compared to the previous results, ALGAE_SQ images classified much better achieving a *Recall* value of 75%. On the other hand, CLEAN_HULL images again suffer a moderate to low *Recall* value of 50%, meaning that only half of the classified correctly.
- **Fine tuning** training encompasses unfreezing of *ResNet50* internal layers. At the same time, by applying lower *learning rate* and a smaller number of epochs to avoid overfitting, model manages to achieve the best results.

Discussion, Comments and Future Work

During execution of the project, we encountered several obstacles, especially in data collection, cleaning and processing. Also, some issues were met when attempting to improve our model's performance.

Specifically:

- First and probably the most impactful problem was the limited size of the dataset. In deep learning models, as the one we utilized here, the dataset is expected to be plentiful; this is where the capabilities of these models are demonstrated, after all. A big dataset for training can lead to more accurate predictions and help the model better converge. In this case, it proved to be difficult to create a big dataset, primarily due to time constraints and limited access, but also due to the blurry quality of many images on the original dataset we drew content from.
- Additionally, many images on the original dataset contained multiple types of fouling simultaneously. For example, algae was present, in parallel with barnacles. This created the problem that the model was getting confused during training, due to our approach of single-class classification modeling. Thus, we attempted to split the different types of fouling during the cropping process, to create separate images that would be more appropriate for the model, without necessarily creating a "fake" image, rather focusing on specific fouling in each case.

To further enhance the capabilities and usage of the model, future developments should focus on 3 pillars:

1) Multi-class model development: The model can be expanded to identify multiple fouling types and their severity. One input, being the single images from the ROV or the divers, will lead to two outputs indicating both all the existing types of interest and the severity of them (numerical scale). This will significantly decrease human error in both detection and severity estimation from diver's side, leading to more reliable data.
2) Location detection: This involves the introduction of a segmentation model that will be able to identify regions of interest within image. This will allow the system to distinguish between different parts of the vessel, particularly the hull (verticals and flat bottom) and the propeller, linking the location with the type of fouling and severity score.

3) Automated data entry: Aiming to bypass the manual data entry in company's business network, the system could be designed to integrate direct data transfer from the classification and location detection models into the company's network and database. A data pipeline could be created that will format the results according to company's data entry standards.

Members/Roles

Our team consists of four members**: Koromilas Dimitrios** (Computer Scientist, working at a regulatory authority), **Lampos Nikolaos** (Mechanical Engineer, working at a renewables energy company), **Moraitis Anastasios** (Marine Engineer, working at a maritime company) and **Stamatis Ioannis** (Portfolio Manager, working at an investment company).

We made a collective decision to evenly split the tasks of the project and contribute equally, avoiding strict task division. Given that this was our first time encountering the topic on such a scale, we aimed to gain a comprehensive understanding of each step by collaborating closely on every aspect. While we worked as a cohesive unit, each team member contributed more significantly in specific areas based on their expertise.

Our diverse undergraduate and professional backgrounds proved to be a key advantage, enabling us to examine the issue from multiple standpoints and incorporate solutions from multiple disciplines.

Time Plan

The time plan we used is presented below and can be split into five basic axes: analyzing the problem and understand its parameters, data collection and cleaning, model selection and implementation, running test and getting results and finally writing the report.

Date	Actions
July 2024	Discussion about our case. Research and understanding its parameters.
	Trying to find a suitable and adequate dataset.
Early and Mid-August 2024	Discussion with a Greek maritime company for the supply of the dataset.
	Decision to use the dataset acquired by the company.
Late August 2024	First steps of understanding, cleaning and organizing the dataset.
Early September 2024	Finalizing the cleaning and the needed transformations of the images dataset.
Mid-September 2024	Discussion, research and tests of various models suitable for the project.
Late September 2024	Reaching final results and conclusions.
	Writing of the report.

