# Final Project Report

Neural Network Ahoy Cutting-edge ship classification for Maritime Mastery

## 1. Introduction

### 1.1 Project overviews

"Neural Networks Ahoy: Cutting-Edge Ship Classification for Maritime Mastery" is a deep learning project aimed at developing an advanced ship classification system. The primary objective is to accurately identify various types of ships from images, catering to the needs of maritime traffic monitoring and coastal defense early warnings.

### 1.2. Objectives

The objective of this project is to develop a ship classification system using computer vision techniques, specifically employing the VGG16 model. The aim is to accurately identify five categories of ships—Cargo, Carrier, Military, Cruise, and Tankers—from images. By leveraging pre-trained weights of VGG16 and customizing its top layer, the system seeks to achieve high classification accuracy.

## 2: Project Initialization and Planning Phase

The "Project Initialization and Planning Phase" marks the project's outset, defining goals, scope, and stakeholders. This crucial phase establishes project parameters, identifies key team members, allocates resources, and outlines a realistic timeline. It also involves risk assessment and mitigation planning. Successful initiation sets the foundation for a well-organized and efficiently executed machine learning project, ensuring clarity, alignment, and proactive measures for potential challenges.

### 2.1: Define Problem Statement

Define Problem Statements (Customer Problem Statement Template) The maritime sector is and includes a wide variety of ship types, each intended for a particular function, such as cargo transportation or naval defense. For maritime operations, such as navigation, port management, and security, ships must be classified effectively.

**Reference:** Click Here

### 2.2: Project Proposal (Proposed Solution)

The purpose of ship classification is to identify various types of ships as accurately as possible, which is of great significance for monitoring the rights and interests of maritime traffic and improving coastal defense early warnings. The images in the data belong to 10 categories of ships - Aircraft Carrier, Bulkers, Car Carrier, Container Ship, Cruise, DDG, Recreational, Sailboat, Submarine, Tug.

**Reference:** Click Here

### 2.3: Initial Project Planning

The project planning template provided outlines a project titled "Neural Networks Ahoy: Cutting-edge Ship Classification for Maritime Mastery." The project involves various sprints with defined tasks, team members, priorities, and timelines. It includes phases like project initialization, data collection, preprocessing, model development, model optimization, and project executable files creation. Each sprint has specific user stories, points, and team members assigned to different tasks. The project progresses through phases such as defining problem statements, project proposals, data collection, model training, optimization, and ends with creating executable files, documentation, and demonstration. The template emphasizes a structured approach to project management, ensuring clear

objectives, team collaboration, and a timeline for completion.
**Reference:** Click Here

# 3: Data Collection and Preprocessing Phase

The images will be preprocessed by resizing, normalizing, augmenting, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

## 3.1: Data Collection Plan, Raw Data Sources Identified

Identify existing datasets containing images of ships categorized into Aircraft Carrier, Bulkers, Car Carrier, Container Ship, Cruise, DDG, Recreational, Sailboat, Submarine, Tug. Explore open data repositories, academic sources, and industry partnerships to access relevant datasets. Evaluate the quality, quantity, and diversity of available datasets to ensure they align with project requirements.

**Reference:** Click Here

## 3.2: Data Quality Report

- **Data Quality Issues**:

**Imbalanced Classes**: Some classes in the Kaggle Dataset have fewer images than others, which is a moderate severity issue. The resolution plan is to add more images from the internet to stabilize the count.
**Low-Quality Images**: A few images in the Kaggle Dataset are of low quality, considered a low severity issue. Since there are only a few such images, they can be ignored. This template will help in identifying and addressing data discrepancies systematically for your data analytics project.

**Reference:** Click Here

## 3.3: Data Preprocessing

Preprocessing Steps:

- Resizing: Images are resized to a target size of ((224,224)).
- Data Augmentation: Includes zoom range, shear range, width shift, height shift, and horizontal flip.
- Normalization: Normalize pixel using preprocessing_input function to convert image from RGB to BGR, as the standard format of ImageNet
- Batch Normalization & Whitening: These techniques are applied to enhance data quality and promote model generalization.

**Reference:** Click Here

# 4: Mode Development Phase

This phase focuses on selecting a suitable model and preparing it for the specific task of ship classification. The VGG16 model is identified as a solid baseline with the potential for fine-tuning to meet the project's requirements.

## 4.1: Model Selection Report

VGG16 is a solid choice as a starting point for your ship classification task. Its pre-trained weights and adaptable architecture make it a good baseline for achieving good classification accuracy.

However, keep in mind potential drawbacks like computational cost and the existence of potentially more efficient alternatives.

**Reference:** Click Here

### 4.2: Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots

**Reference:** Click Here

## 5: Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining the VGG16 model through hyperparameter tuning, which enhances the model's predictive accuracy and efficiency1. The VGG16 model is selected due to its simplicity and high performance in image classification, enabling it to learn complex features effectively.

