Ship Classification Project Report

INTRODUCTION

Project Overview

The maritime industry plays a pivotal role in global trade, with ships serving as the lifeblood of international commerce. Accurate and efficient ship classification is essential for various applications including maritime security, environmental monitoring and trade regulation.

Traditional ship classification methods have often relied on manual inspection, which can be time consuming and error-prone. With the advent of deep learning techniques, there is a tremendous opportunity to revolutionize ship classification by automating the process with high accuracy and speed.

Ship classification involves categorizing vessels into various classes such as military, cargo, cruise, tanker etc. bases on their size, type, function and other attributes. Deep learning models such as Convolution

Neural Networks (CNNs) have demonstrated their ability to extract intricate features from images, enabling the development of accurate and efficient ship classification systems.

The images in the data belong to 5 categories of ships - Cargo, Carrier, Military, Cruise and Tankers. The model used to train the model is VGG19, a computer vision model which is based on Convolutional Neural Networks (CNN). We will be using the pre-trained weights of the model and modify the top layer for performing our custom classification. The model is deployed using the Flask framework.

Purpose

Maritime Safety and Security: Governments and law enforcement agencies can use this technology to monitor and track ships in their territorial waters to ensure compliance with regulations and detect potentially suspicious or unauthorized vessels.

Port Operations and Logistics: Port authorities and logistics companies can use ship classification to optimize and streamline their operations. Knowing the type of ships arriving at ports can help in planning for efficient cargo handling and storage.

Environmental Monitoring: Identifying different types of ships can be crucial for monitoring and mitigating the environmental impact of maritime activities. For instance, different types of ships may have varying levels of emissions or environmental risks.

Insurance and Risk Assessment: Insurance companies and risk assessment agencies can benefit from ship classification to determine the level of risk associated with insuring different types of ships. This information can assist in setting appropriate premiums and managing underwriting processes.

Market Analysis and Trade Trends: Ship classification can provide insights into trade trends and market dynamics. Analysing the types of ships entering and leaving ports can help in understanding the global trade landscape and predicting economic trends.

Search and Rescue Operations: In cases of maritime emergencies, such as shipwrecks or distress calls, quickly identifying the type of vessel involved can aid search and rescue operations. It allows responders to tailor their efforts and resources accordingly.

LITERATURE SURVEY

Existing problem

The current state of ship classification in the maritime industry is labour-intensive, error-prone, and heavily reliant on manual efforts. This manual classification process is slow and often inaccurate due to the diversity and volume of ships entering and leaving ports. It hinders maritime safety, security, logistics, environmental monitoring, insurance, and market analysis efforts.

Automated ship classification is essential to address these challenges, but a reliable and scalable solution, leveraging deep learning with models like Inception v3, is currently lacking. Developing such a system is necessary to improve the efficiency and effectiveness of various maritime operations and enhance safety and security in the maritime domain.

References

https://ieeexplore.ieee.org/abstract/document/9170750

https://www.mdpi.com/1424-8220/18/9/2929

https://www.mdpi.com/2078-2489/12/8/302

https://ieeexplore.ieee.org/abstract/document/8455679

Problem Statement Definition

Efficient ship classification is crucial for maritime safety, security, logistics, and environmental monitoring. Existing methods often rely on manual labour and are error-prone.

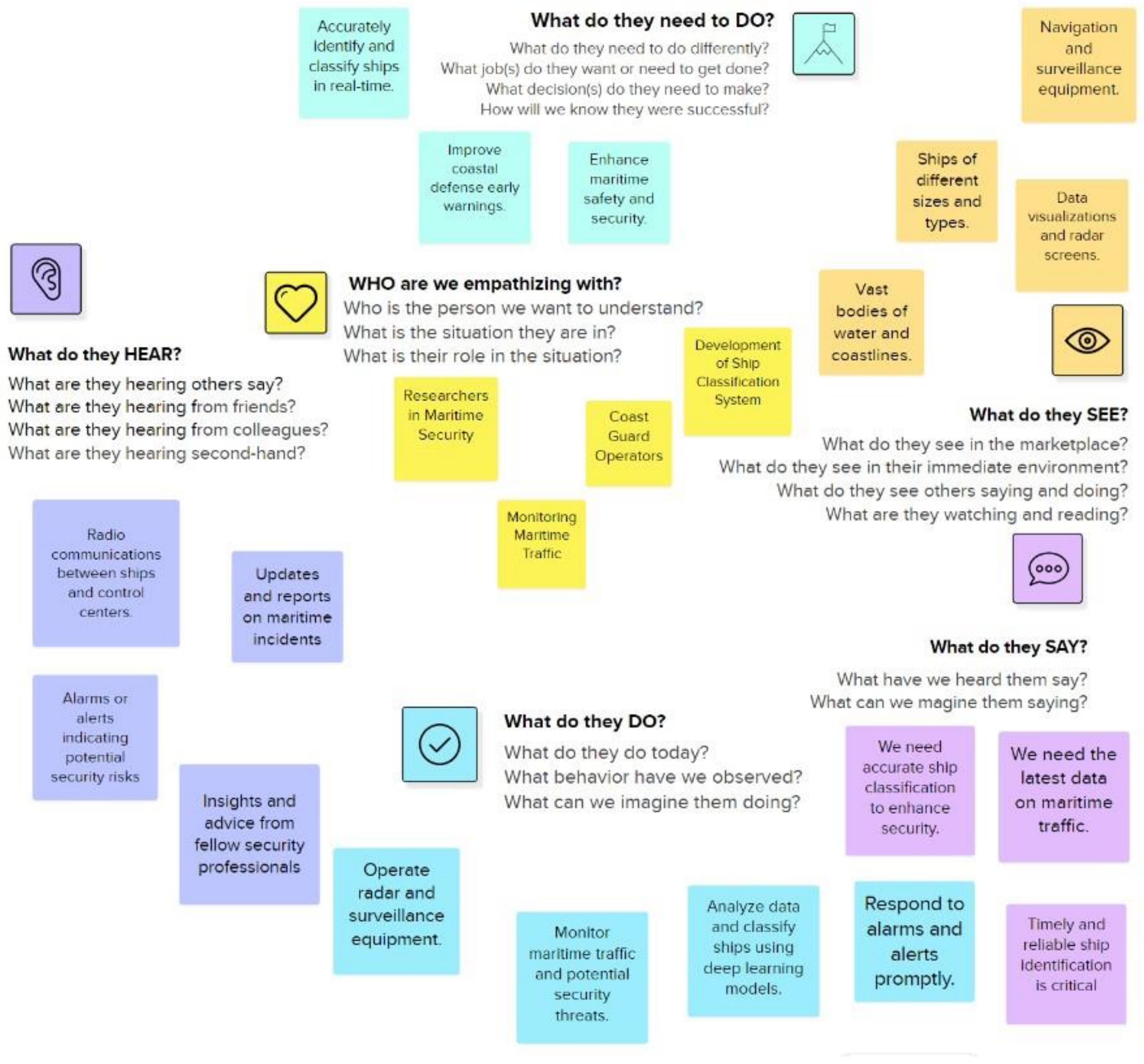
There is a pressing need for an automated ship classification system based on Inception v3 deep learning to categorize ships as cargo carriers, military tankers, and cruise ships. This project aims to design and implement such a system to enhance various maritime operations and contribute to safety, security, and environmental protection.

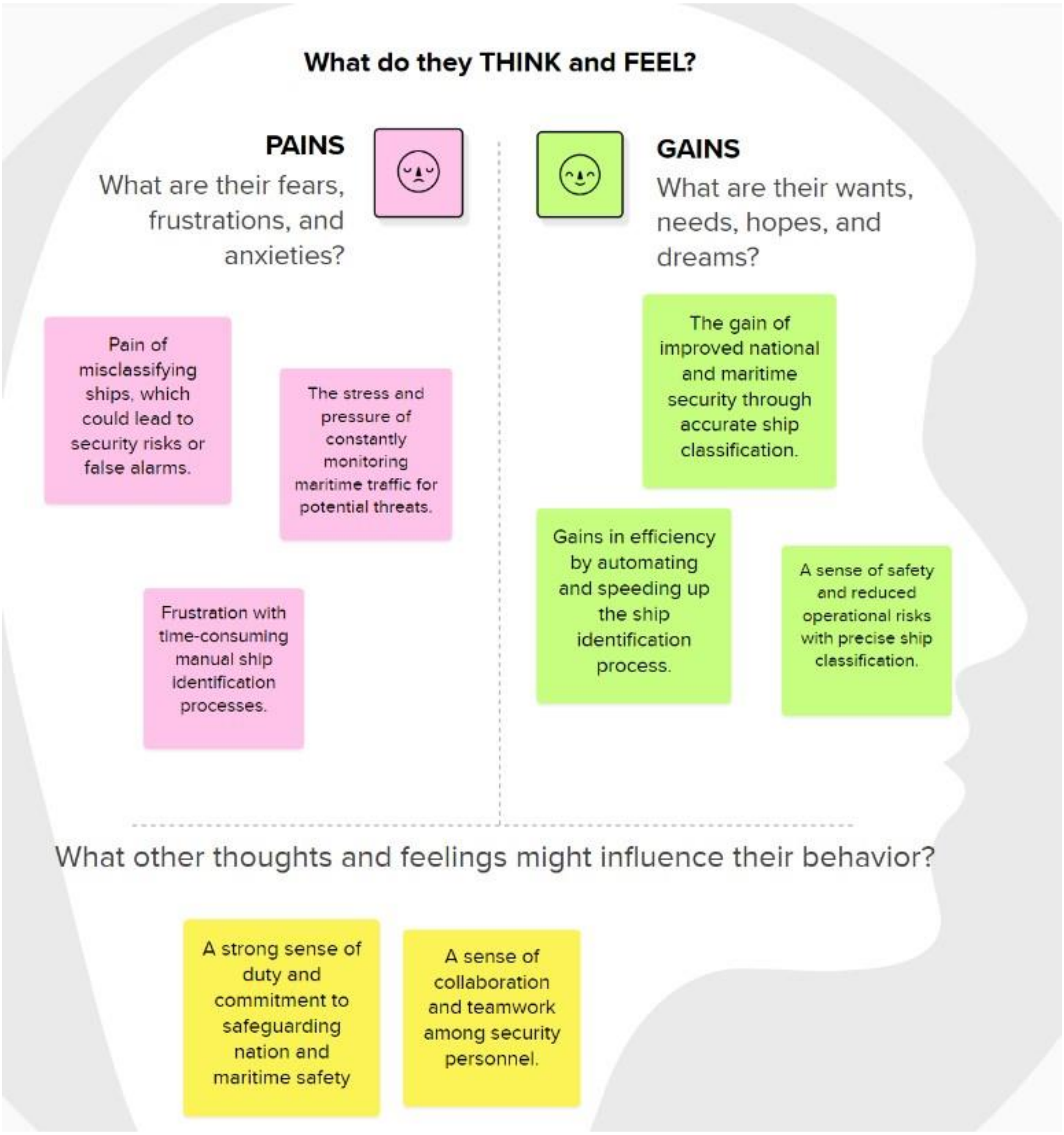
IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to helps teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.


What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Accurately identify and classify ships in real-time.
Improve coastal defense early warnings.
Enhance maritime safety and security.
Navigation and surveillance equipment.
Ships of different sizes and types.
Data visualizations and radar screens.
Vast bodies of water and coastlines.
WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
Development of Ship Classification System
Researchers in Maritime Security
Coast Guard Operators
Monitoring Maritime Traffic
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
Radio communications between ships and control centers.
Updates and reports on maritime incidents
Alarms or alerts indicating potential security risks
Insights and advice from fellow security professionals
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
What do they SAY?
What have we heard them say?
What can we magine them saying?
We need accurate ship classification to enhance security.
We need the latest data on maritime traffic.
Timely and reliable ship identification is critical
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Operate radar and surveillance equipment.
Monitor maritime traffic and potential security threats.
Analyze data and classify ships using deep learning models.
Respond to alarms and alerts promptly.

