Ship Classification Using Machine Learning

1. INTRODUCTION

The maritime industry stands as a cornerstone of global trade and transportation networks, facilitating the movement of goods, people, and resources across the world's oceans. In this vast and dynamic environment, the accurate identification and classification of ships is paramount for a multitude of critical applications. From bolstering maritime security and environmental conservation efforts to optimizing traffic management and supporting research endeavors, the need for automated ship classification systems has never been more pressing.Efficiently identifying and classifying ships is essential for tasks such as maritime security, environmental monitoring, and traffic management. This project aims to develop a robust system for automated ship classification utilizing state-of-the-art deep learning techniques.

1.1 Purpose

Here are five key use cases that highlight the purpose and potential impact of the ship classification project:

.

Maritime Security and Surveillance:

.

- Purpose: To enhance maritime security by accurately identifying and classifying vessels in real-time.
- Use Case: Government agencies and naval forces can use the system to monitor and identify ships in sensitive areas, swiftly detecting any suspicious or unauthorized activities.

.

Environmental Monitoring and Conservation:

.

- Purpose: To track and mitigate environmental impacts of shipping activities.
- Use Case: Environmental agencies and conservation organizations can utilize the system to monitor vessel movements, identify potential pollution sources, and respond promptly to oil spills or other environmental incidents.

.

Port and Traffic Management:

.

- Purpose: To optimize traffic flow and safety within busy ports and waterways.
- Use Case: Port authorities and maritime traffic controllers can employ the system to monitor and manage vessel movements, preventing congestion and reducing the risk of collisions.

2. LITERATURE SURVEY

2.1 Existing problem

Ship classification challenges have witnessed successful applications of deep learning techniques, notably Convolutional Neural Networks(CNNs), invariousresearchendeavors. Transferlearningfrompre-trainedmodelssuch as VGG19, ResNet50, and Inception has proven effective for classifying ships into different categories. These models demonstrate the capability to accurately distinguish between various types of vessels, aiding in timely identification andmanagement.Leveragingpre-trainedmodelslike VGG19, ResNet50, and Inceptionprovidesasolidfoundationfor ship classification tasks. Fine-tuning these models on ship-specific data allows for efficient learning of ship-related features.

High-quality datasets are essential for training Machine learning models that classify ships, as previous studies have shown. Model development has benefited greatly from the availability of existing datasets, such as the ship images dataset and others. To increase the resilience and generalisation of models, researchers have used data preprocessing techniques such image scaling, normalisation, and data augmentation. In order to address frequent obstacles in ship classification, such as uneven class distribution and differences in lighting and background, some research has looked into domain-specific data augmentation techniques.

The literature emphasises the importance of developing accurate models, but it also emphasises the necessity of creating deployment interfaces that are easy to use. The combination of deep learning models with approachable applications to enable sea workers to quickly identify ships traffic has been covered in a number of publications. These apps must to work with web browsers, mobile devices, and Internet of Things gadgets, and they ought to give consumers dependable and timely results. In order to guarantee the system's practical applicability in actual ship classifier settings, interpretability and user input integration are frequently explored in this domain's literature.

In summary, the body of current research offers a solid framework for creating a Machine learning-based system for classifying ships types. It emphasises the crucial roles that high-quality datasets, data preparation, model architecture, and user-friendly deployment have in improving ship management in the port industry and helping ship users.

2.2 References

1. https://www.kaggle.com/code/teeyee314/classification-of-ship-images

2. https://github.com/topics/ship-classification

3. https://www.mdpi.com/2072-4292/11/4/419

4 . https://www.irclass.org/marine/ship-classification/

2.3 Problem Statement Definition

The ship classification project seeks to address the critical need for accurate and efficient identification of vessels within the maritime industry. Existing methods face challenges due to the diverse range of ship types, variable environmental conditions, and the requirement for real-time processing. Traditional rule-based approaches struggle to adapt to these complexities, necessitating the development of an automated system. The objective of this project is to design and implement a state-of-the-art ship classification system using deep learning techniques, particularly Convolutional Neural Networks (CNNs), capable of swiftly and accurately categorizing ships into predefined classes. This system aims to revolutionize maritime operations by enhancing security, traffic management, environmental monitoring, and providing valuable data for research and policy analysis.

Important elements of the issue statement:

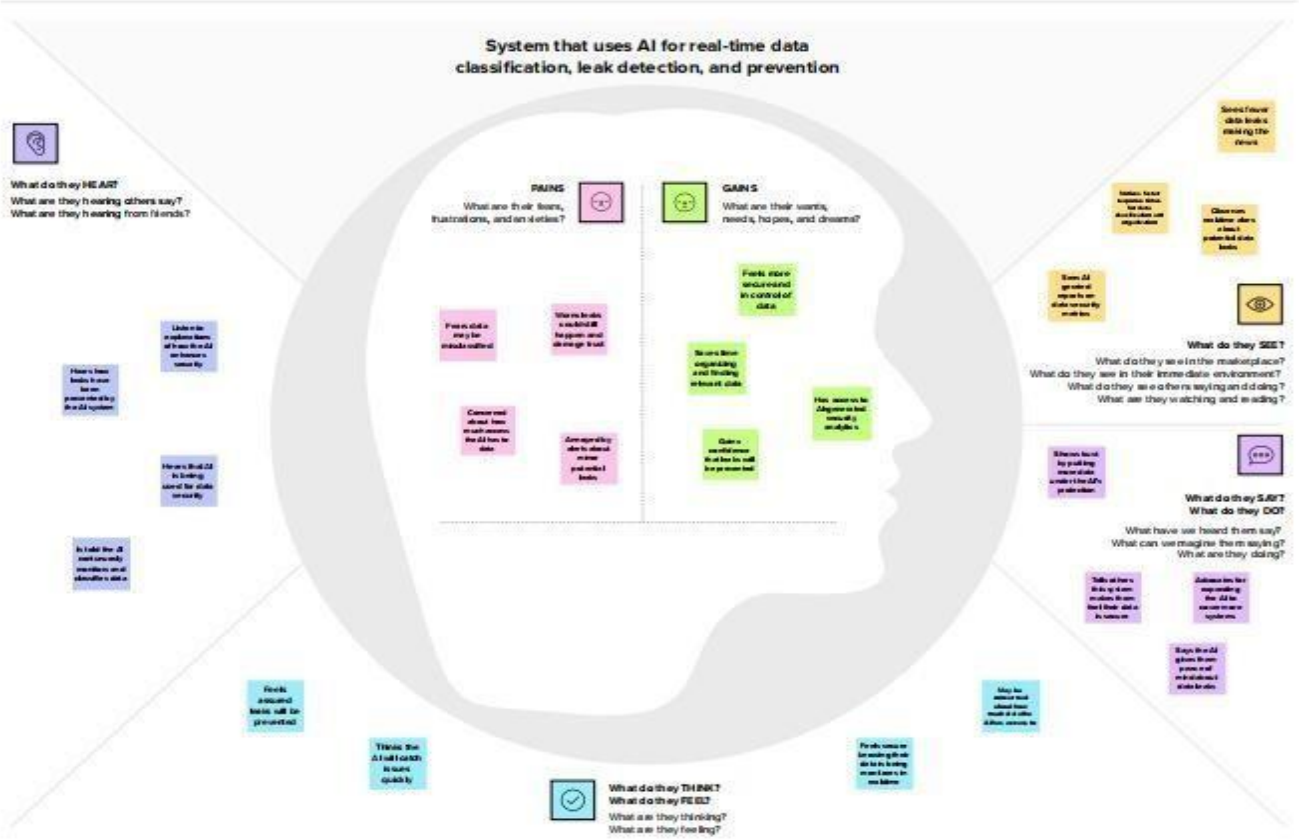
The main challenge is developing a trustworthy system for recognising and categorising the several images that impact ships classifier, including cargo , passenger,milatary . The technology should be able to distinguish between their types that are big and small and those that are boats, and it should give precise details on the kind of ship that is present.

Timeliness and Precision: The system should deliver accurate and quick ship classification data so that users can begin managing and adjusting their traffic that have been recognised right away. This takes care of the requirement for early accidents identification, which can identified very fast give information to the ship operators.

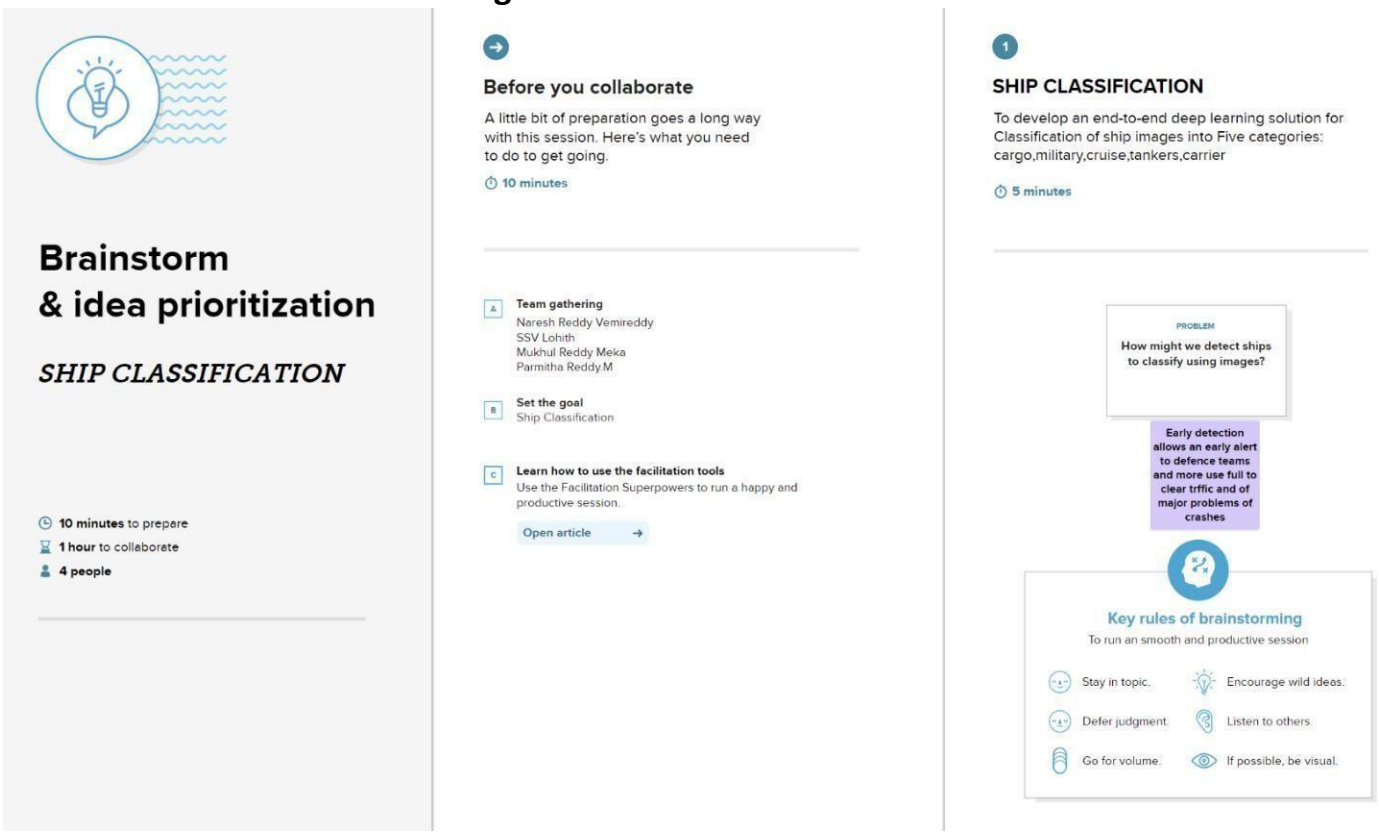
User-Friendly Interface: To solve this problem effectively, a user-friendly interface or application should be developed, allowing operators to easily upload images of their ship images and receive ship classification results. The system should be intuitive and adaptable to the technological and educational background of the users.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Ideation & Brainstorming