### 5.1: Tuning Documentation

The Hyperparameter Tuning Documentation for the "Neural Networks Ahoy: Cutting-edge Ship Classification for Maritime Mastery" project involves tuning the VGG16 model with specific hyperparameters to optimize its performance. The tuned hyperparameters include the input size of the image, the optimizer function with SGD and learning rate, the number of classes to be classified, and the variable to save the model1. Additionally, parameters for the callback function, the number of epochs to run, the dataset to use, and other settings are adjusted to enhance the model's efficiency and accuracy during training and evaluation1. This meticulous tuning process aims to fine-tune the model for peak performance, ensuring that it can effectively classify ship images with high accuracy and efficiency.

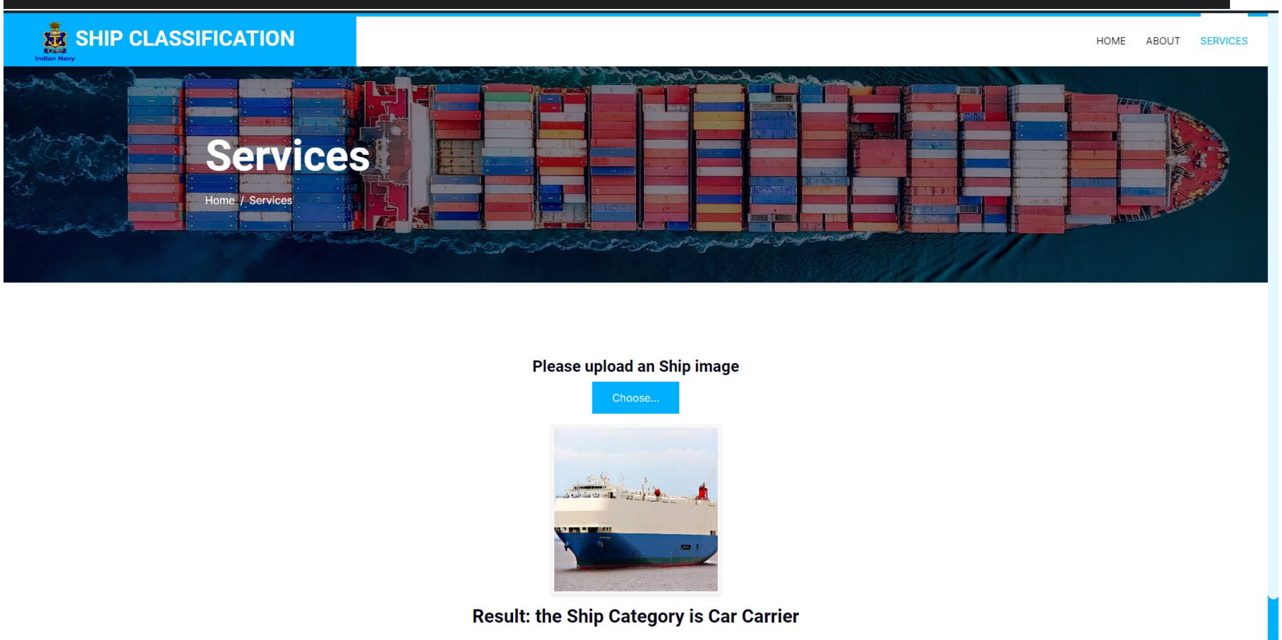
### 5.2: Final Model Selection Justification

VGG16 is selected due to its efficiency in image classification; this is possible because of its simplicity and high performance. It has achieved impressive results in competitions such as ImageNet Large Scale Visual Recognition Challenge with a total of 16 weight layers consisting of 13 convolution layers and 3 fully connected ones. This model, though the lightest among architectures, learns complex features so well that it becomes an ideal option for creating models that can be trusted to work even under difficult conditions.

**Reference:** Click Here

## 6. Results
## 6.1. Output Screenshots

```
Click here to ask Blackbox to help you code faster
img=image.load_img(r'C:\Users\Asus\Documents\Ship Classification\flask\input\Disney-Treasure-Exterior-2-1-1.jpg',target_size=(224,224))
img=image.img_to_array(img)
img=img.reshape((1,img.shape[0],img.shape[1],img.shape[2]))
img=preprocess_input(img)
pred=vgg_model.predict(img)
pred=pred.flatten()
pred=list(pred)
n=max(pred)
val_dict={0: 'Aircraft Carrier',
 1: 'Bulkers',
 2: 'Car Carrier',
 3: 'Container Ship',
 4: 'Cruise',
 5: 'DDG',
 6: 'Recreational',
 7: 'Sailboat',
 8: 'Submarine',
 9: 'Tug'}
result=val_dict[pred.index(n)]
print(result)
[53]
1/1 [==============================] - 0s 104ms/step
Cruise
```