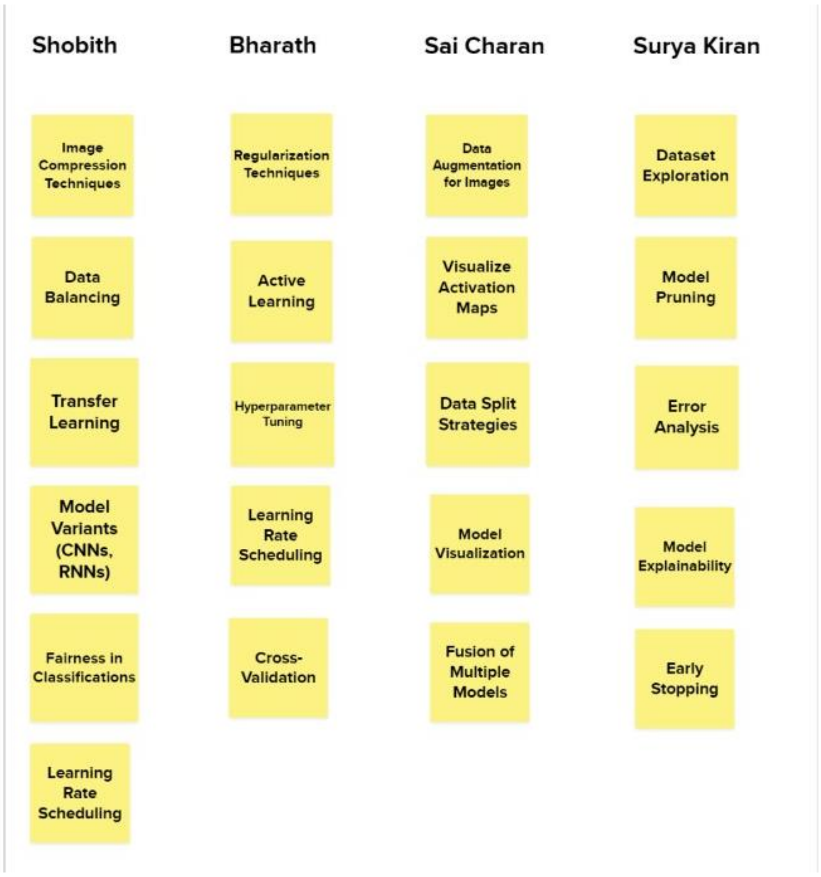

What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
Pain of misclassifying ships, which could lead to security risks or false alarms.
The stress and pressure of constantly monitoring maritime traffic for potential threats.
Frustration with time-consuming manual ship identification processes.
The gain of improved national and maritime security through accurate ship classification.
Gains in efficiency by automating and speeding up the ship identification process.
A sense of safety and reduced operational risks with precise ship classification.
What other thoughts and feelings might influence their behavior?
A strong sense of duty and commitment to safeguarding nation and maritime safety
A sense of collaboration and teamwork among security personnel.

<u>Ideation & Brainstorming</u>

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving.

Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich number of creative solutions.

Ideas



Shobith

- Image Compression Techniques
- Data Balancing
- Transfer Learning
- Model Variants (CNNs, RNNs)
- Fairness in Classifications
- Learning Rate Scheduling

Bharath

- Regularization Techniques
- Active Learning
- Hyperparameter Tuning
- Learning Rate Scheduling
- Cross-Validation

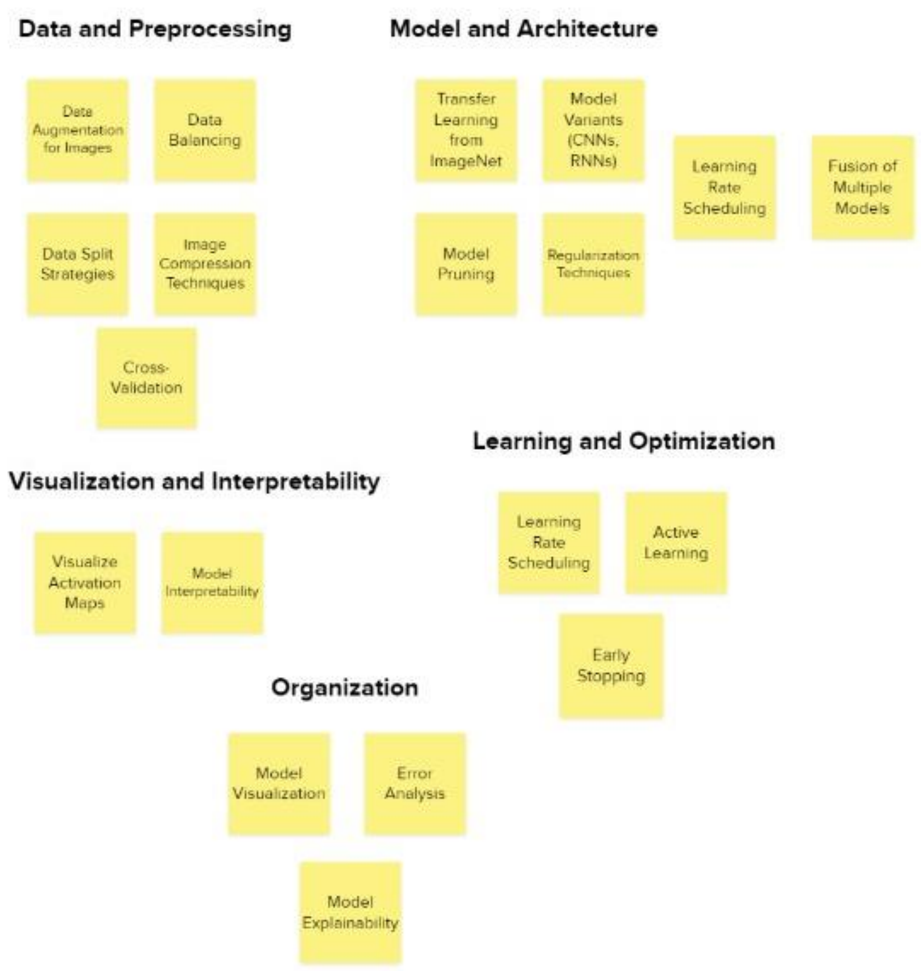
Sai Charan

- Data Augmentation for Images
- Visualize Activation Maps
- Data Split Strategies
- Model Visualization
- Fusion of Multiple Models

Surya Kiran

- Dataset Exploration
- Model Pruning
- Error Analysis
- Model Explainability
- Early Stopping

Grouped Ideas



Data and Preprocessing

- Data Augmentation for Images
- Data Balancing
- Data Split Strategies
- Image Compression Techniques
- Cross-Validation

Model and Architecture

- Transfer Learning from ImageNet
- Model Variants (CNNs, RNNs)
- Learning Rate Scheduling
- Fusion of Multiple Models
- Model Pruning
- Regularization Techniques

Learning and Optimization

- Learning Rate Scheduling
- Active Learning
- Early Stopping

Visualization and Interpretability

- Visualize Activation Maps
- Model Interpretability

Organization

- Model Visualization
- Error Analysis
- Model Explainability

REQUIREMENT ANALYSIS

Functional Requirements

The functional requirements for the ship classification system utilizing Inception v3 deep learning encompass several crucial aspects.

Firstly, the system should accept ship images as input for classification, subsequently categorizing them into cargo carriers, military tankers, and cruise ships. To ensure accurate results, the system must achieve a minimum classification accuracy of 85% and deliver near-real-time classification capabilities, producing results within seconds of image input.

Moreover, it should support data preprocessing steps such as image resizing and pixel value normalization. The system must also offer features for model training, fine-tuning, and evaluation, providing key metrics like training accuracy and validation accuracy for performance assessment.

A user-friendly interface is essential for uploading and processing ship images, and the system should seamlessly integrate with various data sources like surveillance cameras, satellite imagery, and image files.

Non-Functional Requirements

The ship classification system utilizing Inception v3 deep learning is expected to meet a range of non-functional requirements to ensure its effectiveness and performance.

First and foremost, the system must exhibit exceptional performance, classifying ships with minimal latency and maintaining high efficiency even under heavy loads. Scalability is crucial, enabling the system to accommodate a growing user base and increasing image inputs without a substantial reduction in performance.

Reliability is paramount, with the system required to have a low probability of failure or errors in ship classification. Availability is another key aspect, necessitating that the system be accessible 24/7 with minimal downtime for maintenance or updates.

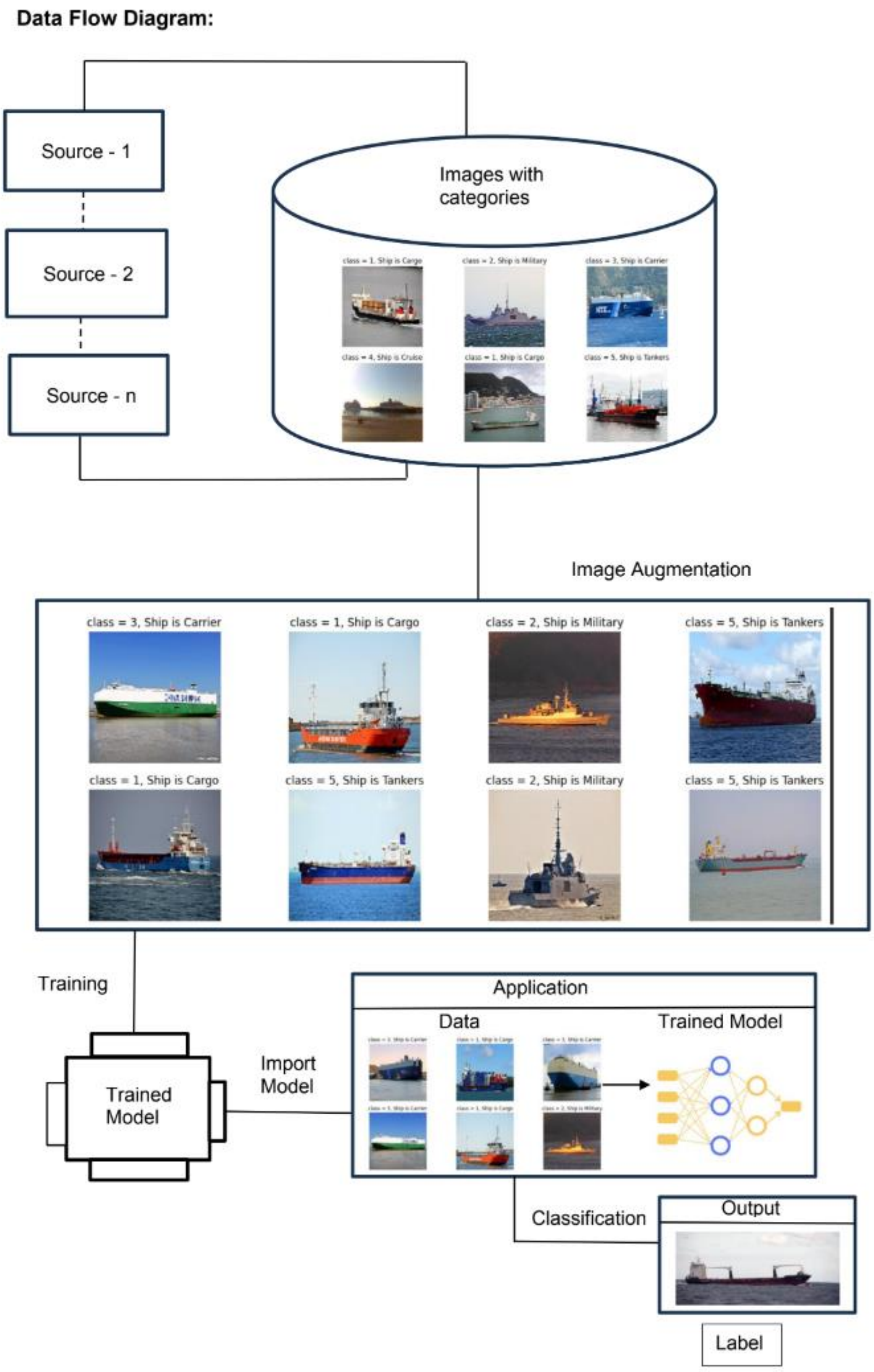
Usability is a non-negotiable requirement, demanding an intuitive and user-friendly interface that ensures straightforward user interactions and necessitates minimal training. Interoperability is critical for reaching a broad user base, requiring compatibility with different operating systems and web browsers.