2

Brainstorm

The proposed solution involves the use of convolutional neural networks(CNN) to extract relevant features from the input images and classify them to 5 categories

10 minutes

Person 1

Collect a diverse dataset of ship images, including all types and various categories based on size. Proper data preprocessing and augmentation are essential to ensure data quality.

Implement image processing techniques to enhance the quality of images, and extract relevant features such as color, texture, and shape to improve the classification accuracy.

Person 2

Develop a user- friendly interface, either a mobile app or a web platform, for defence team or users to easily upload images and receive the ship data classification results.

Explore the possibility of real-time monitoring using IoT devices and cameras on towers in seas to continuously assess the traffic of the ships.

Person 3

Extend the model to classify multiple number of ships within a single image

Explore techniques for identifying not only the presence of ship but also its location on the sea, as this can aid in finding the correct location.

Person 4

Develop and train deep learning models such as Convolutional Neural Networks (CNNs) for ship classification.

Provide a feedback support to understand more about the types of ships

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

⏱ 20 minutes

Data Collection and Preparation:

1. Collect a diverse dataset of ship images, including all big and small ships. 2. Implement data preprocessing and augmentation techniques to improve data quality.

Design model and Architecture of the user interface and use proper methodologies

Image processing and deep learning: Image processing and deep learning algorithms can be used to analyze images of ships and identify types at an early stage.

Train the model with various data sets so that there will be no error while detecting type of ship

Educate the port gurds about the issue on traffic and educate the portgurds about how to use the technology

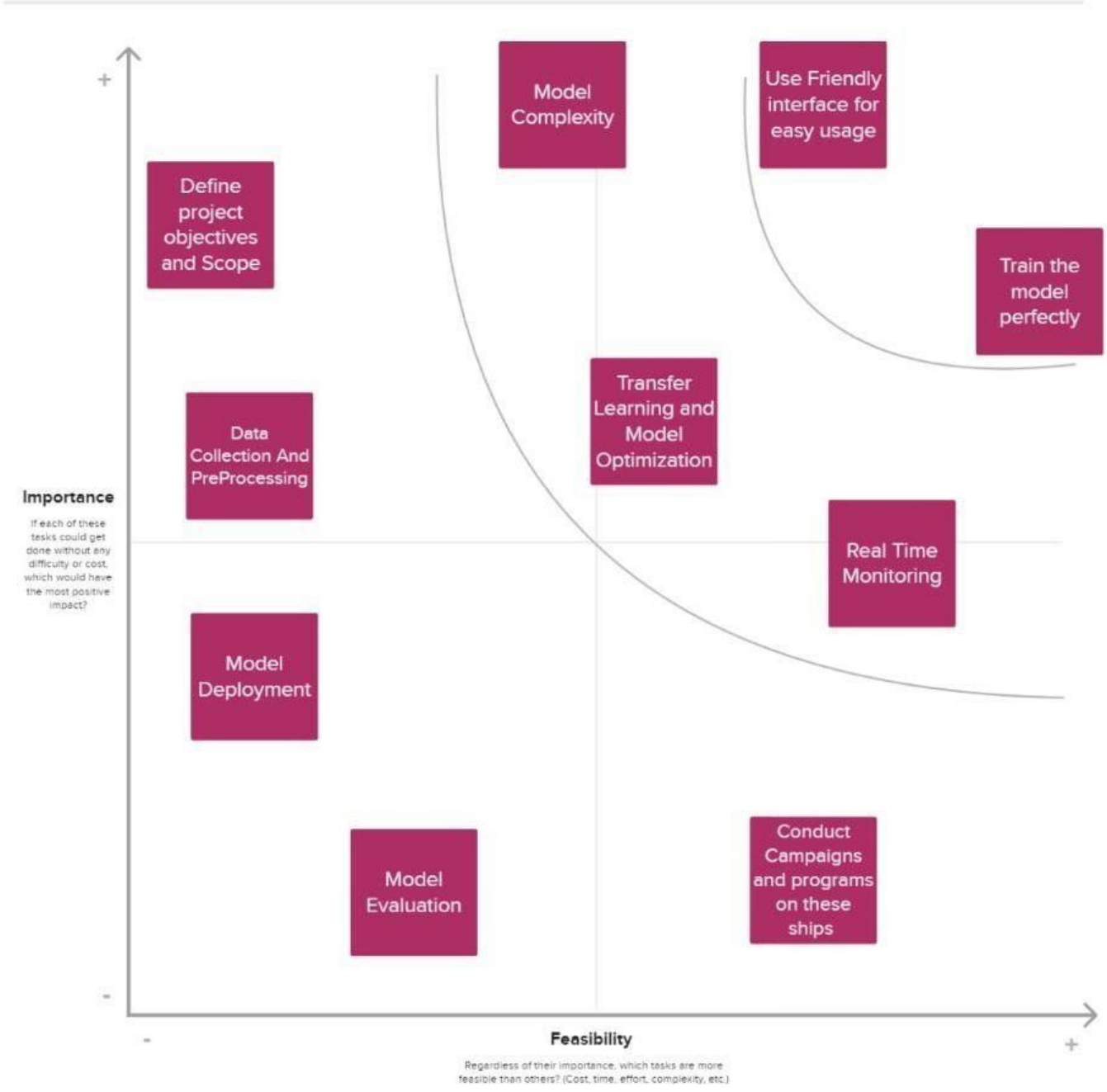
Deploy the model and Create a user-friendly interface, either a mobile app or web platform and observe the traffic and oberve the results

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

Image Upload and Processing: In order to classify ships, users (ship operators) should be able to upload photos of ship images to the system. To guarantee that these photographs are compatible with the Machine learning model, it should preprocess them, resizing and normalising them as needed.

Ship Classification: The system's main job is to correctly identify type of the ship from the supplied photos. It should be able to distinguish between distinct types of ships , giving each case a unique type of ships identifier.

User Interface: A user-friendly interface that is compatible with PCs, tablets, and smartphones must be provided by the system. It ought to offer a simple and intuitive user interface to suit consumers with varying degrees of technological expertise.

Model Interpretability: The system ought to provide justifications or illustrations of the model's decision-making procedure, giving users an understanding of the rationale behind the classification of a certain ailment. This feature improves user comprehension and trust.

Scalability and Integration: To enable wider acceptance and influence within the marine sector , make sure the system is scalable and compatible with other marine technology and information systems.

4.2 Non-Functional requirements

Accuracy and Precision: For operators to receive trustworthy findings, the ship classification system needs to attain a high degree of accuracy. It should also minimise false positives and false negatives by precisely identifying ships.

Response Time: In order to enable operators to take rapid action, the system should provide ship categorization results as soon as possible, ideally in real-time or within a few seconds of image input.

Security and Privacy: Put security measures in place to guard user information and guarantee the privacy of photos that users post. Images and personal data should be treated carefully and in accordance with applicable data protection laws.

Robustness and Reliability: Strong and dependable, the system should be able to adapt to changes in ship distances , lighting, and image quality. In the event of an unplanned outage, failover procedures ought to be included.

Scalability and Performance: Make sure the system is capable of managing a sizable volume of user requests and expanding to accommodate a rising user base without experiencing performance issues. Optimization and load balancing ought to be implemented.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

Data Collection: Obtaining a wide range of ships images datasets from image archives, surveys, and other sources is the task of this phase. Following collection, the photos are kept in a raw data repository**.**

Data Pre-processing: Images of raw ship images are pre-processed in order to get them ready for model training. To improve dataset diversity, this can involve resizing photographs, normalising pixel values, and using data augmentation techniques.

Model Training: This step involves using the pre-processed data to train a machine learning model so that it can identify different type of ships. For later use, the trained model is stored.