## 7. Advantages & Disadvantages
**Advantages:**

1.Enhanced Efficiency:
By streamlining cargo logistics, we reduce transit times and minimize delays.
Automated processes lead to faster document handling and fewer errors.
2.Cost Savings:
Optimized routes and efficient cargo handling result in reduced fuel consumption and operational costs.
Automation reduces manual labor, leading to long-term savings.
3.Improved Customer Satisfaction:
Real-time cargo tracking enhances transparency and allows clients to monitor their shipments.
Faster delivery times positively impact customer experience.
4.Competitive Edge:
The XYZ Project positions our company as an industry leader in innovative supply chain solutions.
Improved efficiency attracts new clients and retains existing ones.

**Disadvantages:**
1.Implementation Challenges:
Integrating a digital platform across various stakeholders (shipping companies, port authorities, etc.) can be complex.
Resistance to change from existing manual processes may hinder adoption.
2.Initial Investment:
The project requires substantial upfront investment in software development, hardware upgrades, and training.
Balancing the budget while meeting project goals is crucial.
3.Regulatory Compliance:
Navigating international regulations and compliance standards can be time-consuming.
Legal complexities may arise, especially when dealing with multiple jurisdictions.
4.Dependency on Technology:
Relying heavily on digital platforms introduces risks related to cybersecurity, system failures, and data breaches.
Backup plans and robust security measures are essential.

## 8. Conclusion

"Neural Networks Ahoy: Cutting-Edge Ship Classification for Maritime Mastery" presents a robust solution leveraging Convolutional Neural Networks (CNNs), specifically the VGG16 model, to accurately classify images of ships into ten distinct categories. Through meticulous data preparation, model selection, customization, and training, the project achieves high classification accuracy, making it a valuable tool for maritime traffic monitoring, coastal defence early warnings, and other maritime applications.

The utilization of transfer learning with VGG16 allows for efficient adaptation of pre-trained models to the ship classification task, while the deployment via the Flask framework ensures seamless integration into existing systems or standalone applications. The project's impact is far-reaching, providing tangible benefits in enhancing maritime security, optimizing traffic management, and facilitating various maritime-related endeavours.

With the necessary hardware resources and software requirements outlined, the project stands ready for implementation, offering scalability and real-time prediction capabilities. Ultimately, "Neural Networks Ahoy" represents a significant advancement in leveraging cutting-edge deep learning techniques for maritime classification, promising a safer and more efficient maritime domain.

## 9. Future Scope

The scope of the project involves preprocessing a dataset containing images of ships and categorizing them into the five specified classes. This classification system will utilize the Flask framework for deployment, providing a user-friendly interface for ship classification. The system will be designed to handle various ship images under different lighting, weather conditions, and perspectives, ensuring robustness and versatility.