The system must also comply with legal and regulatory requirements, particularly regarding data privacy and security. The system should handle peak loads and surges in usage efficiently without compromising performance or data integrity.

PROJECT DESIGN

Data Flow Diagrams & User Stories

Data flows from the Dataset to the Data Preprocessing process, where it is transformed into pre-processed data. The pre-processed data flows to the Model Training process. The trained model can be used for classification. Separate evaluation data is used in the Model Evaluation process, and the results are used for assessing the accuracy.


Data Flow Diagram:
Source - 1
Source - 2
Source - n
Images with categories
class = 1, Ship is Cargo
class = 2, Ship is Military
class = 3, Ship is Carrier
class = 4, Ship is Cruise
class = 5, Ship is Tankers
Image Augmentation
class = 3, Ship is Carrier
class = 1, Ship is Cargo
class = 2, Ship is Military
class = 5, Ship is Tankers
class = 1, Ship is Cargo
class = 5, Ship is Tankers
class = 2, Ship is Military
class = 5, Ship is Tankers
Training
Trained Model
Import Model
Application
Data
Trained Model
Classification
Output
Label

User Stories

User Type	Functional Requirement	User Story No.	User Story/Task	Acceptance Criteria	Priority	Release
Maritime Security	Project setup and Infrastructure	User –1	To set up the development of the ship classification with the required tools and frameworks.	Successful Configuration	High	Sprint 1
Coastal Guards	Development environment	User - 2	Gather a diverse dataset of images containing different types of ships (Carrier, Cargo, Cruise, Tanker, Military) for the training of the deep learning model.	Gathered a diverse dataset of images depicting various types of ships	High	Sprint 1
Individuals	Data collection	User –3	Preprocess the collected dataset by resizing the images, normalizing the pixel values and splitting it into training and test sets.	Preprocessing and the splitting of the ship dataset	High	Sprint 2
Researchers and Academics	Data Preprocessing	User –4	Explore and evaluate various deep learning architectures (E.g., CNN) to select the most suitable model for ship classification.	Exploring the different deep learning models	High	Sprint 2
Organizations	Model Development	User –5	Train the selected deep learning model using the pre-processed dataset and monitor its performance on the validation set.	Validation using the test part of the dataset	High	Sprint 3
Institutions	Training	User –6	Implement data augmentation techniques to improve the robustness and accuracy of the model.	Test it with the augmented dataset	Medium	Sprint 4
	Model Deployment	User –7	Deploy the trained deep learning model as an API or web service and make it accessible for ship classification.	Check the scalability of the model	Medium	Sprint 4
	Testing and Quality Assurance	User –8	Conduct a thorough testing of the model and web interface to identify and report any issues of bugs, finetune the parameters and optimize the performance according to the user performance.	Creating the web application	Medium	Sprint 6

Solution Architecture

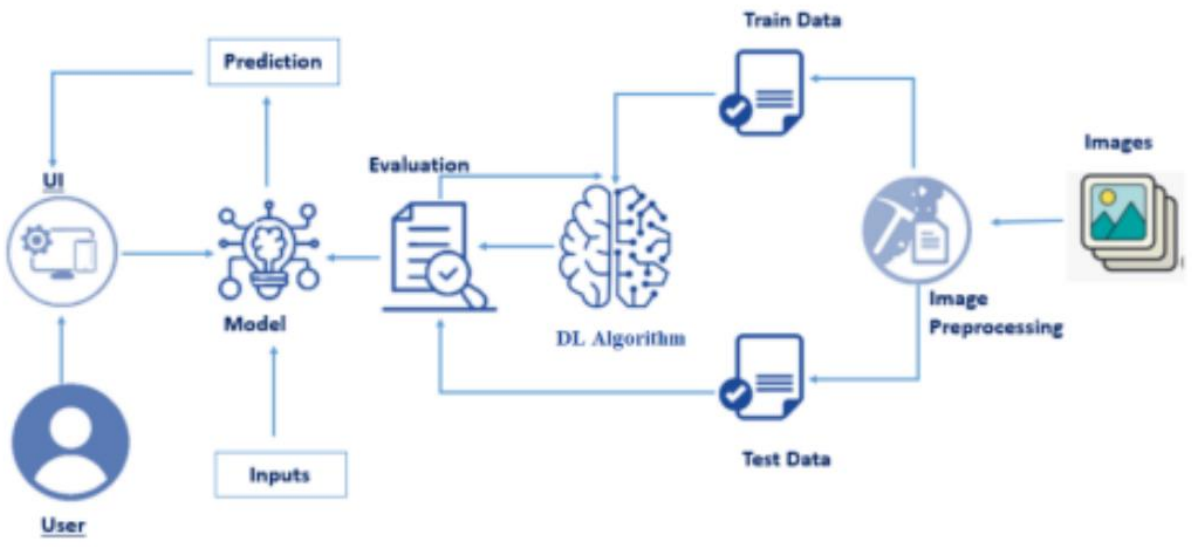
Our solution for ship classification employs a sophisticated architecture, drawing upon deep learning models, primarily using transfer learning and Convolutional Neural Networks (CNNs). This architecture is designed to significantly enhance the ship classification process, offering a comprehensive approach to maritime safety and traffic management.

Key Components:

1. Deep Learning Models: The core of our architecture is the deep learning models, which are trained on extensive datasets comprising various ship types, sizes, and environmental conditions. Transfer learning allows us to leverage pre-trained models and adapt them to ship classification tasks, expediting the learning process.

2. Continuous Learning Loop: One of the distinctive features of our architecture is the continuous learning loop. As new data becomes available, the system updates its knowledge, ensuring it remains adaptable and accurate in classifying ships. This adaptability is crucial for real-world scenarios where ship types and behaviours can evolve over time.

3. Real-time Classification: Our system is optimized for real-time ship classification. It can swiftly analyse incoming data, such as radar information, satellite images, and AIS data, to provide instant ship type identification. This capability is essential for prompt decision-making and response in maritime operations.



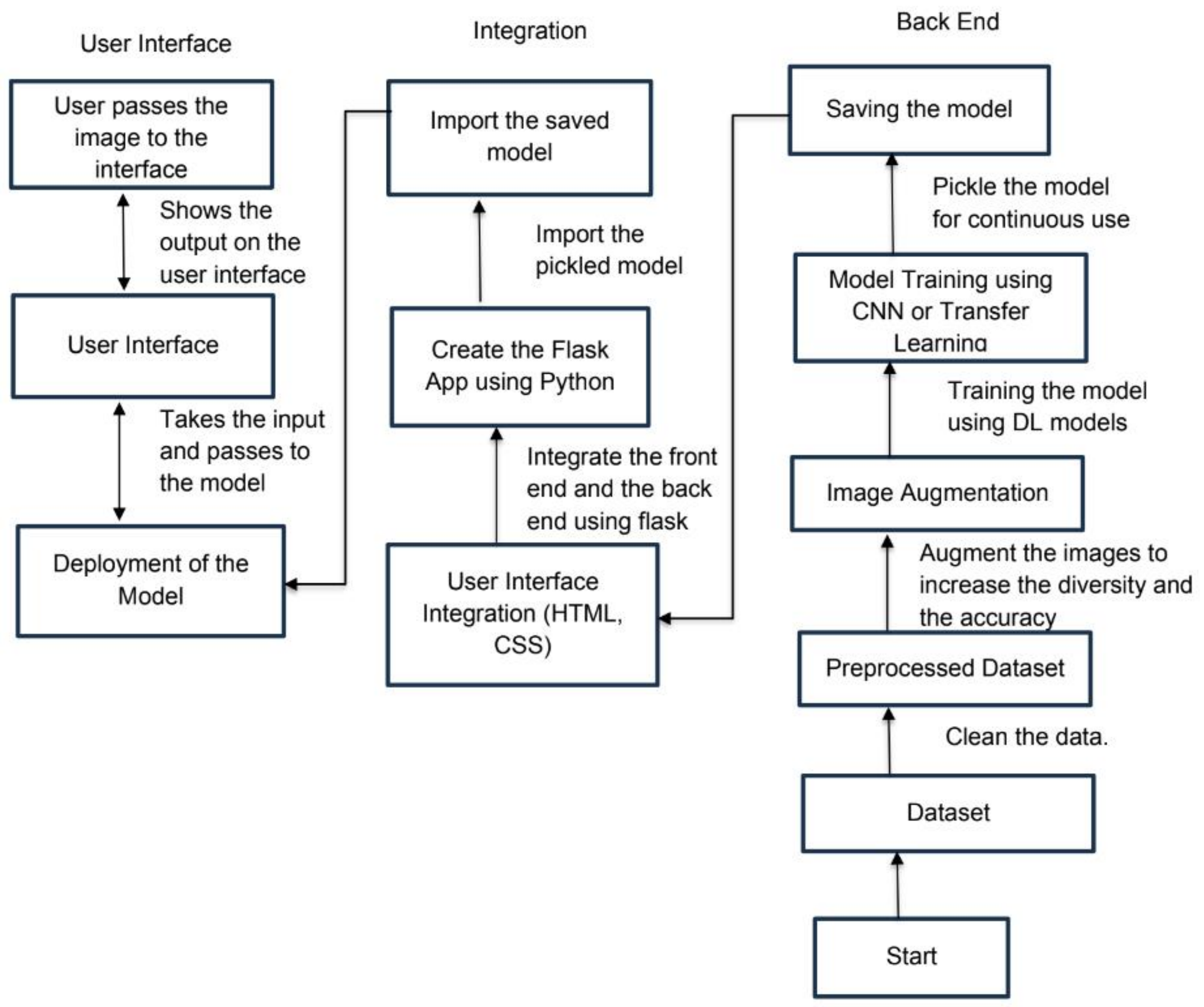
PROJECT PLANNING & SCHEDULING

Technical Architecture

The ship classification system's technical structure is a process that starts with the deployment of Inception v3, to classify ships. Users interact with the system by providing ship images through the user interface. When required, the system imports the pre-trained model to assist in ship classification.

The behind-the-scenes work is done using Flask, a Python web framework, ensuring a smooth user experience by connecting the front-end and back-end. The union of the user interface and the back-end is a critical step that ties everything together. As the system progresses, image augmentation techniques enrich the dataset, and the trained model is preserved for continuous and efficient ship classification.

In the core of the architecture, deep learning models are employed to train and fine-tune the system, with the dataset being the fundamental element containing the ship images essential for training and validation.