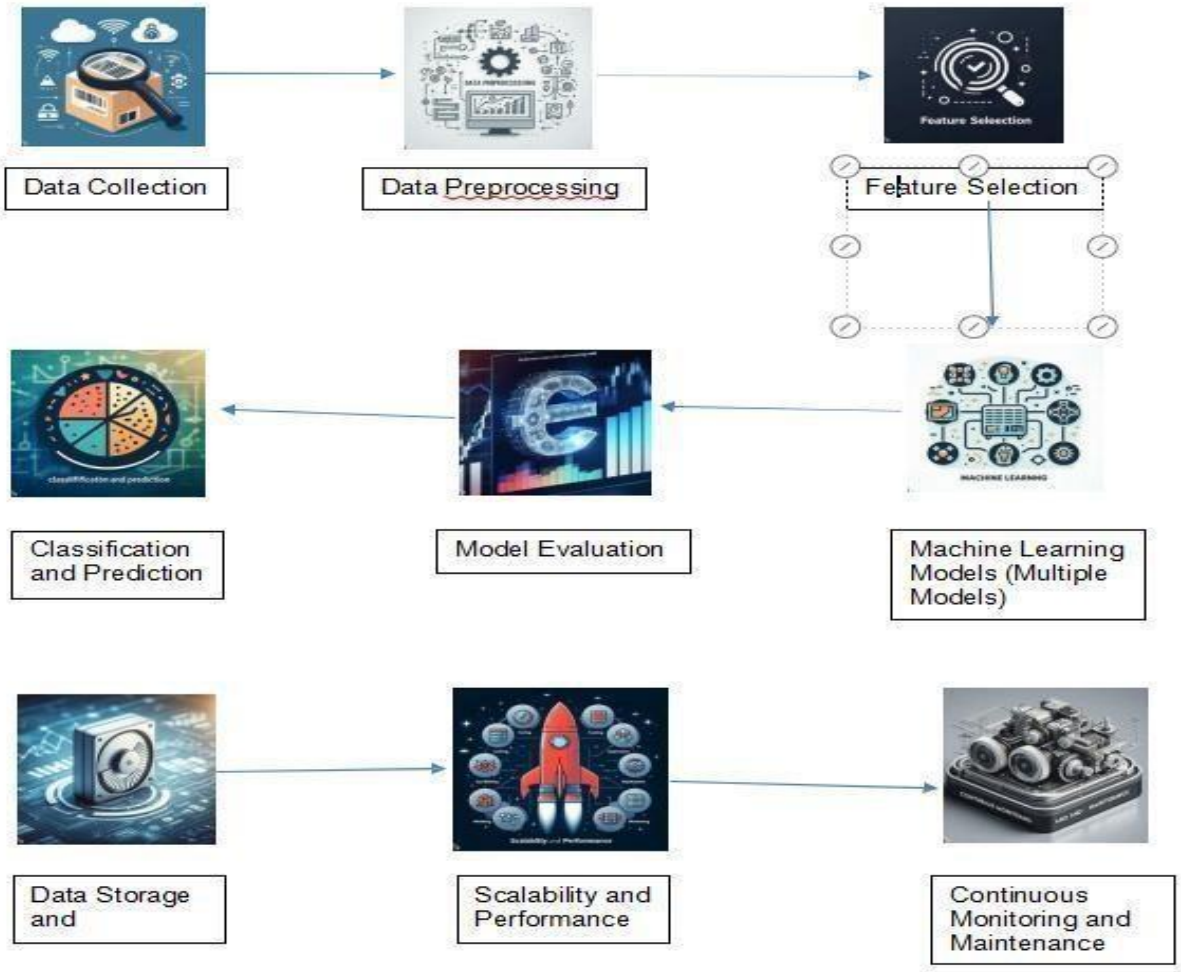
Model Evaluation: The performance of the trained model is assessed using a separate dataset not used in training, to gauge its accuracy, sensitivity, and specificity in ship classification.

Model Deployment: This step involves deploying the trained model to make it accessible for real-world ship classification applications, either on local devices or in the cloud.

User Interaction: Through an intuitive application or API, end users engage with the deployed model, allowing them to upload photographs of ship images for ships identification and obtain timely results.

Data Flow Diagram:

Solution Architecture Diagram:



User Stories:

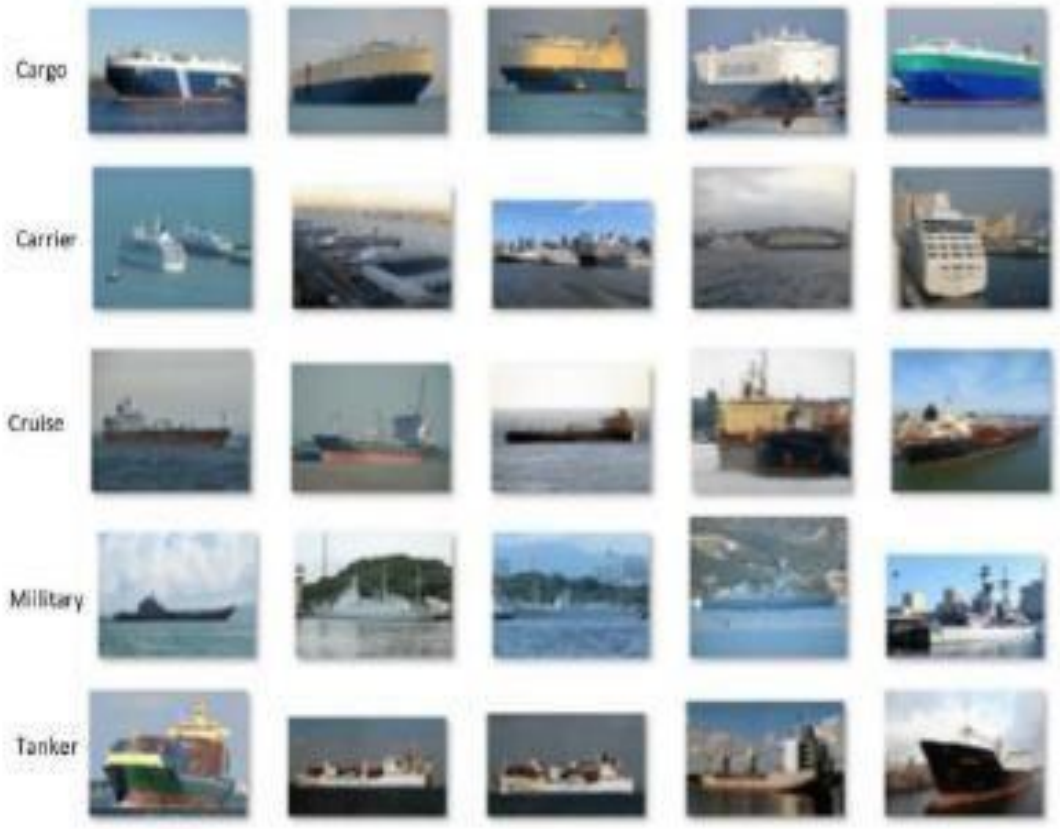
Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Ships' Owners and Operators:	Design and Construction:	USN-1	Ship owners and operators rely on ship classification to ensure that their vessels are designed and constructed to meet international standards for safety, reliability, and performance. This helps in reducing risks associated with ship operations.	access the ship classification system via a user-friendly interface, either through a web application or a mobile app.	High	Sprint-1
Shipbuilders and Shipyards:	Testing and Certification:	USN-2	Classification societies may oversee and certify various aspects of ship construction, including materials, welding processes, and testing procedures, to ensure compliance with industry standards.	ship classification system should be equipped with an extensive database that covers a wide range of ship images, including both common and rare occurrences, to provide comprehensive information for research purposes.	High	Sprint-1
Maritime Regulators and Authorities:		USN-3	I require a user-friendly ship classification societies work closely with regulatory bodies to ensure that ships meet safety and environmental standards, helping to mitigate risks and protect marine environments.	The ship classification system should provide timely and accurate information on various ship images, including their sizes, colour, and appropriate management strategies, enabling agronomists to make informed decisions and provide effective guidance to marine officer .	Medium	Sprint-2

Naval and Coast Guard Agencies:	Search and Rescue Operations:	USN-4	ship classification for the Rescue operations are more important by comparing the images with ship classifier we can have an estimation of ship type ship size etc.	It should have the capability to Medium handle a large volume of data and provide real-time analysis,enabling military officers to make informed decisions and implement timely interventions for traffic	Medium	Sprint-1

Solution Architecture

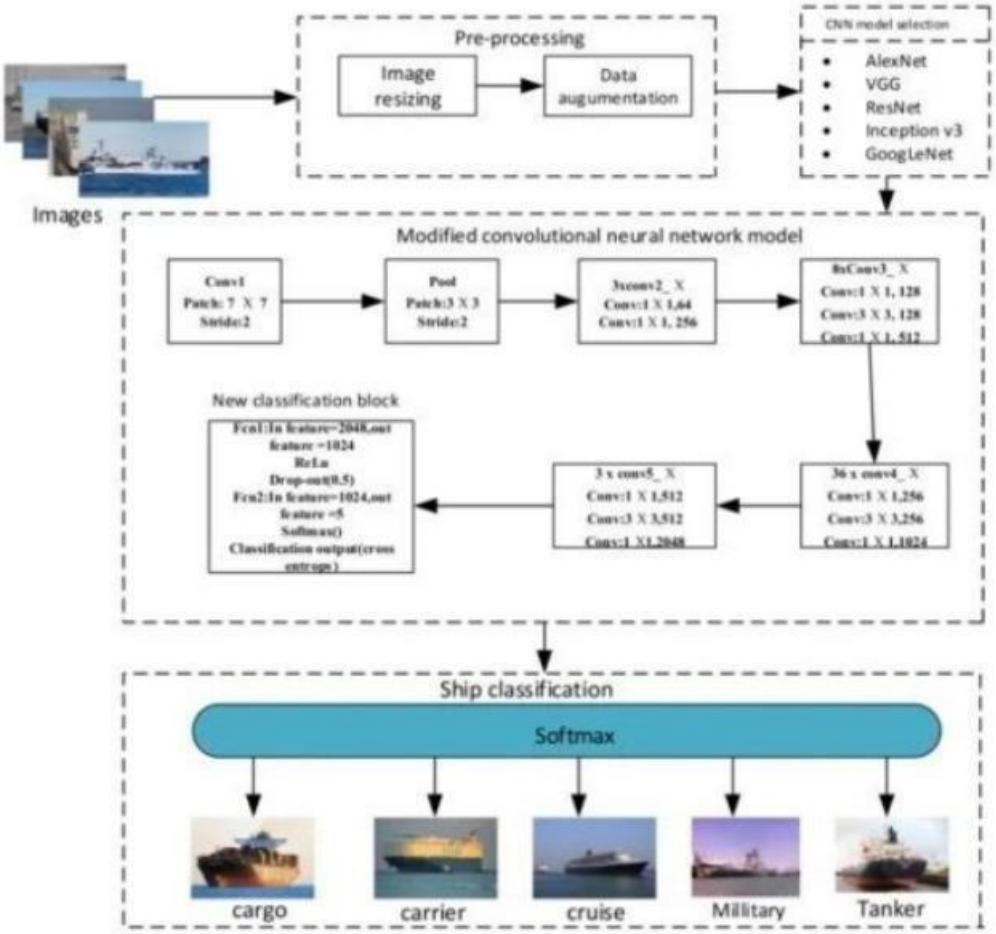
Diagram:



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture

Solution Architecture Diagram:



6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Ship Classification	USN-1	As a user, I can upload images of different ships classified by various factors such as shape model and working in Ship Classification	3	High	Mukhul Reddy
Sprint-1	Ship Classification	USN-2	As a user, I can view the predicted Ship class and its probability for the uploaded image of Ships.	2	High	Parmitha Reddy
Sprint-2	Ship Classification	USN-3	As a user, I can access historical data and analysis of previously used Ship models.	2	Medium	Naresh Reddy
Sprint-2	Ship Classification	USN-4	As a user, I can provide feedback on the accuracy of the Ship classification for continuous model improvement.	1	Medium	
Sprint-3	Ship Classification	USN-5	As a user, I can receive recommendations on management strategies based on the classified images of Ships.	3	Low	

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022
Sprint-2	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022
Sprint-3	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022
Sprint-4	20	5 Days	23 Oct 2022	27 Oct 2022	20	27 Oct 2022

7. CODING & SOLUTIONING

Activity 1.1: Importing the libraries

In [2]:
```
import os
import gc
import sys

import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import skimage
from skimage.feature import hog, canny
from skimage.filters import sobel
from skimage import color

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split

from keras import layers
import keras.backend as K
from keras.models import Sequential, Model
from keras.preprocessing import image
from keras.layers import Input, Dense, Activation, Dropout
from keras.layers import Flatten, BatchNormalization, Conv2D
from keras.layers import MaxPooling2D, AveragePooling2D, GlobalAveragePooling2D
from keras.applications.imagenet_utils import preprocess_input
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications import ResNet50
from tf_explain.core.activations import ExtractActivations
from tf_explain.core.grad_cam import GradCAM

from PIL import Image
from tqdm import tqdm
import random as rnd
import cv2
from keras.preprocessing.image import ImageDataGenerator
from numpy import expand_dims

!pip install livelossplot
from livelossplot import PlotLossesKeras

%matplotlib inline
```

```
Collecting livelossplot
  Downloading livelossplot-0.5.5-py3-none-any.whl (22 kB)
Requirement already satisfied: bokeh in /opt/conda/lib/python3.7/site-packages (from livelossplot) (2.4.2)
Requirement already satisfied: ipython==7.* in /opt/conda/lib/python3.7/site-packages (from livelossplot) (7.30.1)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-packages (from livelossplot) (3.5.1)
Requirement already satisfied: numpy<1.22 in /opt/conda/lib/python3.7/site-packages (from livelossplot) (1.20.3)
Requirement already satisfied: backcall in /opt/conda/lib/python3.7/site-packages (from ipython==7.*->livelossplot) (0.2.0)
Requirement already satisfied: matplotlib-inline in /opt/conda/lib/python3.7/site-packages (from ipython==7.*->livelossplot)
(0.1.3)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.7/site-packages (from ipython==7.*->livelossplot) (4.8.0)
Requirement already satisfied: setuptools>=18.5 in /opt/conda/lib/python3.7/site-packages (from ipython==7.*->livelossplot) (5
9.5.0)
```

Loading Dataset

We'll use here the Pandas to load the dataset into memory

Loading Dataset

We'll use here the Pandas to load the dataset into memory

In [3]:
```
main_path = 'images/'
```

In [4]:
```
main_df = pd.read_csv('train.csv')
```

In [5]:
```
paths = os.listdir(main_path)
```

In [6]:
```
main_df['path'] = main_path + main_df['image']
```

In [7]:
```
categories = list(main_df['category'])
categorys = {1:'Cargo', 2:'Military', 3:'Carrier', 4:'Cruise', 5:'Tankers'}
classes = []
```

In [8]:
```
for category in categories:
    classes.append(categorys[category])
```

In [9]:
```
main_df['classes'] = classes
```

In [10]:
```
main_df.head()
```

Out[10]:

	image	category	path	classes
0	2823080.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo
1	2870024.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo
2	2662125.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military
3	2900420.jpg	3	../input/game-of-deep-learning-ship-datasets/t...	Carrier
4	2804883.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military

In [11]:
```
test_df = pd.read_csv('test_ApKoW4T.csv')
```

In [12]:
```
test_df['path'] = main_path + test_df['image']
```

Analyzing counts

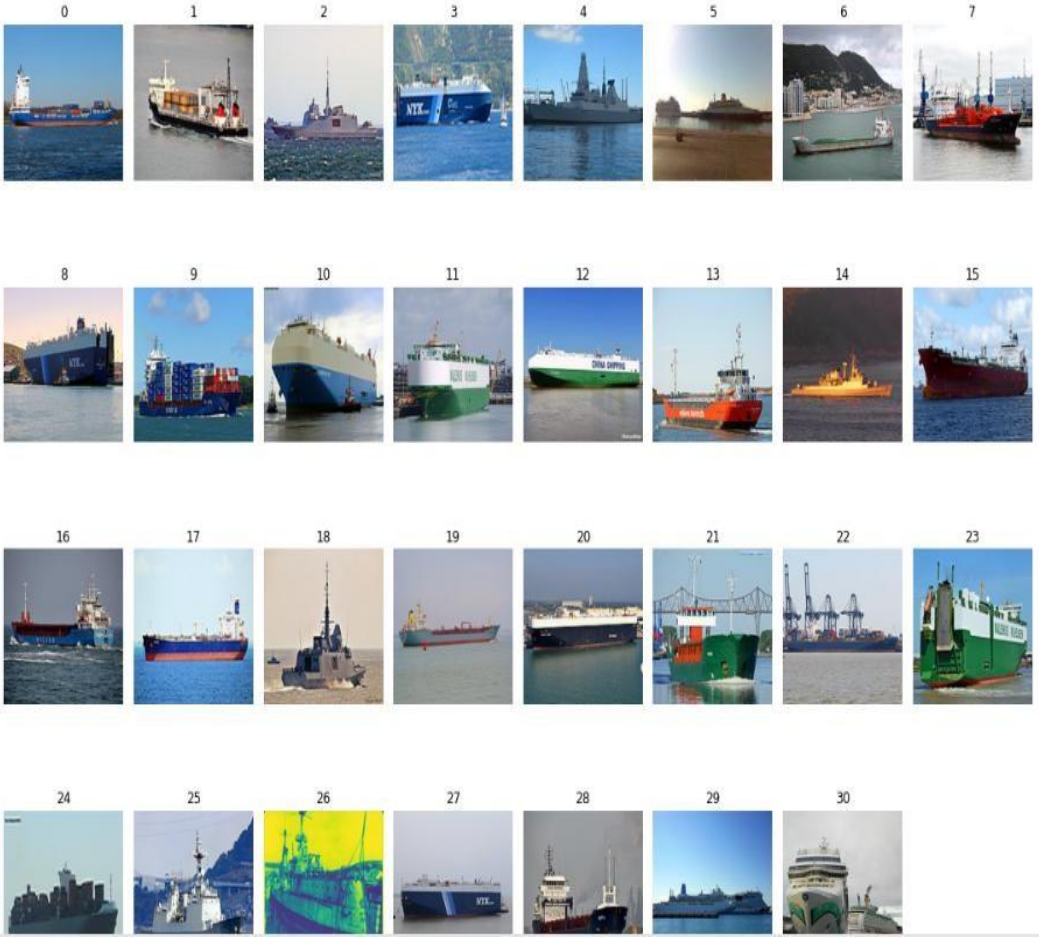
In [13]:
```
print('Train samples: ', len(main_df))
print('Test samples: ', len(test_df))
```

```
Train samples:  6252
Test samples:  2680
```

Glimpses of Dataset

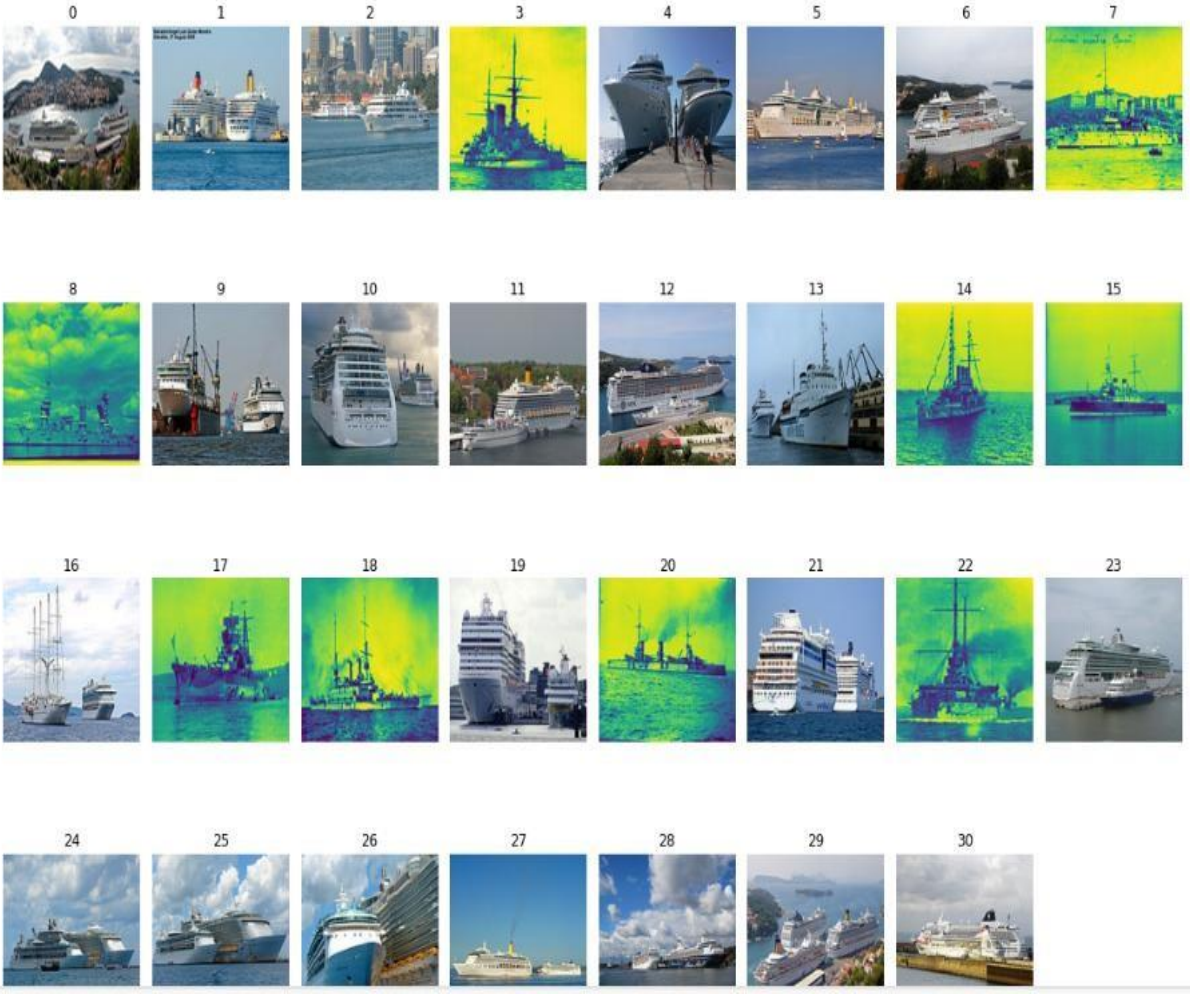
Main Dataset

In [14]:
```
plt.figure(figsize = (15,12))
for idx,image_path in enumerate(main_df['path']):
    if idx==31:
        break
    plt.subplot(4,8,idx+1)
    img = Image.open(image_path)
    img = img.resize((224,224))
    plt.imshow(img)
    plt.axis('off')
    plt.title(idx)
plt.tight_layout()
plt.show()
```