References

Evelyn J. Mannix, SusanWei, Bartholomew A. Woodham, PeterWilkinson & Andrew P. Robinson, "Automating the assessment of biofouling in images using expert agreement as a gold standard", 2021, *Nature (https://www.nature.com/articles/s41598-021-81011-2#Sec1)*

Christos Petridis, Michael Vassilakopoulos, "Detecting Hull Fouling using Machine Learning Algorithms trained on Ship Propulsion Data to Improve Resource Management and Increase Environmental Benefits", 2024, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (https://isprs-annals.copernicus.org/articles/X-4-W4-2024/185/2024/isprs-annals-X-4-W4-2024-185-2024.pdf)

Andrea Farkas, Nastia Degiuli, Ivana Martic, Roko Dejhalla, "Impact of Hard Fouling on the Ship Performance of Different Ship Forms", *Journal of Marine Science and Engineering,* 2020 (https://www.mdpi.com/2077-1312/8/10/748#:~:text=The%20differences%20in%20the%20impact,for%20the%20self%2Dpropulsion%20point)

European Commission, "*Reducing emissions from the shipping sector"* (https://climate.ec.europa.eu/eu-action/transport/reducing-emissions-shipping-sector_en)

Carbon Market Watch, *"FAQ: The EU ETS for shipping explained"* (https://carbonmarketwatch.org/2023/12/06/faq-the-eu-ets-for-shipping-explained/)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", 2015 (https://arxiv.org/pdf/1512.03385)

https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

https://keras.io/guides/transfer_learning/

https://keras.io/examples/vision/image_classification_efficientnet_fine_tuning/

https://medium.com/@nitishkundu1993/exploring-resnet50-an-in-depth-look-at-the-model-architecture-and-code-implementation-d8d8fa67e46f

https://medium.com/@preeti.rana.ai/using-a-pre-trained-image-classification-model-e-g-3ee5cfc7b896

https://en.wikipedia.org/wiki/Convolutional_neural_network

https://builtin.com/machine-learning/relu-activation-function

https://blog.roboflow.com/what-is-resnet-50/

https://databasecamp.de/en/ml/resnet-en

Appendix

Augmentation (machine learning)

Techniques used to enhance the size and diversity of a training dataset, often involving transformations like rotation, flipping, or cropping to improve model robustness.

Batch size (machine learning)

The number of training examples utilized in one iteration during the training of a machine learning model. It impacts the speed and stability of the training process.

Carbon Intensity Indicator (CII)

A metric used to assess the carbon emissions of a vessel per ton of cargo transported, helping to measure environmental impact in shipping.

Convolutional Neural Networks (CNNs)

A class of deep learning models particularly effective in processing and analyzing visual data, often used in image classification tasks.

Energy Efficiency Design Index (EEDI)

A regulatory measure that mandates a minimum energy efficiency level for new ships, promoting environmentally friendly shipping practices.

Energy Efficiency Existing Ship Index (EEXI)

A framework that establishes energy efficiency requirements for existing ships, encouraging upgrades to reduce greenhouse gas emissions.

Epochs (machine learning)

One complete cycle through the entire training dataset during the training of a machine learning model, affecting how well the model learns.

International Maritime Organization (I.M.O.)

A specialized agency of the United Nations responsible for regulating shipping and ensuring safe, secure, and environmentally sound shipping practices.

Keras (machine learning)

An open-source software library for building neural networks, providing a user-friendly interface for designing and training deep learning models.

Machine Learning (ML) algorithms

A computational method that enables machines to learn from data and improve their performance on a specific task without being explicitly programmed.

Maritime charterers

Companies or individuals that hire a vessel for the transportation of goods, determining the operational and economic terms of the charter agreement.

Model's accuracy (machine learning)

A metric that evaluates how well a machine learning model performs, typically defined as the proportion of correct predictions made by the model.

Normalization (machine learning)

A preprocessing technique that scales input data to a standard range, improving the efficiency and performance of machine learning algorithms.

Remotely Operated Vehicle (R.O.V.)

An unmanned, remotely controlled underwater vehicle used for inspection, maintenance, and data collection in marine environments.

Ship Biofouling

The accumulation of microorganisms, plants, algae, and animals on the submerged surfaces of ships, which can lead to increased drag and reduced fuel efficiency.

Ship's main engine

The primary propulsion unit of a ship, responsible for generating the power needed for movement and maneuvering in water.

Ship's propulsive power

The total power output of a ship's engines, which determines its speed and efficiency in navigating through water.

Test dataset (machine learning)

A subset of data used to evaluate the performance of a trained machine learning model, ensuring it generalizes well to unseen data.

Train dataset (machine learning)

The portion of data used to train a machine learning model, allowing it to learn patterns and make predictions.

Validation dataset (machine learning)

A dataset used to fine-tune the parameters of a machine learning model and assess its performance during training, helping to prevent overfitting.