User Interface
User passes the image to the interface
Shows the output on the user interface
User Interface
Takes the input and passes to the model
Deployment of the Model
Integration
Import the saved model
Import the pickled model
Create the Flask App using Python
Integrate the front end and the back end using flask
User Interface Integration (HTML, CSS)
Back End
Saving the model
Pickle the model for continuous use
Model Training using CNN or Transfer Learning
Training the model using DL models
Image Augmentation
Augment the images to increase the diversity and the accuracy
Preprocessed Dataset
Clean the data.
Dataset
Start

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story No.	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup and Infrastructure	USN-1	Set up the development environment with the required tools and frameworks to start the ship classification project.	3	High	Shobith
Sprint-1	Development Environment	USN-2	Gather a diverse dataset of images containing different types of ships (Military, Cargo, Cruise, Carrier, Tanker) for training the deep learning model.	2	High	Shobith
Sprint-2	Data Collection	USN-3	Preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	2	High	Bharath
Sprint-2	Data Preprocessing	USN-4	Explore and evaluate different deep learning architectures (e.g., CNNs) to select the most suitable model for ship classification.	3	High	Bharath
Sprint-3	Model Development	USN-5	Train the selected deep learning model using the pre-processed dataset and monitor its performance on the validation set.	3	High	Surya Kiran
Sprint-3	Training	USN-6	Implement data augmentation techniques (e.g., rotation, flipping) to improve the model's robustness and accuracy.	2	Medium	Surya Kiran
Sprint-4	Model Deployment & Integration	USN-7	Deploy the trained deep learning model as an API or web service to make it accessible for garbage classification.	2	Medium	Sai Charan
Sprint-5	Testing & Quality Assurance	USN-8	Conduct thorough testing of the web interface to identify and report any issues or bugs and optimize its performance based on user feedback and testing results.	3	Medium	Sai Charan

Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Story Points Completed	Sprint Release Date (Actual)
Sprint-1	5	3 Days	26 Oct 2023	29 Oct 2023	5	29 Oct 2023
Sprint-2	5	3 Days	29 Oct 2023	31 Oct 2023	5	31 Oct 2023
Sprint-3	5	2 Days	1 Nov 2023	3 Nov 2023	5	
Sprint-4	2	2 Days	3 Nov 2023	5 Nov 2023	5	
Sprint-5	3	1 Days	5 Nov 2023	6 Nov 2023	5	

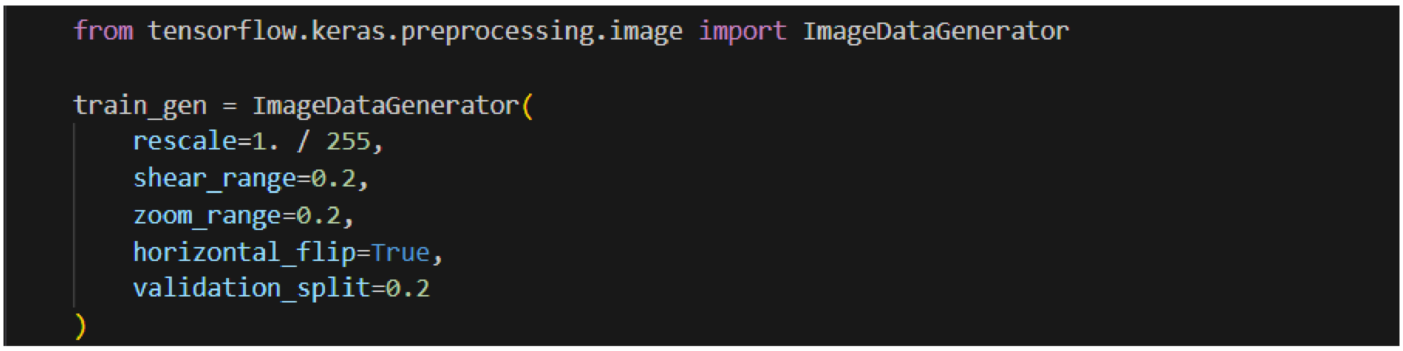
CODING & SOLUTIONING

Data Pre-Processing

The dataset images are to be pre-processed before giving it to the model.

Import ImageDataGenerator Library and Configure it

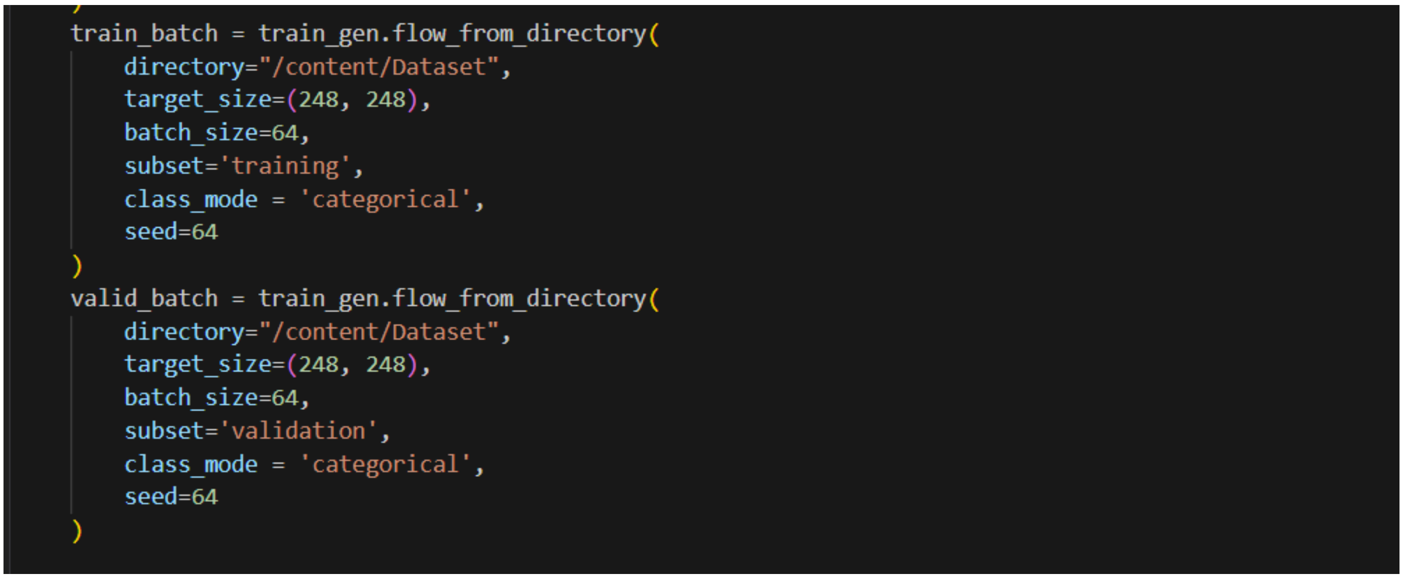
ImageDataGenerator class is used to augment the images with different modifications like considering the rotation, flipping the image etc.



```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_gen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.2
)
```

Apply ImageDataGenerator functionality to Train and Validation set

Specify the path of both the folders in flow_from_directory method. We are importing the images in 248x248 pixels.



```
    )
    train_batch = train_gen.flow_from_directory(
        directory="/content/Dataset",
        target_size=(248, 248),
        batch_size=64,
        subset='training',
        class_mode = 'categorical',
        seed=64
    )
    valid_batch = train_gen.flow_from_directory(
        directory="/content/Dataset",
        target_size=(248, 248),
        batch_size=64,
        subset='validation',
        class_mode = 'categorical',
        seed=64
    )
```

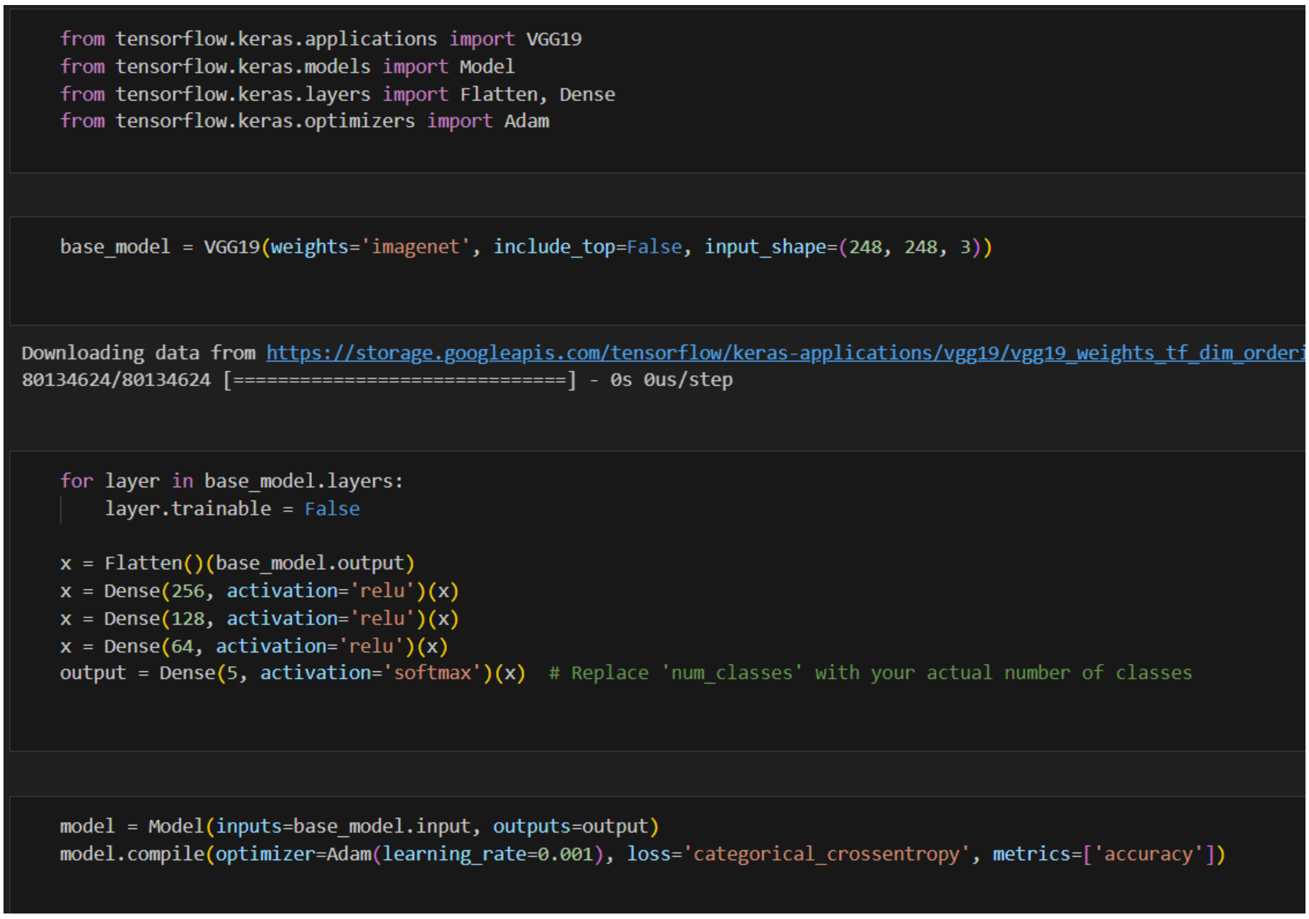
Building the model

We will be creating the pre-trained VGG16 model and Inception V3 model along with ResNet50 model for predicting custom classes and choose the model with the highest accuracy.

Creating and compiling the model

VGG19

Firstly, import the necessary libraries and define a function for creation of the model. Next, call the pre trained model with the parameter include_top=False, as we need to use the model for predicting custom images. Next, add dense layers to the top model. Finally, add an output layer with the number of neurons equal to out output classes.



```
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.optimizers import Adam
```

```
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(248, 248, 3))
```