Test Dataset

In [15]:
```
plt.figure(figsize = (15,12))
for idx,image_path in enumerate(test_df['path']):
    if idx==31:
        break
    plt.subplot(4,8,idx+1)
    img = Image.open(image_path)
    img = img.resize((224,224))
    plt.imshow(img)
    plt.axis('off')
    plt.title(idx)
plt.tight_layout()
plt.show()
```



Counts of Ships Categories

In [16]:
```
plot = sns.countplot(x = main_df['classes'], color = '#2596be')
sns.set(rc={'figure.figsize':(15,12)})
sns.despine()
plot.set_title('Class Distribution\n', font = 'serif', x = 0.1, y=1, fontsize = 18);
plot.set_ylabel("Count", x = 0.02, font = 'serif', fontsize = 12)
plot.set_xlabel("Ships classes", fontsize = 15, font = 'serif')

for p in plot.patches:
    plot.annotate(format(p.get_height(), '.0f'), (p.get_x() + p.get_width() / 2, p.get_height()),
       ha = 'center', va = 'center', xytext = (0, -20),font = 'serif', textcoords = 'offset points', size = 15)
```

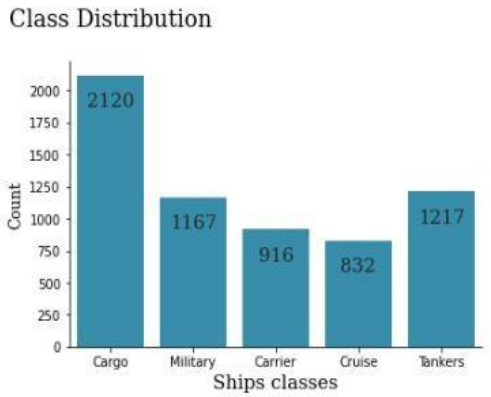


Image Resolutions

In [17]:
```
widths, heights = [], []

for path in tqdm(main_df["path"]):
    width, height = Image.open(path).size
    widths.append(width)
    heights.append(height)

main_df["width"] = widths
main_df["height"] = heights
main_df["dimension"] = main_df["width"] * main_df["height"]
```

```
100%|██████████| 6252/6252 [00:22<00:00, 282.73it/s]
```

Lets see some small images

In [18]:
```
main_df.sort_values('width').head(84)
```

Out[18]:

	image	category	path	classes	width	height	dimension
2766	2833376.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	91	158	14378
3182	2825824.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	96	158	15168
3186	1124952.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	100	158	15800
2732	2855931.jpg	5	../input/game-of-deep-learning-ship-datasets/t...	Tankers	105	158	16590
1576	914599.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	105	158	16590
...	...	...	...	...	...	...	...
3295	1923069.jpg	4	../input/game-of-deep-learning-ship-datasets/t...	Cruise	192	158	30336
1613	2809866.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	192	158	30336
2840	2849133.jpg	3	../input/game-of-deep-learning-ship-datasets/t...	Carrier	192	158	30336
3672	2777434.jpg	4	../input/game-of-deep-learning-ship-datasets/t...	Cruise	192	158	30336
5736	2868088.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo	194	158	30652

84 rows × 7 columns

Let see some large images

In [19]:
```
main_df.sort_values('width', ascending = False).head(84)
```

Out[19]:

	image	category	path	classes	width	height	dimension
0	2823080.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo	210	140	29400
4268	2566396.jpg	4	../input/game-of-deep-learning-ship-datasets/t...	Cruise	210	140	29400
4142	2849919.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo	210	126	26460
4141	2868004.jpg	5	../input/game-of-deep-learning-ship-datasets/t...	Tankers	210	140	29400
4140	2783824.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	210	152	31920
...	...	...	...	...	...	...	...
4111	2856008.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo	210	157	32970
4110	2438564.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	210	140	29400
4109	2713013.jpg	2	../input/game-of-deep-learning-ship-datasets/t...	Military	210	140	29400
4108	2888369.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo	210	140	29400
4107	2860032.jpg	1	../input/game-of-deep-learning-ship-datasets/t...	Cargo	210	139	29190

84 rows × 7 columns

In [20]:
```
main_df.mean()
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduct
ions (with 'numeric_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before
calling the reduction.
  """Entry point for launching an IPython kernel.
```

Out[20]:
```
category         2.657550
width          209.056302
height         138.979687
dimension    29036.629878
dtype: float64
```

Color Analysis

We need to do some color analysis to get an ida about the augmentation technique needed for this problem

In [21]:
```
def is_grey_scale(givenImage):
    w,h = givenImage.size
    for i in range(w):
        for j in range(h):
            r,g,b = givenImage.getpixel((i,j))
            if r != g != b: return False
    return True
```

Check color scale of Train images

In [22]:
```
sampleFrac = 0.4
#get our sampled images
isGreyList = []
for imageName in main_df['path'].sample(frac=sampleFrac):
    val = Image.open(imageName).convert('RGB')
    isGreyList.append(is_grey_scale(val))
print(np.sum(isGreyList) / len(isGreyList))
del isGreyList
```

```
0.026389444222311077
```

Check color scale of Test images

Get mean intensity for each channel RGB

In [24]:
```
def get_rgb_men(row):
    img = cv2.imread(row['path'])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    return np.sum(img[:,:,0]), np.sum(img[:,:,1]), np.sum(img[:,:,2])

tqdm.pandas()
main_df['R'], main_df['G'], main_df['B'] = zip(*main_df.progress_apply(lambda row: get_rgb_men(row), axis=1))
```

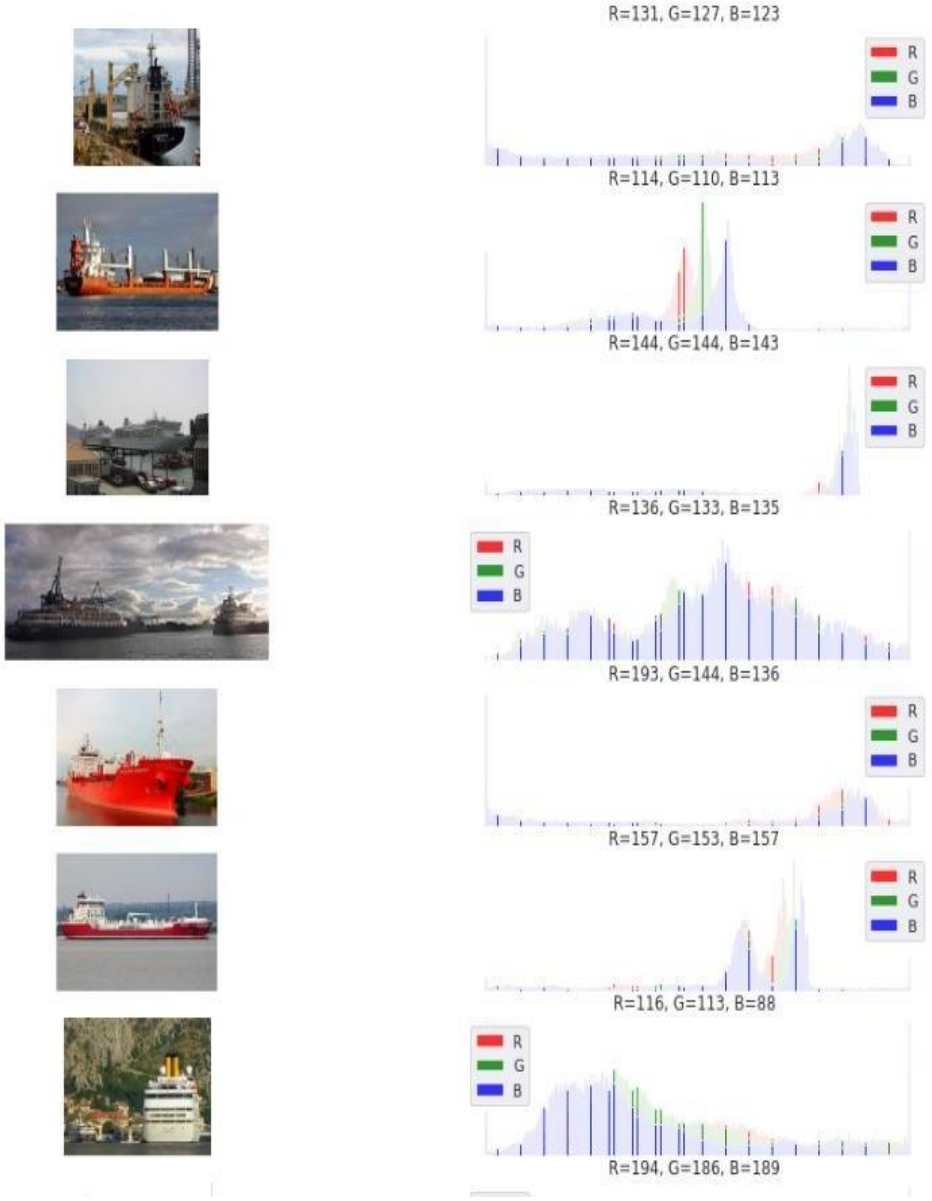
```
100%|██████████| 6252/6252 [00:10<00:00, 592.50it/s]
```