# 10. Appendix

## 10.1. Source Code

```
💡 Click here to ask Blackbox to help you code faster
import pandas as pd
import numpy as np
import os
import torch
```

```
💡 Click here to ask Blackbox to help you code faster
Base_path_to_datas_to_train = r"C:\Users\Asus\Documents\Ship Classification\flask\input\train"
Base_path_to_datas_to_val = r"C:\Users\Asus\Documents\Ship Classification\flask\input\valid"
Base_path_to_datas_to_test = r"C:\Users\Asus\Documents\Ship Classification\flask\input\test"


PIN_MEMORY=False
#=========================================================================
clas_of_img={}
#=========================================================================
results_dict = {}
modeli=[]
#=========================================================================
torch.manual_seed(5017)
torch.cuda.manual_seed(5017)
```

```
💡 Click here to ask Blackbox to help you code faster
def gen_classes(path,class_dict):
    for indx, path in enumerate(os.listdir(Base_path_to_datas_to_train)):
        class_dict[indx] = path


    return len(class_dict)
```

```
💡 Click here to ask Blackbox to help you code faster
Num_classes = gen_classes(Base_path_to_datas_to_train,clas_of_img)
clas_of_img
```

```
💡 Click here to ask Blackbox to help you code faster
from tensorflow.keras.preprocessing.image import ImageDataGenerator# type: ignore
from keras.applications.vgg16 import preprocess_input# type: ignore
from tensorflow.keras.applications import VGG16# type: ignore

train_datagen = ImageDataGenerator(
    rotation_range=45,
    horizontal_flip=True,
    width_shift_range=0.5,
    height_shift_range=0.5,
    validation_split=0.2,
    preprocessing_function=preprocess_input
)

test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
```

```
Click here to ask Blackbox to help you code faster
train_set = train_datagen.flow_from_directory(Base_path_to_datas_to_train, batch_size=32, target_size=(224,224))
```

```
Click here to ask Blackbox to help you code faster
validation_set = train_datagen.flow_from_directory(Base_path_to_datas_to_val, batch_size=32,target_size=(224,224),shuffle=False)
```

```
Click here to ask Blackbox to help you code faster
test_set = test_datagen.flow_from_directory(Base_path_to_datas_to_test, batch_size=32,target_size=(224,224))
```

```
Click here to ask Blackbox to help you code faster
from keras.layers import Dense,Flatten,Dropout
from keras.models import Model
```

```
Click here to ask Blackbox to help you code faster
def create_model(input_shape,n_classes,optimizer='rmsprop'):
    conv_base=VGG16(include_top=False,weights='imagenet',input_shape=input_shape)
    for layer in conv_base.layers:
        layer.trainable = False
    top_model=conv_base.output
    top_model=Flatten(name="flatten")(top_model)
    top_model=Dense(700,activation='relu')(top_model)
    top_model=Dense(1272,activation='relu')(top_model)
    top_model=Dropout(0,2)(top_model)
    output_layer=Dense(n_classes, activation='softmax')(top_model)
    model=Model(inputs=conv_base.input,outputs=output_layer)
    model.compile(optimizer=optimizer,loss='categorical_crossentropy',metrics=['accuracy'])
    return model
```

```
Click here to ask Blackbox to help you code faster
from tensorflow.keras.optimizers import SGD# type: ignore
```

```
Click here to ask Blackbox to help you code faster
input_shape=(224,224,3)
optim=SGD(learning_rate=0.001)
n_classes=10
vgg_model=create_model(input_shape,n_classes,optim)
```

```
Click here to ask Blackbox to help you code faster
vgg_model.summary()
```

```
Click here to ask Blackbox to help you code faster
from keras.callbacks import ModelCheckpoint
cp=ModelCheckpoint('best1.h5',monitor='val_loss',verbose=1,save_best_only=True)
```

```
💡 Click here to ask Blackbox to help you code faster
epoch=18
history=vgg_model.fit_generator(generator=train_set,
                                steps_per_epoch=train_set.n//train_set.batch_size,
                                validation_steps=validation_set.n//validation_set.batch_size,
                                validation_data=validation_set,
                                callbacks=[cp],
                                epochs=epoch)
```

```
💡 Click here to ask Blackbox to help you code faster
print(validation_set.classes)
```

```
💡 Click here to ask Blackbox to help you code faster
results = vgg_model.evaluate(validation_set)
print(results)
```

```
💡 Click here to ask Blackbox to help you code faster
predictions = vgg_model.predict(test_set)  # Get predictions
predictions = np.argmax(predictions, axis=1)  # Get predicted classes

ground_truth = test_set.classes  # Get ground truth labels

# Calculate confusion matrix
from sklearn.metrics import classification_report
cm = classification_report(ground_truth, predictions)
print(cm)
```

```
💡 Click here to ask Blackbox to help you code faster
predictions = vgg_model.predict(validation_set)  # Get predictions
predictions = np.argmax(predictions, axis=1)  # Get predicted classes

ground_truth = validation_set.classes  # Get ground truth labels

# Calculate confusion matrix
from sklearn.metrics import classification_report
cm = classification_report(ground_truth, predictions)
print(cm)
```

```
💡 Click here to ask Blackbox to help you code faster
from tensorflow.keras.preprocessing import image# type: ignore
```

```
💡 Click here to ask Blackbox to help you code faster
img=image.load_img(r'C:\Users\Asus\Documents\Ship Classification\flask\input\Disney-Treasure-Exterior-2-1-1.jpg',target_size=(224,224))
img=image.img_to_array(img)
img=img.reshape((1,img.shape[0],img.shape[1],img.shape[2]))
img=preprocess_input(img)
pred=vgg_model.predict(img)
pred=pred.flatten()
pred=list(pred)
n=max(pred)
val_dict={0: 'Aircraft Carrier',
 1: 'Bulkers',
 2: 'Car Carrier',
 3: 'Container Ship',
 4: 'Cruise',
 5: 'DDG',
 6: 'Recreational',
 7: 'Sailboat',
 8: 'Submarine',
 9: 'Tug'}
result=val_dict[pred.index(n)]
print(result)
```

```
💡 Click here to ask Blackbox to help you code faster
vgg_model.save('vgg16-ship-classification.h5')
```

## 10.2. GitHub & Project Demo Link

In the upcoming module called Project Demonstration, individuals will be required to record a video by sharing their screens. They will need to explain their project and demonstrate its execution during the presentation.

For project file demonstration video, kindly click the link. Click Here

For project file submission in GitHub, kindly click the link and refer to the flow. Click Here