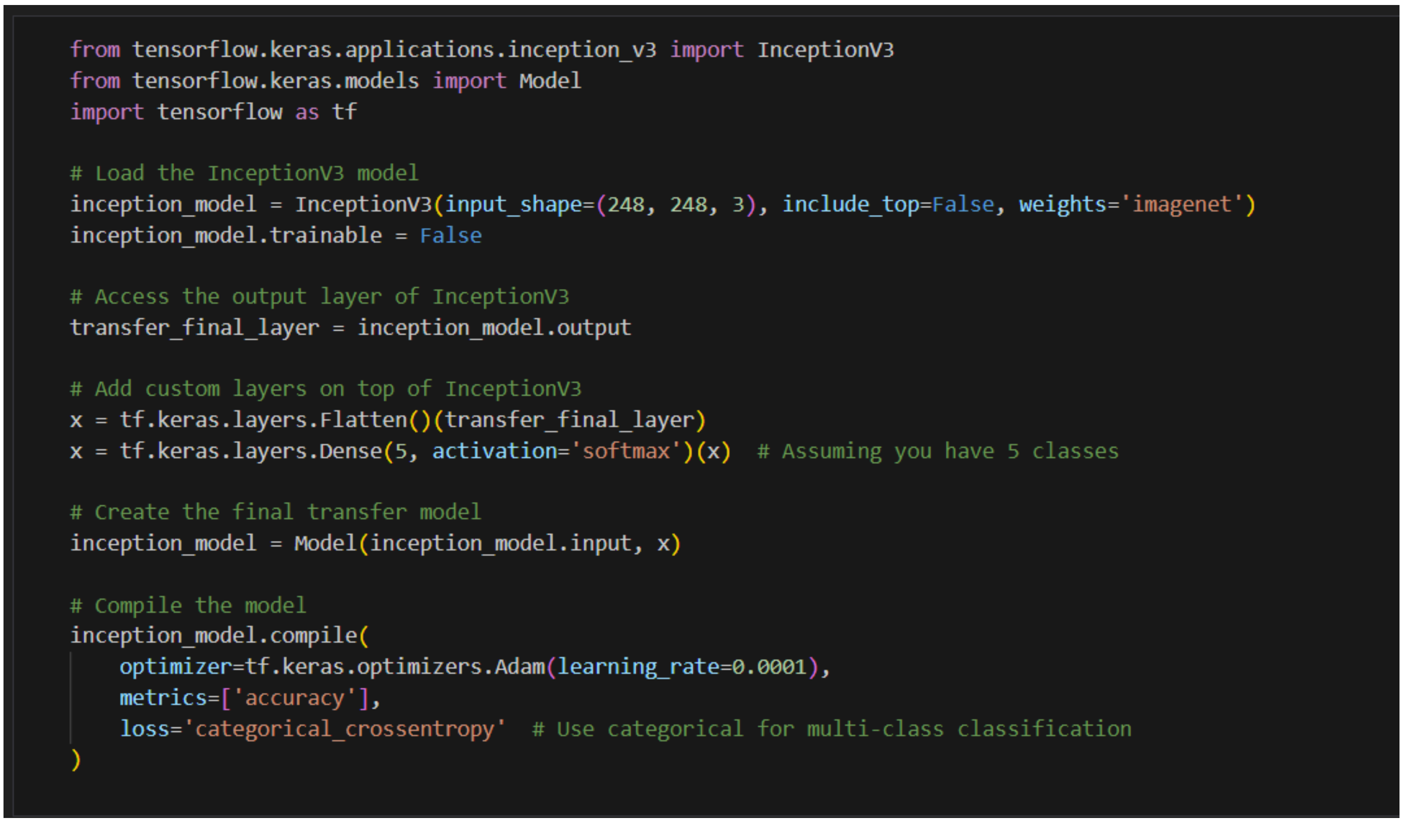
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_orderi
80134624/80134624 [==============================] - 0s 0us/step
```

```
for layer in base_model.layers:
    layer.trainable = False

x = Flatten()(base_model.output)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
output = Dense(5, activation='softmax')(x)  # Replace 'num_classes' with your actual number of classes
```

```
model = Model(inputs=base_model.input, outputs=output)
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Inception V3

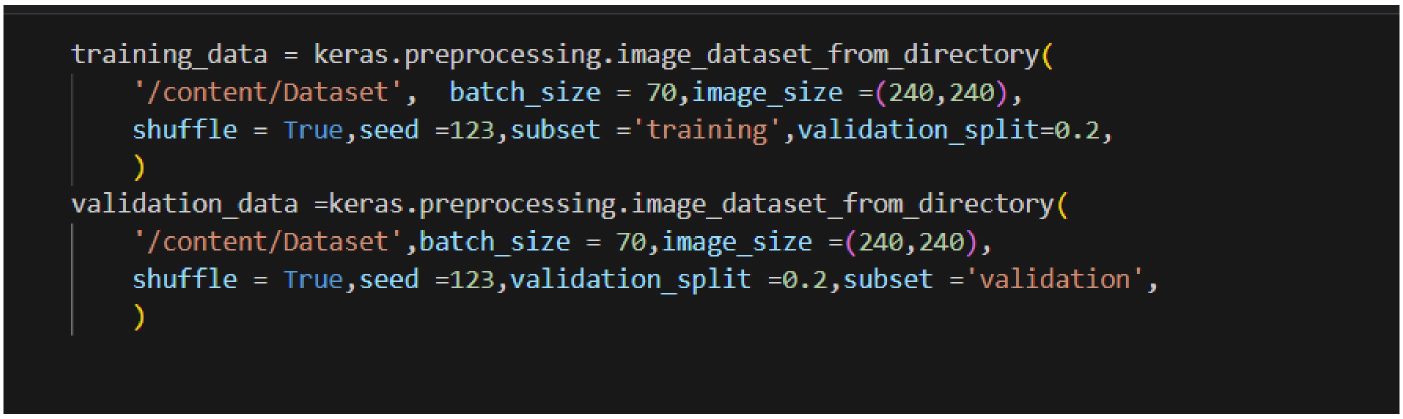
Firstly, import the necessary libraries and define a function for creation of the model. Next, call the pre trained model with the parameter include_top=False, as we need to use the model for predicting custom images. Next, add dense layers to the top model. Finally, add an output layer with the number of neurons equal to out output classes.



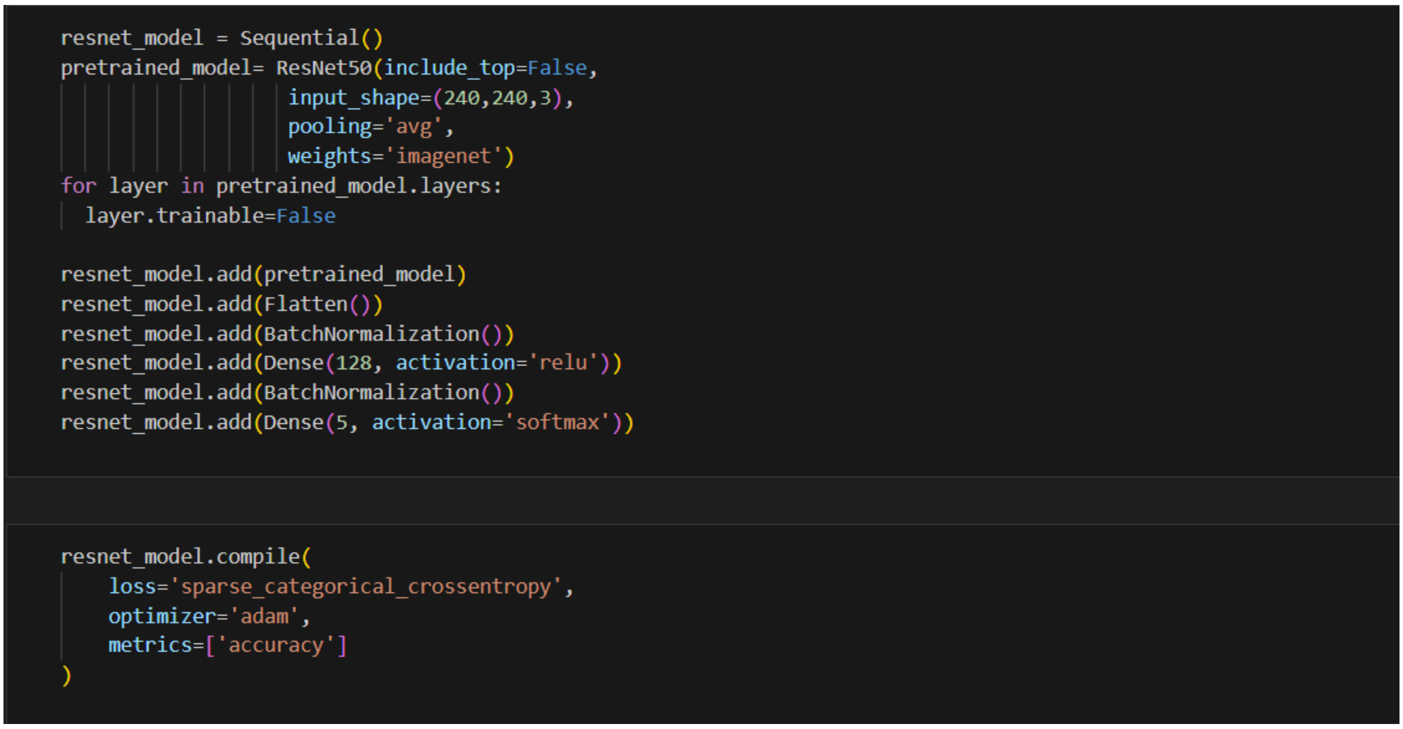
```python
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Model
import tensorflow as tf

# Load the InceptionV3 model
inception_model = InceptionV3(input_shape=(248, 248, 3), include_top=False, weights='imagenet')
inception_model.trainable = False

# Access the output layer of InceptionV3
transfer_final_layer = inception_model.output

# Add custom layers on top of InceptionV3
x = tf.keras.layers.Flatten()(transfer_final_layer)
x = tf.keras.layers.Dense(5, activation='softmax')(x)  # Assuming you have 5 classes

# Create the final transfer model
inception_model = Model(inception_model.input, x)

# Compile the model
inception_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    metrics=['accuracy'],
    loss='categorical_crossentropy'  # Use categorical for multi-class classification
)
```

ResNet 50

Firstly, import the necessary libraries and define a function for creation of the model. Next, call the pre trained model with the parameter include_top=False, as we need to use the model for predicting custom images. Next, add dense layers to the top model. Finally, add an output layer with the number of neurons equal to out output classes.



```python
training_data = keras.preprocessing.image_dataset_from_directory(
    '/content/Dataset',  batch_size = 70,image_size =(240,240),
    shuffle = True,seed =123,subset ='training',validation_split=0.2,
    )
validation_data =keras.preprocessing.image_dataset_from_directory(
    '/content/Dataset',batch_size = 70,image_size =(240,240),
    shuffle = True,seed =123,validation_split =0.2,subset ='validation',
    )
```