In [25]:
```
def show_color_dist(df, count):
    fig, axr = plt.subplots(count,2,figsize=(15,15))
    if df.empty:
        print("Image internsity of selected color is weak")
        return
    for idx, i in enumerate(np.random.choice(df['path'], count)):
        img = cv2.imread(i)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        axr[idx,0].imshow(img)
        axr[idx,0].axis('off')
        axr[idx,1].set_title('R={:.0f}, G={:.0f}, B={:.0f} '.format(np.mean(img[:,:,0]), np.mean(img[:,:,1]), np.mean(img[:,:,2]
        x, y = np.histogram(img[:,:,0], bins=255)
        axr[idx,1].bar(y[:-1], x, label='R', alpha=0.8, color='red')
        x, y = np.histogram(img[:,:,1], bins=255)
        axr[idx,1].bar(y[:-1], x, label='G', alpha=0.8, color='green')
        x, y = np.histogram(img[:,:,2], bins=255)
        axr[idx,1].bar(y[:-1], x, label='B', alpha=0.8, color='blue')
        axr[idx,1].legend()
        axr[idx,1].axis('off')
```

Red images and their color distribution

Since we are picking random images, some image may appear multiple times

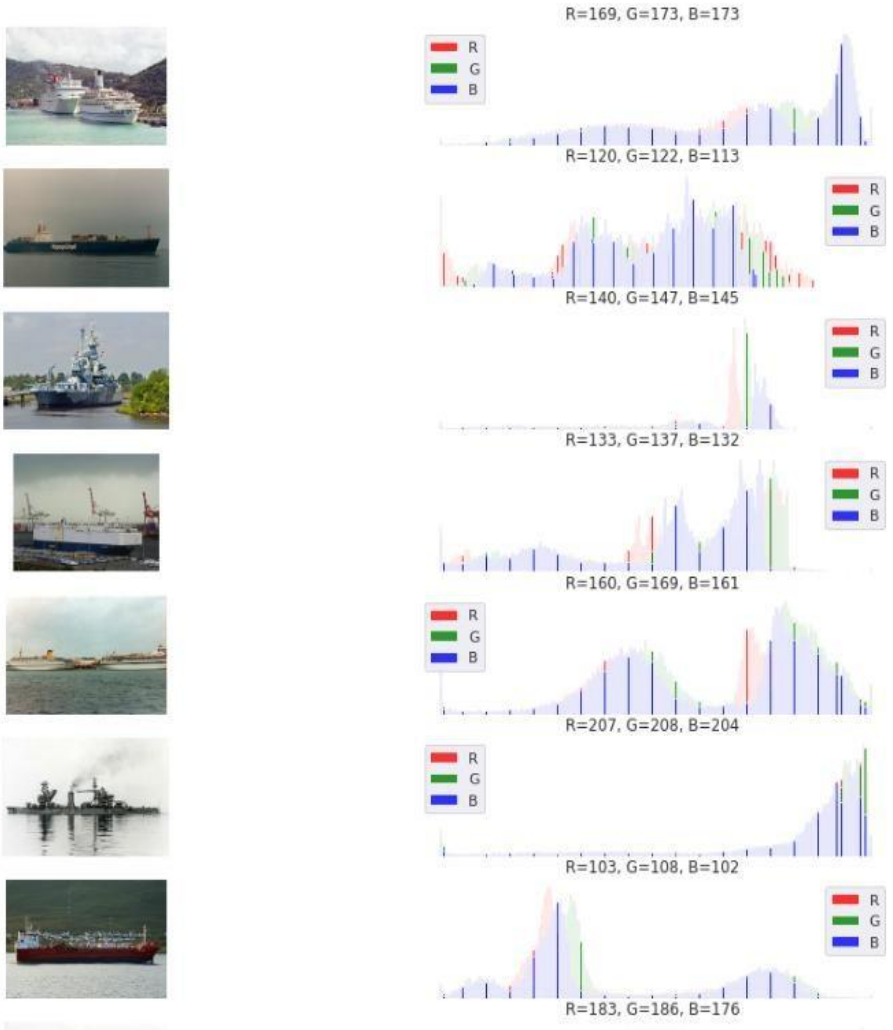
```
In [26]: df = main_df[((main_df['B']) < main_df['R']) & ((main_df['G']) < main_df['R'])]
         show_color_dist(df, 8)
```



Green images and their color distribution

Since we are picking random images, some image may appear multiple times

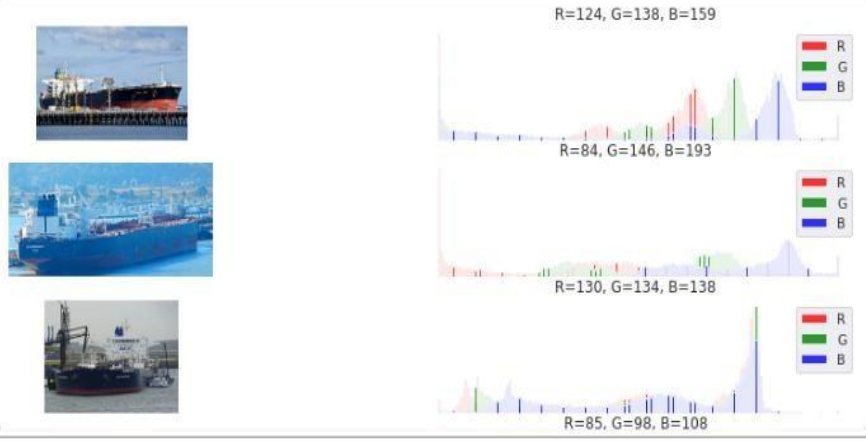
```
In [27]: df = main_df[(main_df['G'] > main_df['R']) & (main_df['G'] > main_df['B'])]
         show_color_dist(df, 8)
```



Blue images and their color distribution

Since we are picking random images, some image may appear multiple times

```
In [28]: df = main_df[(main_df['B'] > main_df['R']) & (main_df['B'] > main_df['G'])]
         show_color_dist(df, 8)
```



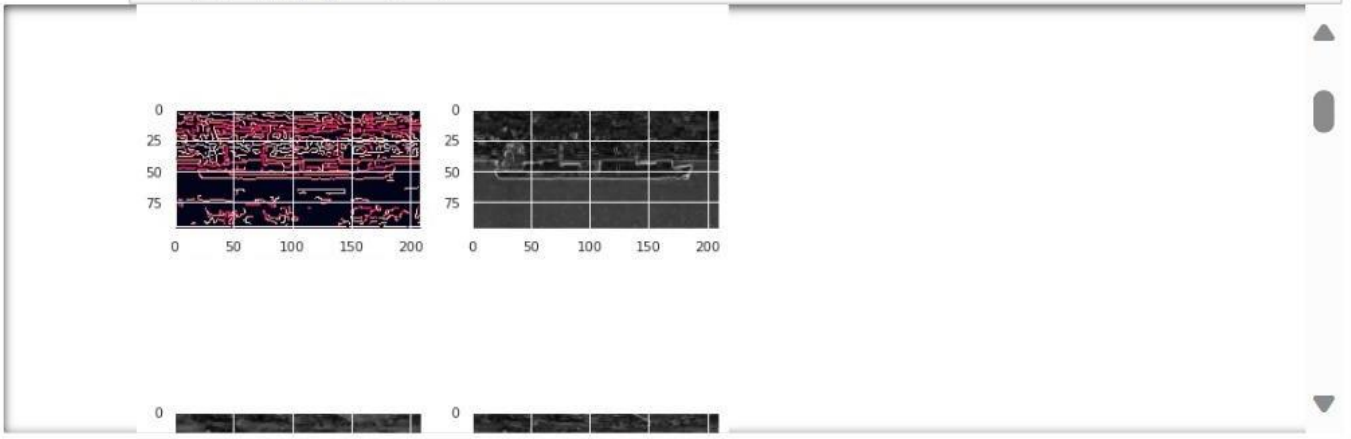
Feature

Analyzing Edges

A Sobel filter is one means of getting a basic edge magnitude/gradient image. Can be useful to threshold and find prominent linear features, etc. Several other similar filters in skimage.filters are also good edge detectors: roberts, scharr, etc. and you can control direction, i.e. use an anisotropic version.

In [29]:
```
def edges_images_gray(class_name):
    classes_df = main_df[main_df['classes'] ==  class_name].reset_index(drop = True)
    for idx,i in enumerate(np.random.choice(classes_df['path'],4)):
        image = cv2.imread(i)
        gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        edges = sobel(image)
        gray_edges=canny(gray)
        dimension = edges.shape
        fig = plt.figure(figsize=(8, 8))
        plt.suptitle(class_name)
        plt.subplot(2,2,1)
        plt.imshow(gray_edges)
        plt.subplot(2,2,2)
        plt.imshow(edges[:dimension[0],:dimension[1],0], cmap="gray")
        plt.subplot(2,2,3)
        plt.imshow(edges[:dimension[0],:dimension[1],1], cmap='gray')
        plt.subplot(2,2,4)
        plt.imshow(edges[:dimension[0],:dimension[1],2], cmap='gray')
        plt.show()
```

In [30]:
```
for class_name in main_df['classes'].unique():
    edges_images_gray(class_name)
```

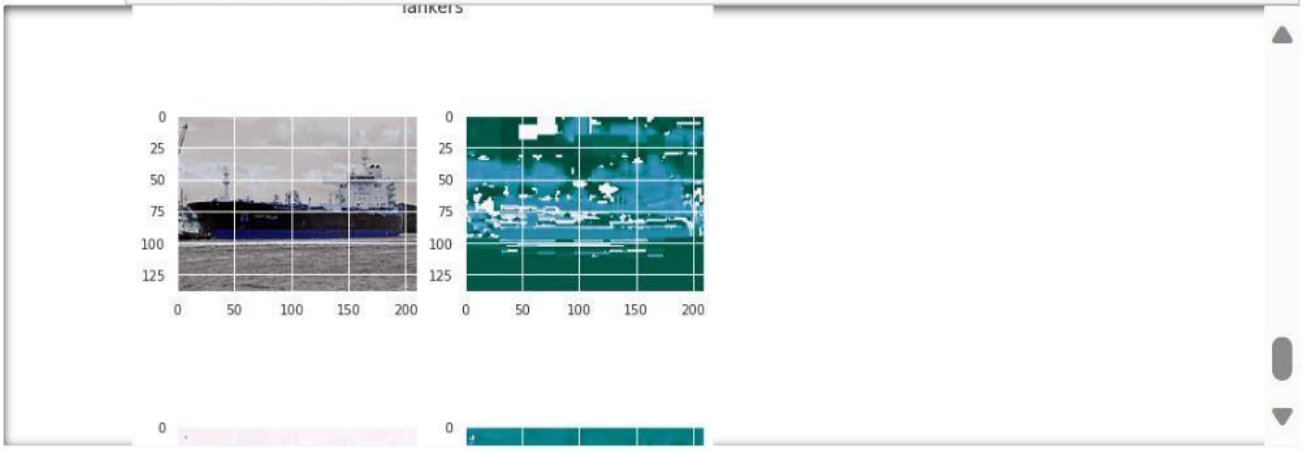


HSV Transform

Since this contest is about time series ordering, I think it's possible there may be useful information in a transform to HSV color space. HSV is useful for identifying shadows and illumination, as well as giving us a means to identify similar objects that are distinct by color between scenes (hue), though there's no guarantee the hue will be stable.

In [31]:
```
def hsv_images(class_name):
    classes_df = main_df[main_df['classes'] ==  class_name].reset_index(drop = True)
    for idx,i in enumerate(np.random.choice(classes_df['path'],4)):
        image = cv2.imread(i)
        hsv = color.rgb2hsv(image)
        dimension = hsv.shape
        fig = plt.figure(figsize=(8, 8))
        plt.suptitle(class_name)
        plt.subplot(2,2,1)
        plt.imshow(image)
        plt.subplot(2,2,2)
        plt.imshow(hsv[:dimension[0],:dimension[1],0], cmap="PuBuGn")
        plt.subplot(2,2,3)
        plt.imshow(hsv[:dimension[0],:dimension[1],1], cmap='PuBuGn')
        plt.subplot(2,2,4)
        plt.imshow(hsv[:dimension[0],:dimension[1],2], cmap='PuBuGn')
        plt.show()
```

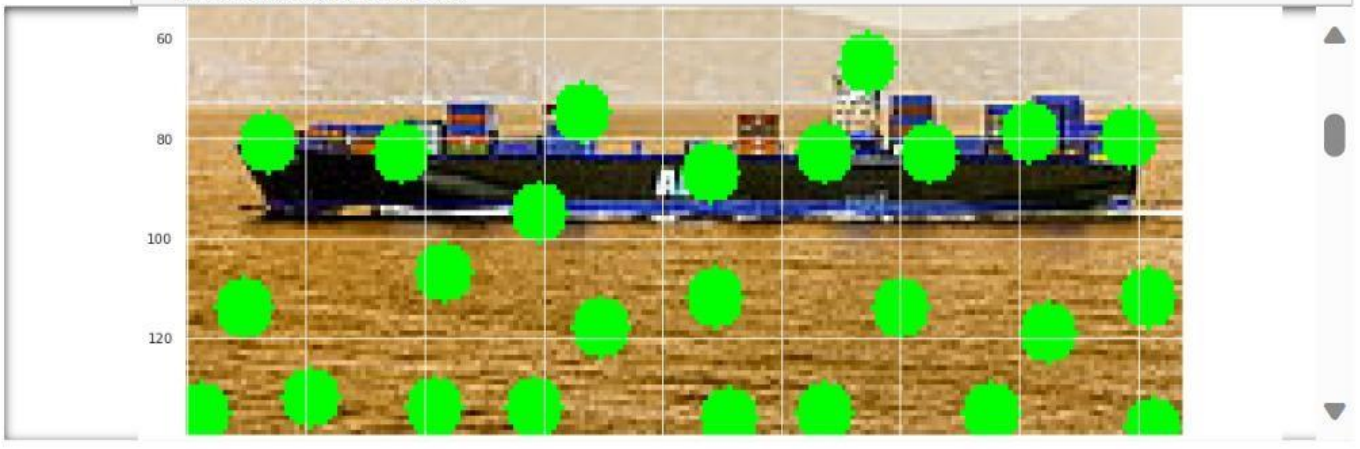
In [32]:
```
for class_name in main_df['classes'].unique():
    hsv_images(class_name)
```



Corners

In [33]:
```
def corners_images_gray(class_name):
    classes_df = main_df[main_df['classes'] ==  class_name].reset_index(drop = True)
    for idx,i in enumerate(np.random.choice(classes_df['path'],4)):
        image = cv2.imread(i)
        gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        corners_gray = cv2.goodFeaturesToTrack(gray, maxCorners=50, qualityLevel=0.02, minDistance=20)
        corners_gray = np.float32(corners_gray)
        for item in corners_gray:
            x, y = item[0]
            cv2.circle(image, (int(x), int(y)), 6, (0, 255, 0), -1)
        fig = plt.figure(figsize=(32, 32))
        plt.suptitle(class_name)
        plt.subplot(2,2,1)
        plt.imshow(image, cmap="BuGn")
        plt.show()
```

In [34]:
```
for class_name in main_df['classes'].unique():
    corners_images_gray(class_name)
```



Sift Features

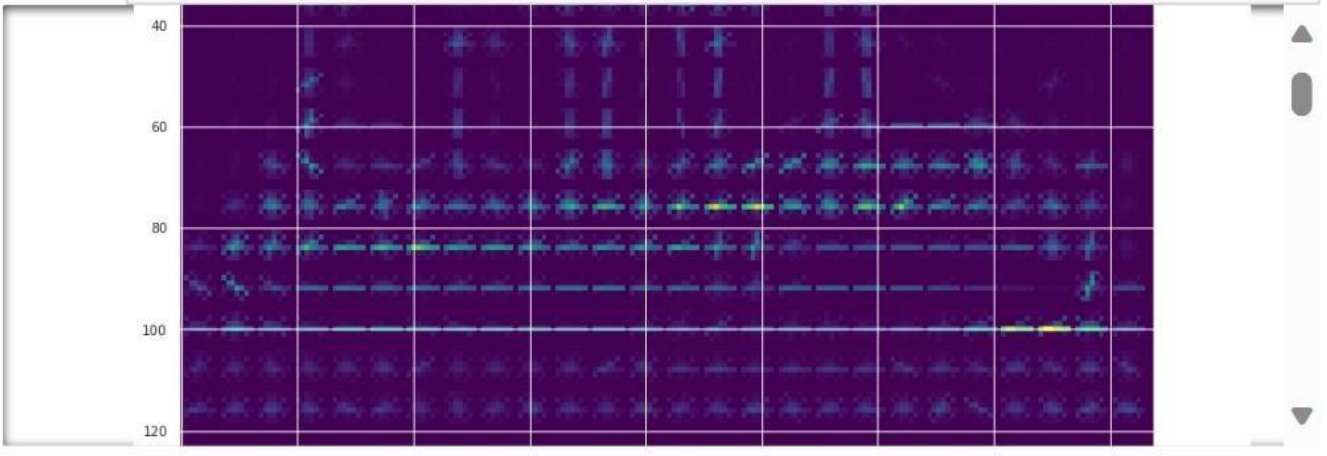
In [35]:
```
def sift_images_gray(class_name):
    classes_df = main_df[main_df['classes'] ==  class_name].reset_index(drop = True)
    for idx,i in enumerate(np.random.choice(classes_df['path'],4)):
        image = cv2.imread(i)
        gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        sift = cv2.SIFT_create()
        kp, des = sift.detectAndCompute(gray, None)
        kp_img = cv2.drawKeypoints(image, kp, None, color=(0, 255, 0), flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
        fig = plt.figure(figsize=(32, 32))
        plt.suptitle(class_name)
        plt.subplot(2,2,1)
        plt.imshow(kp_img, cmap="viridis")
        plt.show()
```

In [36]:
```
for class_name in main_df['classes'].unique():
    sift_images_gray(class_name)
```

HOG

In [37]:
```
def hog_images(class_name):
    classes_df = main_df[main_df['classes'] ==  class_name].reset_index(drop = True)
    for idx,i in enumerate(np.random.choice(classes_df['path'],4)):
        image = cv2.imread(i)
        fd, hog_image = hog(image, orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True, multichannel=
        fig = plt.figure(figsize=(32, 32))
        plt.suptitle(class_name)
        plt.subplot(2,2,1)
        plt.imshow(hog_image, cmap="viridis")
        plt.show()
```

In [38]:
```
for class_name in main_df['classes'].unique():
    hog_images(class_name)
```



Augmentations

In [39]:
```
def plot_augimages(paths, datagen):
    plt.figure(figsize = (14,28))
    plt.suptitle('Augmented Images')

    midx = 0
    for path in paths:
        data = Image.open(path)
        data = data.resize((192,192))
        samples = expand_dims(data, 0)
        it = datagen.flow(samples, batch_size=1)

        # Show Original Image
        plt.subplot(10,5, midx+1)
        plt.imshow(data)
        plt.axis('off')

        # Show Augmented Images
        for idx, i in enumerate(range(4)):
            midx += 1
            plt.subplot(10,5, midx+1)

            batch = it.next()
            image = batch[0].astype('uint8')
            plt.imshow(image)
            plt.axis('off')
        midx += 1

    plt.tight_layout()
    plt.show()
```

Split X and Y

In [40]:
```
X, y = main_df[['path', 'classes']], main_df['classes']
```

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Data Generators

VGG Generators

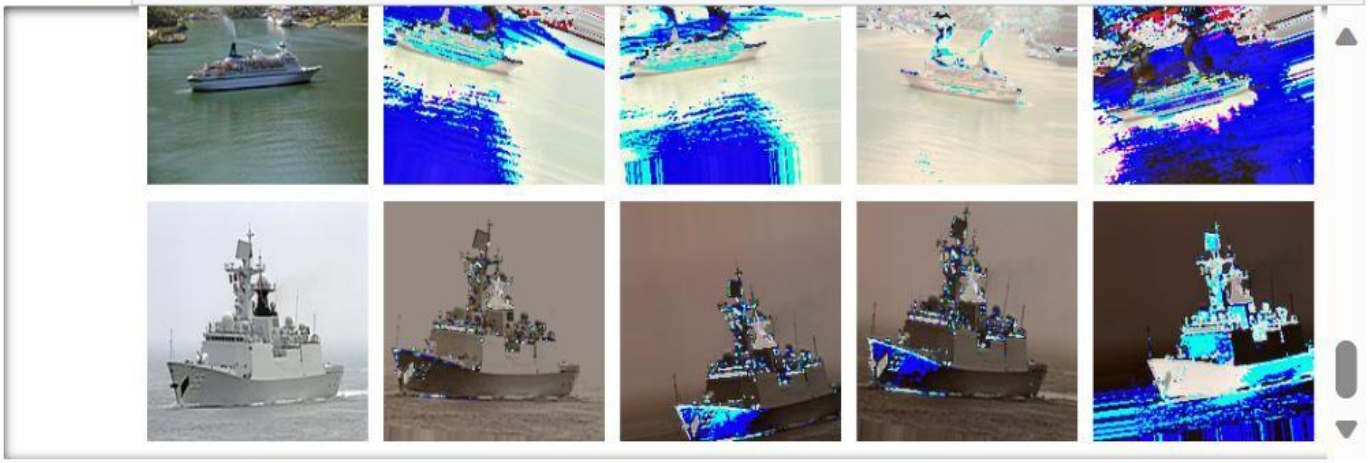
In [42]:
```
from keras.applications.vgg16 import preprocess_input
```