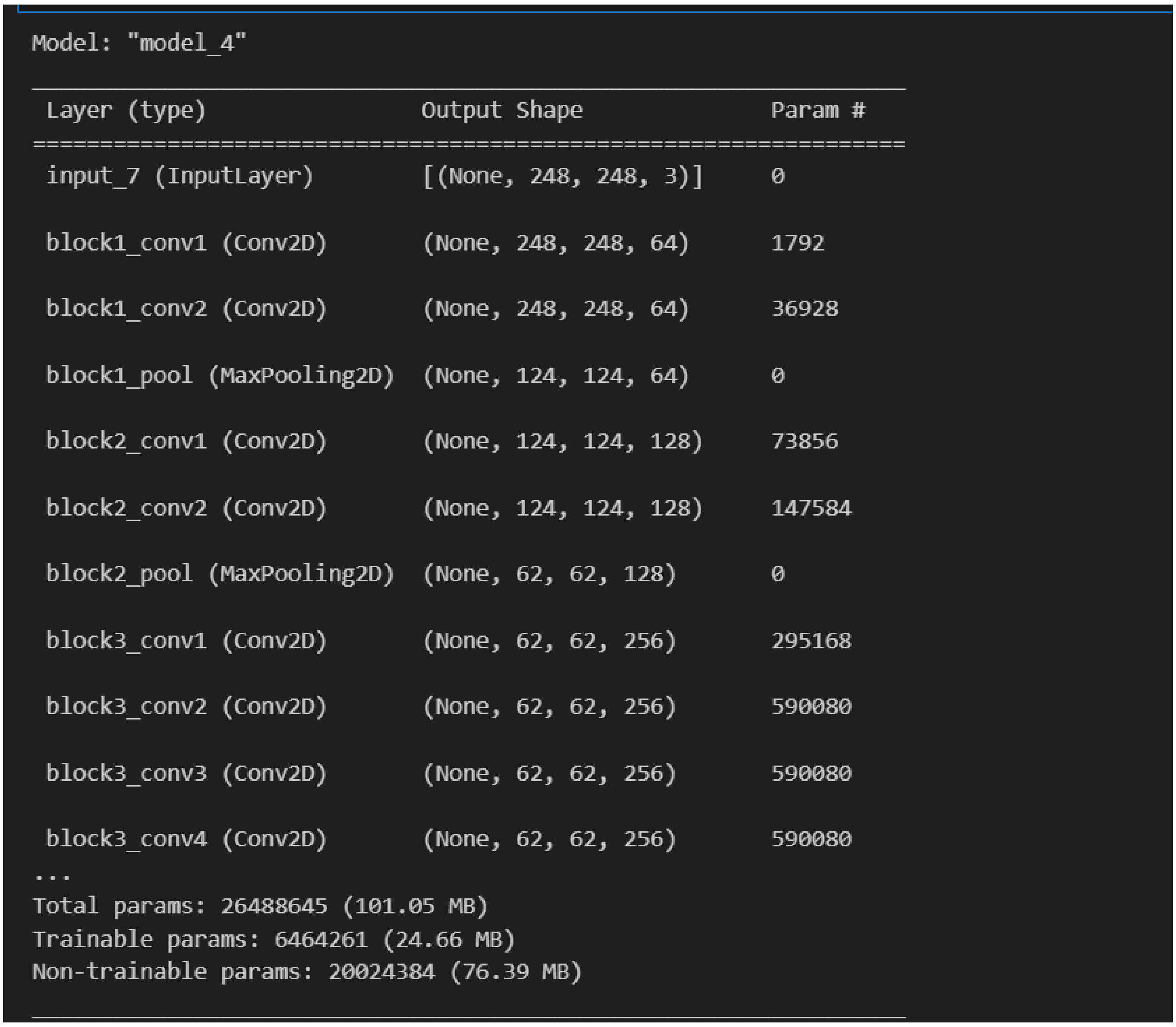


```
resnet_model = Sequential()
pretrained_model= ResNet50(include_top=False,
                           input_shape=(240,240,3),
                           pooling='avg',
                           weights='imagenet')
for layer in pretrained_model.layers:
  layer.trainable=False

resnet_model.add(pretrained_model)
resnet_model.add(Flatten())
resnet_model.add(BatchNormalization())
resnet_model.add(Dense(128, activation='relu'))
resnet_model.add(BatchNormalization())
resnet_model.add(Dense(5, activation='softmax'))
```

```
resnet_model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

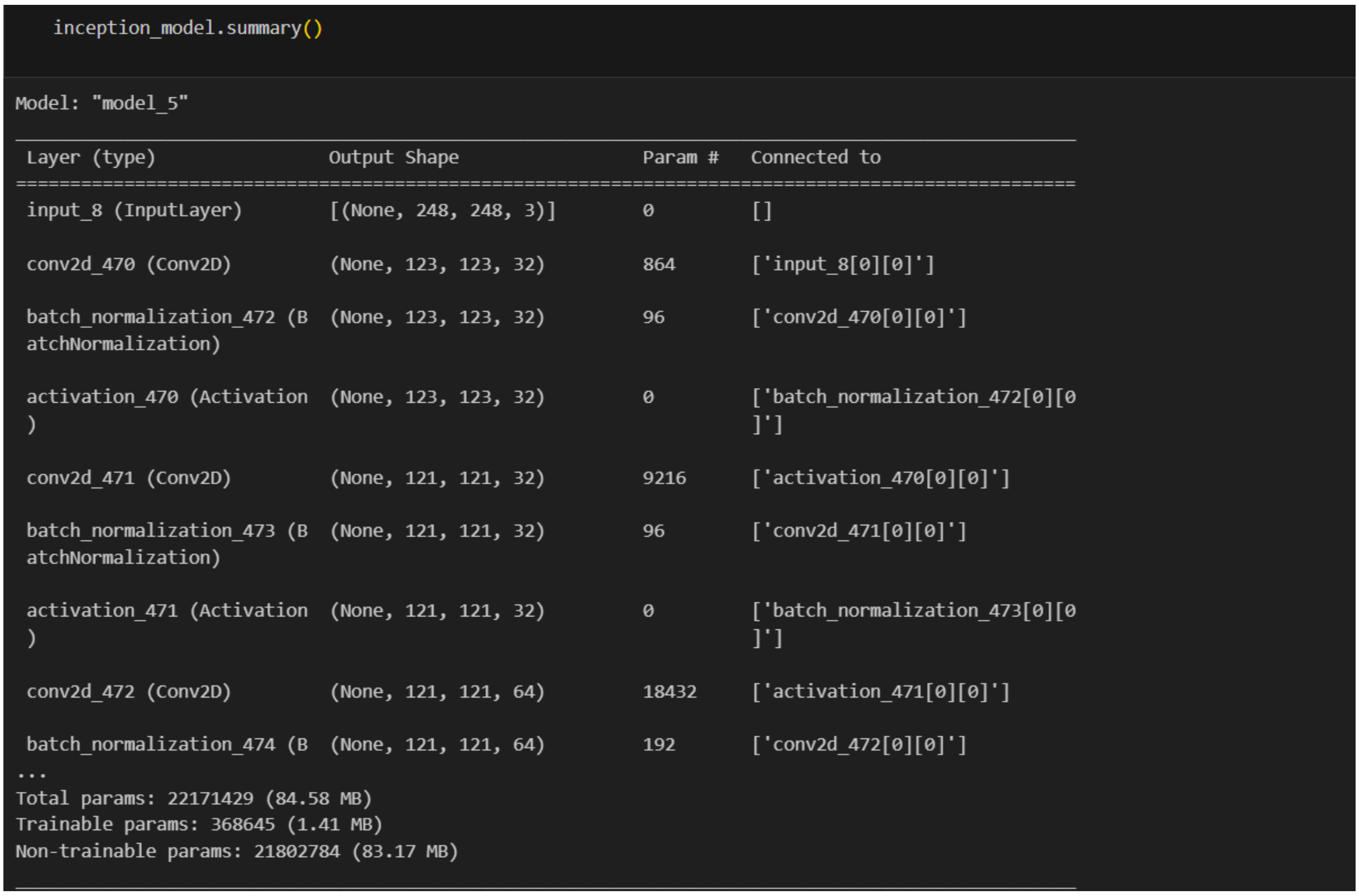
Model Summary

VGG19



```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 248, 248, 3)]     0

 block1_conv1 (Conv2D)       (None, 248, 248, 64)      1792

 block1_conv2 (Conv2D)       (None, 248, 248, 64)      36928

 block1_pool (MaxPooling2D)  (None, 124, 124, 64)      0

 block2_conv1 (Conv2D)       (None, 124, 124, 128)     73856

 block2_conv2 (Conv2D)       (None, 124, 124, 128)     147584

 block2_pool (MaxPooling2D)  (None, 62, 62, 128)       0

 block3_conv1 (Conv2D)       (None, 62, 62, 256)       295168

 block3_conv2 (Conv2D)       (None, 62, 62, 256)       590080

 block3_conv3 (Conv2D)       (None, 62, 62, 256)       590080

 block3_conv4 (Conv2D)       (None, 62, 62, 256)       590080
...
Total params: 26488645 (101.05 MB)
Trainable params: 6464261 (24.66 MB)
Non-trainable params: 20024384 (76.39 MB)
_________________________________________________________________
```

Inception V3

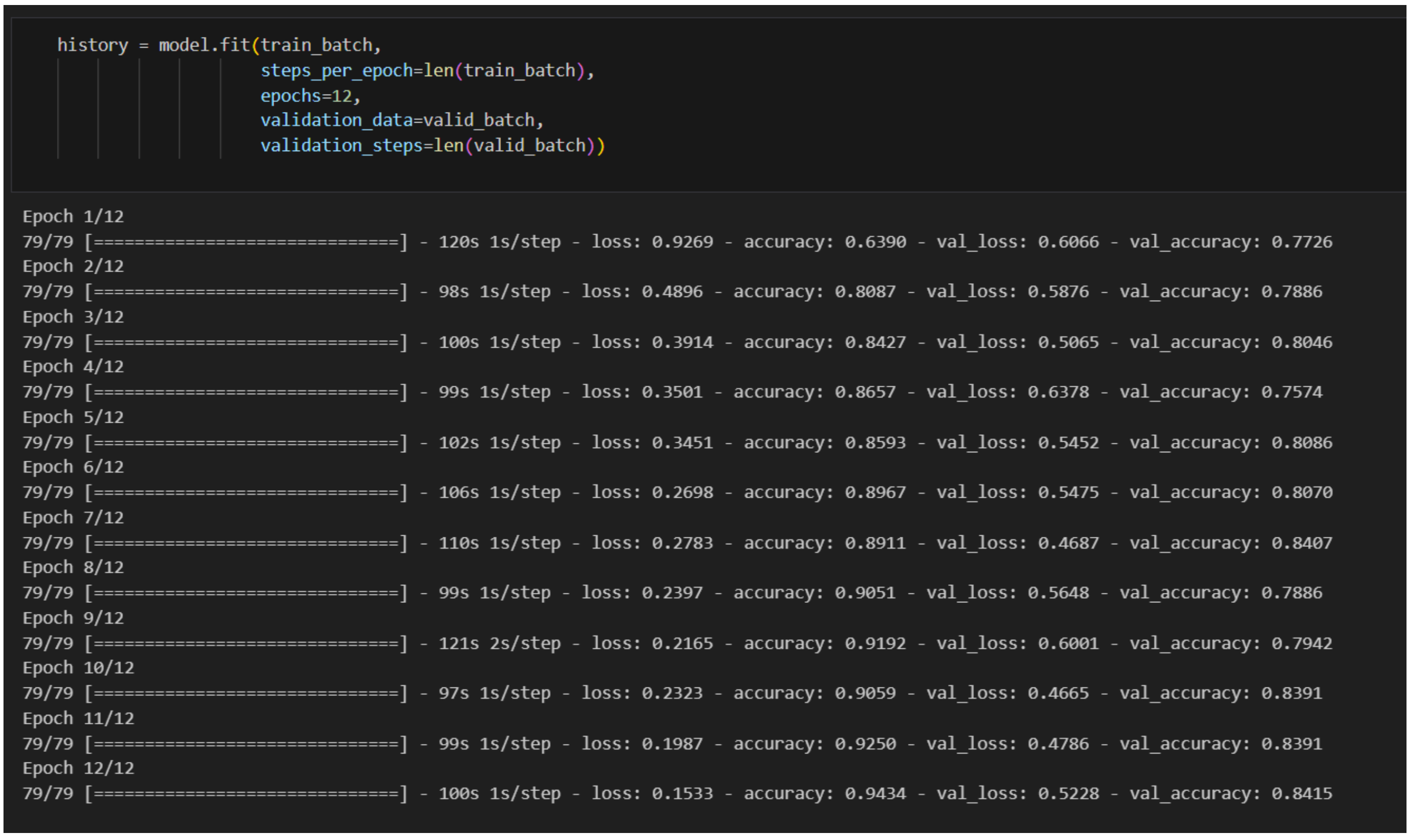