In [43]:
```
vgg_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.10,
    brightness_range=[0.6,1.4],
    channel_shift_range=0.7,
    width_shift_range=0.15,
    height_shift_range=0.15,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode='nearest',
    preprocessing_function=preprocess_input
)

train_generator_vgg = vgg_datagen.flow_from_dataframe(
        X_train,  # This is the source directory for training images
        x_col='path',
        y_col='classes',
        target_size=(192, 192),  # All images will be resized to 150x150
        batch_size=32,
        class_mode="categorical",
        shuffle=True,
)
val_generator_vgg = vgg_datagen.flow_from_dataframe(
        X_test,  # This is the source directory for training images
        x_col='path',
        y_col='classes',
        target_size=(192, 192),  # All images will be resized to 150x150
        batch_size=32,
        class_mode="categorical",
        shuffle=True,
)
```

```
Found 5001 validated image filenames belonging to 5 classes.
Found 1251 validated image filenames belonging to 5 classes.
```

In [44]:
```
plot_augimages(np.random.choice(main_df['path'],10), vgg_datagen)
```



Resnet50 Generators

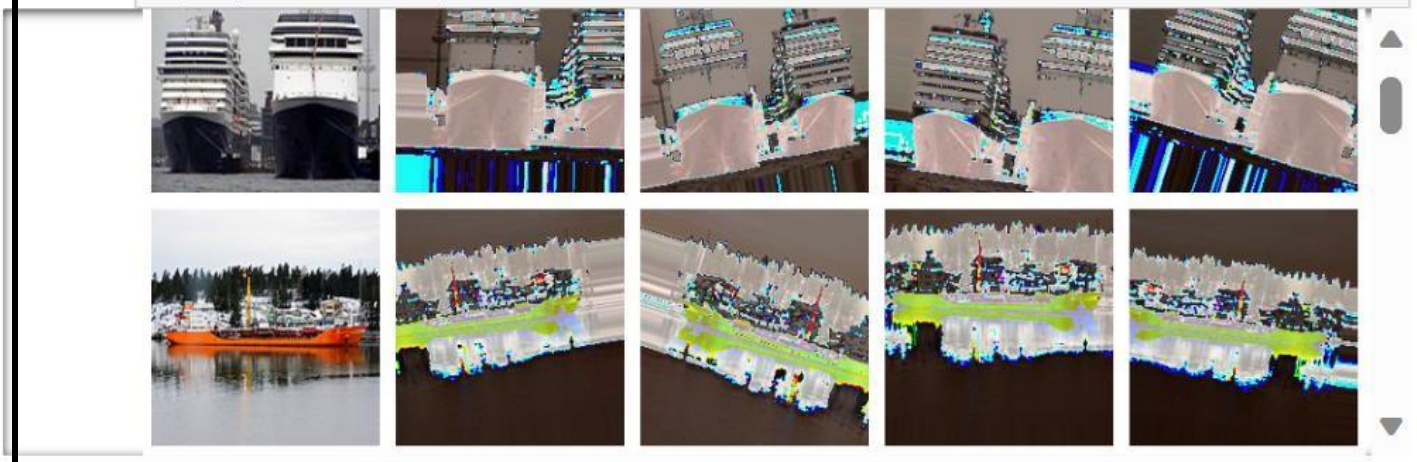
In [45]:
```
from tensorflow.keras.applications.resnet50 import preprocess_input
```

In [46]:
```
resnet50_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.10,
    brightness_range=[0.6,1.4],
    channel_shift_range=0.7,
    width_shift_range=0.15,
    height_shift_range=0.15,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode='nearest',
    preprocessing_function=preprocess_input
)

train_generator_resnet50 = resnet50_datagen.flow_from_dataframe(
        X_train,  # This is the source directory for training images
        x_col='path',
        y_col='classes',
        target_size=(192, 192),  # All images will be resized to 150x150
        batch_size=32,
        class_mode="categorical",
        shuffle=True,
)
val_generator_resnet50 = resnet50_datagen.flow_from_dataframe(
        X_test,  # This is the source directory for training images
        x_col='path',
        y_col='classes',
        target_size=(192, 192),  # All images will be resized to 150x150
        batch_size=32,
        class_mode="categorical",
        shuffle=True,
)
```

```
Found 5001 validated image filenames belonging to 5 classes.
Found 1251 validated image filenames belonging to 5 classes.
```

In [47]:
```
try:
    plot_augimages(np.random.choice(main_df['path'],10), resnet50_datagen)
except:
    print("continue no issues")
```



Modelling

In [52]:
```
resnet50 = ResNet50(include_top = False, input_shape = (192,192,3), weights = 'imagenet')

# training of all the convolution is set to false
for layer in resnet50.layers:
    layer.trainable = False

x = GlobalAveragePooling2D()(resnet50.output)
predictions = Dense(5, activation='softmax')(x)

model_resnet50 = Model(inputs = resnet50.input, outputs = predictions)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94773248/94765736 [==============================] - 1s 0us/step
94781440/94765736 [==============================] - 1s 0us/step
```

Resnet Compilation

In [55]:
```
model_resnet50.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
```

Fitting Resnet50

In [ ]:
```
history_resnet50 = model_resnet50.fit(
      train_generator_resnet50,
      validation_data=val_generator_resnet50,
      epochs=50,
      verbose=2)
```

Analyzing perfomance of Resnet50

In [ ]:
```
plt.figure(figsize=(15,5))
plt.plot(history_resnet50.history['accuracy'])
plt.plot(history_resnet50.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

In [ ]:
```
plt.figure(figsize=(15,5))
plt.plot(history_resnet50.history['loss'])
plt.plot(history_resnet50.history['val_loss'])
plt.title('Model loss')
plt.ylabel('loss')
plt.xlabel('Epoch')
plt.show()
```

8. ADVANTAGES & DISADVANTAGES

Advantages:

Safety Assurance: Classification societies play a crucial role in ensuring the safety of ships by establishing and enforcing standards for design, construction, and maintenance. This helps prevent accidents and contributes to overall maritime safety.

Risk Management: Ship classification provides a systematic approach to risk management. By adhering to classification standards, shipowners and operators can reduce the risk of accidents, breakdowns, and other operational issues.

Insurance and Financing: Many insurance companies and financial institutions rely on the classification of a ship when determining insurance rates and financing terms. A well-classified ship may be seen as a lower risk, leading to more favorable terms.

Compliance with Regulations: Classification societies help shipowners comply with international and national regulations governing the maritime industry. This includes environmental regulations, safety standards, and other legal requirements.

Global Recognition: Ships that meet classification standards are recognized globally, making it easier for them to operate in international waters and ports without facing regulatory barriers.

Disadvantages:

Costs: Obtaining and maintaining classification can be expensive for shipowners. They need to invest in design, construction, and maintenance practices that adhere to the classification society's standards, which can increase overall costs.

Subjectivity: The classification process involves a certain degree of subjectivity, as surveyors from different classification societies may interpret standards differently. This subjectivity could lead to variations in the assessment of a ship's compliance.

Inflexibility: Classification standards may not always keep pace with rapidly evolving technologies and industry practices. This can lead to situations where innovative ship designs or technologies struggle to conform to existing classification norms.

Dependency on Classification Societies: Shipowners may become heavily dependent on classification societies for certification and compliance, creating a potential conflict of interest. This dependency may limit competition and innovation in the classification industry.

Bureaucracy: The classification process involves various administrative procedures, paperwork, and inspections. This bureaucratic aspect can sometimes lead to delays and inefficiencies in the shipbuilding and operational processes.

9. CONCLUSION

In conclusion, ship classification plays a pivotal role in ensuring the safety, reliability, and compliance of maritime vessels with international and national standards. The advantages of ship classification are evident in the enhanced safety assurance, effective risk management, and the facilitation of global recognition. Shipowners benefit from improved access to insurance and financing, while compliance with regulations is streamlined through the guidance of classification societies.However, the process of ship classification is not without its challenges. The associated costs can be a significant burden for shipowners, and the subjectivity in the assessment process may introduce variations in compliance interpretations. Additionally, the potential inflexibility of classification standards to adapt to rapidly evolving technologies poses a challenge for innovative designs.

10. FUTURE SCOPE

The future scope of ship classification is likely to be shaped by technological advancements, evolving industry needs, and an increased focus on sustainability. Here are some potential aspects that could influence the future of ship classification:

Incorporation of Emerging Technologies:

Integration of advanced technologies such as artificial intelligence, machine learning, and the Internet of Things (IoT) in ship design, construction, and maintenance processes.

Emphasis on digitalization and smart technologies to enhance real-time monitoring, predictive maintenance, and overall operational efficiency.

Focus on Environmental Sustainability:

Integration of green technologies to comply with stricter environmental regulations and reduce the environmental impact of shipping.

Classification standards may evolve to include criteria related to energy efficiency, emissions reduction, and the use of alternative fuels.

Cybersecurity Measures:

Growing emphasis on cybersecurity measures to protect ships from potential cyber threats and ensure the secure operation of onboard systems.

Development of standards and guidelines for cybersecurity within the context of ship classification.

Adaptation to Autonomous Shipping:

Classification societies may need to develop standards and guidelines for the safe integration of autonomous and unmanned vessels into the maritime industry.

Evaluation and certification processes for autonomous technologies to ensure compliance with safety and operational standards.

11. APPENDIX

The drive link for the ipynb file with the Ship classification model using Resnet_50

https://drive.google.com/file/d/1x9_e6hocWvNdI8oUxT1GEi55ba7Agr4b/view?usp=sharing

The github link for the ipynb file

https://github.com/Mukhul17/web105/blob/main/ship-classifier.ipynb