```
inception_model.summary()
```

```
Model: "model_5"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_8 (InputLayer)           [(None, 248, 248, 3)]  0         []

 conv2d_470 (Conv2D)            (None, 123, 123, 32)  864        ['input_8[0][0]']

 batch_normalization_472 (B     (None, 123, 123, 32)  96         ['conv2d_470[0][0]']
 atchNormalization)

 activation_470 (Activation     (None, 123, 123, 32)  0          ['batch_normalization_472[0][0
 )                                                                ]']

 conv2d_471 (Conv2D)            (None, 121, 121, 32)  9216       ['activation_470[0][0]']

 batch_normalization_473 (B     (None, 121, 121, 32)  96         ['conv2d_471[0][0]']
 atchNormalization)

 activation_471 (Activation     (None, 121, 121, 32)  0          ['batch_normalization_473[0][0
 )                                                                ]']

 conv2d_472 (Conv2D)            (None, 121, 121, 64)  18432      ['activation_471[0][0]']

 batch_normalization_474 (B     (None, 121, 121, 64)  192        ['conv2d_472[0][0]']
...
Total params: 22171429 (84.58 MB)
Trainable params: 368645 (1.41 MB)
Non-trainable params: 21802784 (83.17 MB)
__________________________________________________________________________________________________
```

PERFORMANCE TESTING

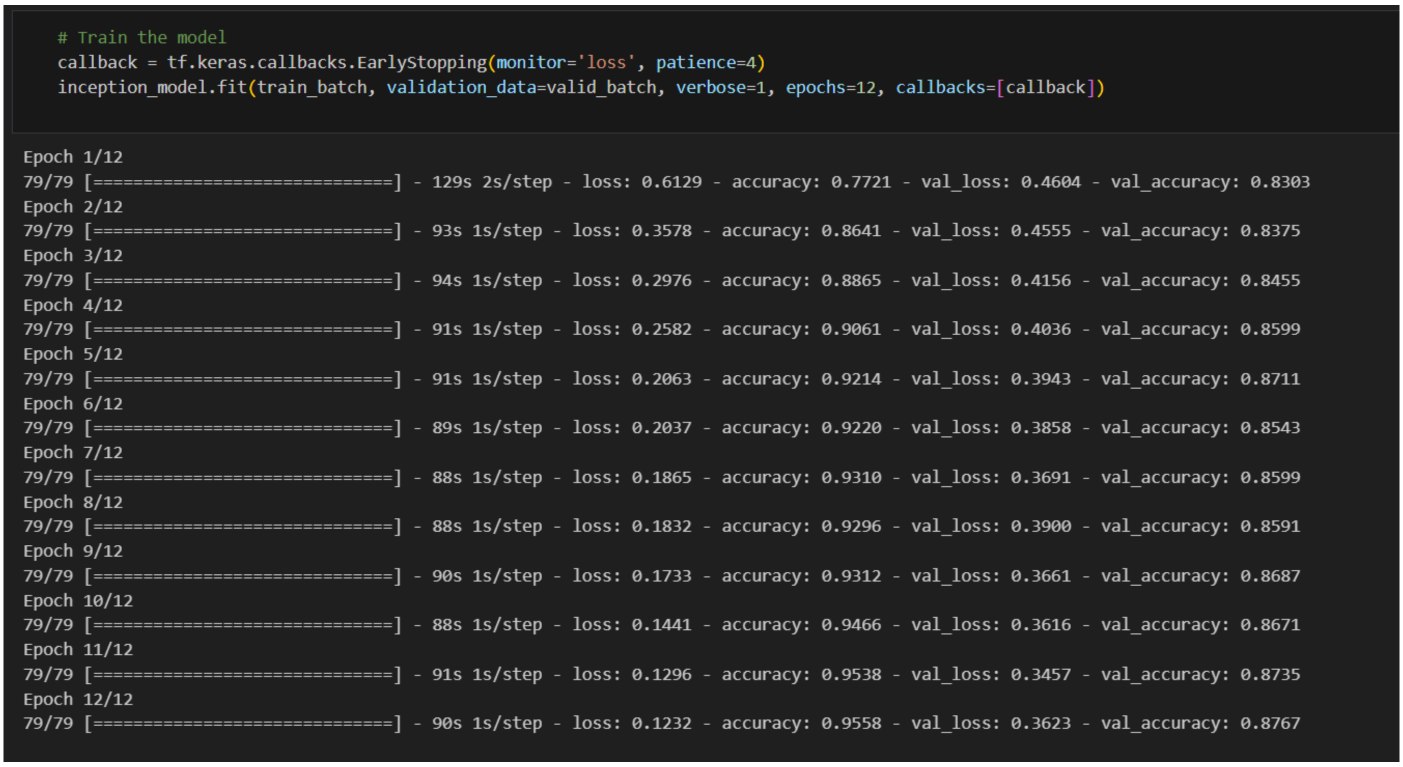
VGG19



```
history = model.fit(train_batch,
                    steps_per_epoch=len(train_batch),
                    epochs=12,
                    validation_data=valid_batch,
                    validation_steps=len(valid_batch))
```

```
Epoch 1/12
79/79 [==============================] - 120s 1s/step - loss: 0.9269 - accuracy: 0.6390 - val_loss: 0.6066 - val_accuracy: 0.7726
Epoch 2/12
79/79 [==============================] - 98s 1s/step - loss: 0.4896 - accuracy: 0.8087 - val_loss: 0.5876 - val_accuracy: 0.7886
Epoch 3/12
79/79 [==============================] - 100s 1s/step - loss: 0.3914 - accuracy: 0.8427 - val_loss: 0.5065 - val_accuracy: 0.8046
Epoch 4/12
79/79 [==============================] - 99s 1s/step - loss: 0.3501 - accuracy: 0.8657 - val_loss: 0.6378 - val_accuracy: 0.7574
Epoch 5/12
79/79 [==============================] - 102s 1s/step - loss: 0.3451 - accuracy: 0.8593 - val_loss: 0.5452 - val_accuracy: 0.8086
Epoch 6/12
79/79 [==============================] - 106s 1s/step - loss: 0.2698 - accuracy: 0.8967 - val_loss: 0.5475 - val_accuracy: 0.8070
Epoch 7/12
79/79 [==============================] - 110s 1s/step - loss: 0.2783 - accuracy: 0.8911 - val_loss: 0.4687 - val_accuracy: 0.8407
Epoch 8/12
79/79 [==============================] - 99s 1s/step - loss: 0.2397 - accuracy: 0.9051 - val_loss: 0.5648 - val_accuracy: 0.7886
Epoch 9/12
79/79 [==============================] - 121s 2s/step - loss: 0.2165 - accuracy: 0.9192 - val_loss: 0.6001 - val_accuracy: 0.7942
Epoch 10/12
79/79 [==============================] - 97s 1s/step - loss: 0.2323 - accuracy: 0.9059 - val_loss: 0.4665 - val_accuracy: 0.8391
Epoch 11/12
79/79 [==============================] - 99s 1s/step - loss: 0.1987 - accuracy: 0.9250 - val_loss: 0.4786 - val_accuracy: 0.8391
Epoch 12/12
79/79 [==============================] - 100s 1s/step - loss: 0.1533 - accuracy: 0.9434 - val_loss: 0.5228 - val_accuracy: 0.8415
```

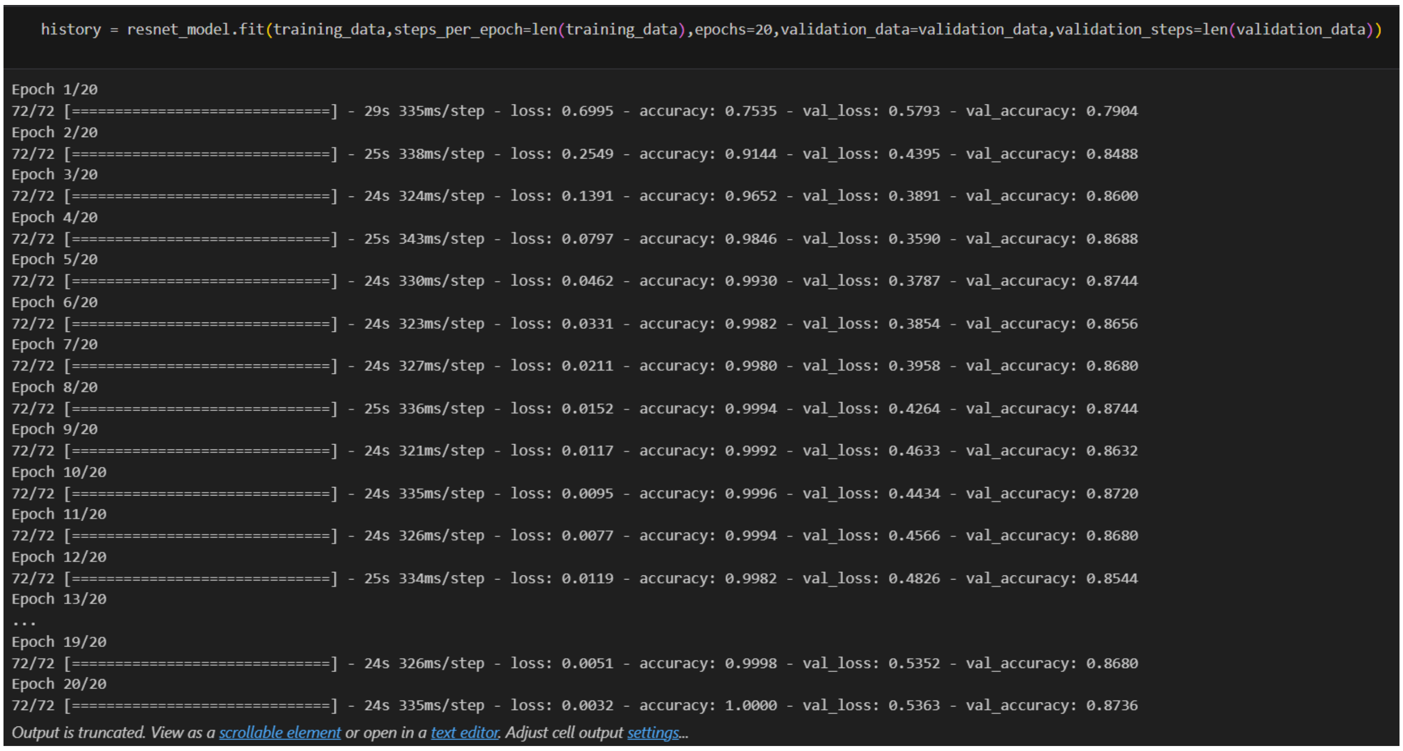
Inception V3



```
# Train the model
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=4)
inception_model.fit(train_batch, validation_data=valid_batch, verbose=1, epochs=12, callbacks=[callback])
```

```
Epoch 1/12
79/79 [==============================] - 129s 2s/step - loss: 0.6129 - accuracy: 0.7721 - val_loss: 0.4604 - val_accuracy: 0.8303
Epoch 2/12
79/79 [==============================] - 93s 1s/step - loss: 0.3578 - accuracy: 0.8641 - val_loss: 0.4555 - val_accuracy: 0.8375
Epoch 3/12
79/79 [==============================] - 94s 1s/step - loss: 0.2976 - accuracy: 0.8865 - val_loss: 0.4156 - val_accuracy: 0.8455
Epoch 4/12
79/79 [==============================] - 91s 1s/step - loss: 0.2582 - accuracy: 0.9061 - val_loss: 0.4036 - val_accuracy: 0.8599
Epoch 5/12
79/79 [==============================] - 91s 1s/step - loss: 0.2063 - accuracy: 0.9214 - val_loss: 0.3943 - val_accuracy: 0.8711
Epoch 6/12
79/79 [==============================] - 89s 1s/step - loss: 0.2037 - accuracy: 0.9220 - val_loss: 0.3858 - val_accuracy: 0.8543
Epoch 7/12
79/79 [==============================] - 88s 1s/step - loss: 0.1865 - accuracy: 0.9310 - val_loss: 0.3691 - val_accuracy: 0.8599
Epoch 8/12
79/79 [==============================] - 88s 1s/step - loss: 0.1832 - accuracy: 0.9296 - val_loss: 0.3900 - val_accuracy: 0.8591
Epoch 9/12
79/79 [==============================] - 90s 1s/step - loss: 0.1733 - accuracy: 0.9312 - val_loss: 0.3661 - val_accuracy: 0.8687
Epoch 10/12
79/79 [==============================] - 88s 1s/step - loss: 0.1441 - accuracy: 0.9466 - val_loss: 0.3616 - val_accuracy: 0.8671
Epoch 11/12
79/79 [==============================] - 91s 1s/step - loss: 0.1296 - accuracy: 0.9538 - val_loss: 0.3457 - val_accuracy: 0.8735
Epoch 12/12
79/79 [==============================] - 90s 1s/step - loss: 0.1232 - accuracy: 0.9558 - val_loss: 0.3623 - val_accuracy: 0.8767
```

ResNet 50



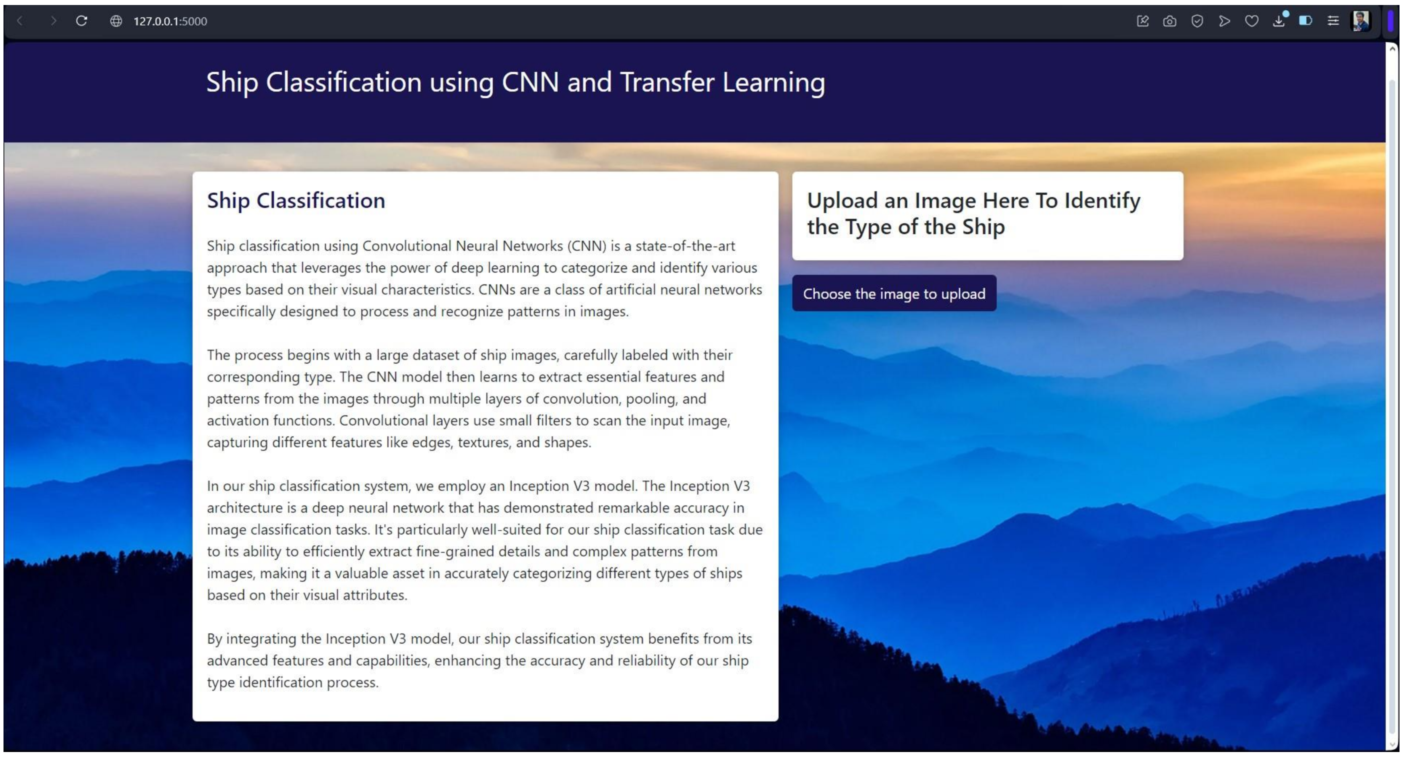
```
history = resnet_model.fit(training_data,steps_per_epoch=len(training_data),epochs=20,validation_data=validation_data,validation_steps=len(validation_data))
```

```
Epoch 1/20
72/72 [==============================] - 29s 335ms/step - loss: 0.6995 - accuracy: 0.7535 - val_loss: 0.5793 - val_accuracy: 0.7904
Epoch 2/20
72/72 [==============================] - 25s 338ms/step - loss: 0.2549 - accuracy: 0.9144 - val_loss: 0.4395 - val_accuracy: 0.8488
Epoch 3/20
72/72 [==============================] - 24s 324ms/step - loss: 0.1391 - accuracy: 0.9652 - val_loss: 0.3891 - val_accuracy: 0.8600
Epoch 4/20
72/72 [==============================] - 25s 343ms/step - loss: 0.0797 - accuracy: 0.9846 - val_loss: 0.3590 - val_accuracy: 0.8688
Epoch 5/20
72/72 [==============================] - 24s 330ms/step - loss: 0.0462 - accuracy: 0.9930 - val_loss: 0.3787 - val_accuracy: 0.8744
Epoch 6/20
72/72 [==============================] - 24s 323ms/step - loss: 0.0331 - accuracy: 0.9982 - val_loss: 0.3854 - val_accuracy: 0.8656
Epoch 7/20
72/72 [==============================] - 24s 327ms/step - loss: 0.0211 - accuracy: 0.9980 - val_loss: 0.3958 - val_accuracy: 0.8680
Epoch 8/20
72/72 [==============================] - 25s 336ms/step - loss: 0.0152 - accuracy: 0.9994 - val_loss: 0.4264 - val_accuracy: 0.8744
Epoch 9/20
72/72 [==============================] - 24s 321ms/step - loss: 0.0117 - accuracy: 0.9992 - val_loss: 0.4633 - val_accuracy: 0.8632
Epoch 10/20
72/72 [==============================] - 24s 335ms/step - loss: 0.0095 - accuracy: 0.9996 - val_loss: 0.4434 - val_accuracy: 0.8720
Epoch 11/20
72/72 [==============================] - 24s 326ms/step - loss: 0.0077 - accuracy: 0.9994 - val_loss: 0.4566 - val_accuracy: 0.8680
Epoch 12/20
72/72 [==============================] - 25s 334ms/step - loss: 0.0119 - accuracy: 0.9982 - val_loss: 0.4826 - val_accuracy: 0.8544
Epoch 13/20
...
Epoch 19/20
72/72 [==============================] - 24s 326ms/step - loss: 0.0051 - accuracy: 0.9998 - val_loss: 0.5352 - val_accuracy: 0.8680
Epoch 20/20
72/72 [==============================] - 24s 335ms/step - loss: 0.0032 - accuracy: 1.0000 - val_loss: 0.5363 - val_accuracy: 0.8736
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

With the inception model having the highest validation accuracy we are choosing to pickle the inception model to use in out flask application for the ship classification project.

RESULTS

The home page looks like this.


Ship Classification using CNN and Transfer Learning

Ship Classification

Ship classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various types based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.

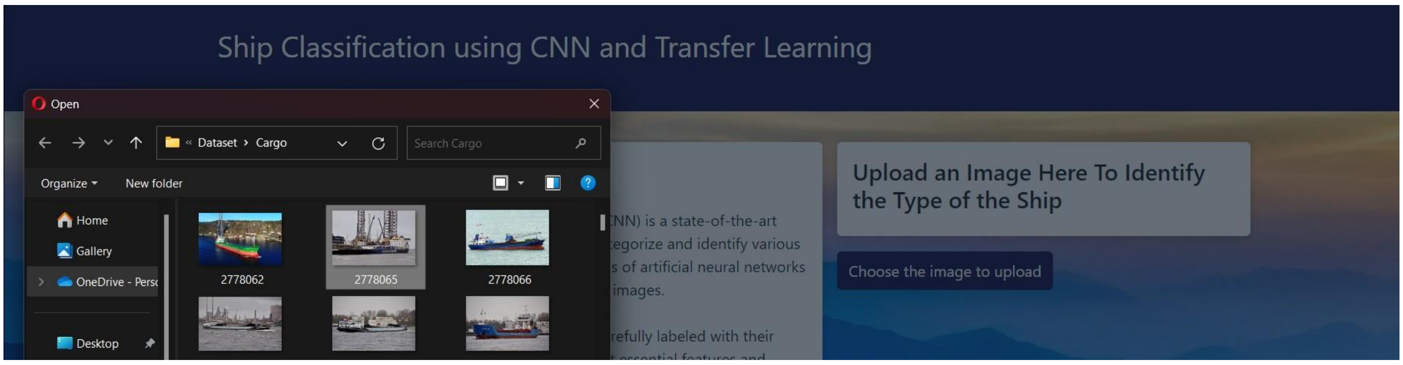
The process begins with a large dataset of ship images, carefully labeled with their corresponding type. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, pooling, and activation functions. Convolutional layers use small filters to scan the input image, capturing different features like edges, textures, and shapes.

In our ship classification system, we employ an Inception V3 model. The Inception V3 architecture is a deep neural network that has demonstrated remarkable accuracy in image classification tasks. It's particularly well-suited for our ship classification task due to its ability to efficiently extract fine-grained details and complex patterns from images, making it a valuable asset in accurately categorizing different types of ships based on their visual attributes.

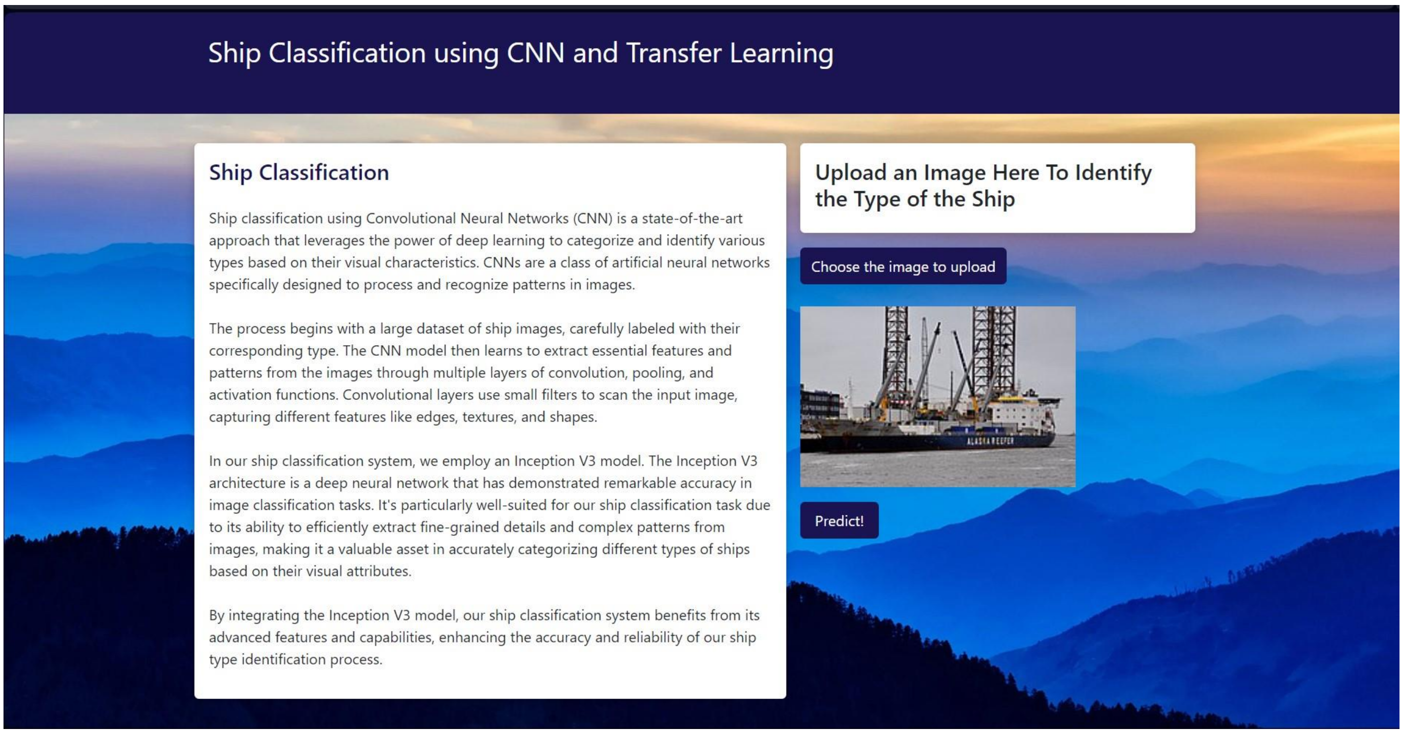
By integrating the Inception V3 model, our ship classification system benefits from its advanced features and capabilities, enhancing the accuracy and reliability of our ship type identification process.

Upload an Image Here To Identify the Type of the Ship

Click on choose the image to upload and select an image to be classified.


Ship Classification using CNN and Transfer Learning
Open
Dataset › Cargo
Search Cargo
Organize
New folder
Home
Gallery
OneDrive - Pers
Desktop
2778062
2778065
2778066
Upload an Image Here To Identify the Type of the Ship
Choose the image to upload

As we can see the selected image is a cargo type of a ship



Ship Classification using CNN and Transfer Learning

Ship Classification

Ship classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various types based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.

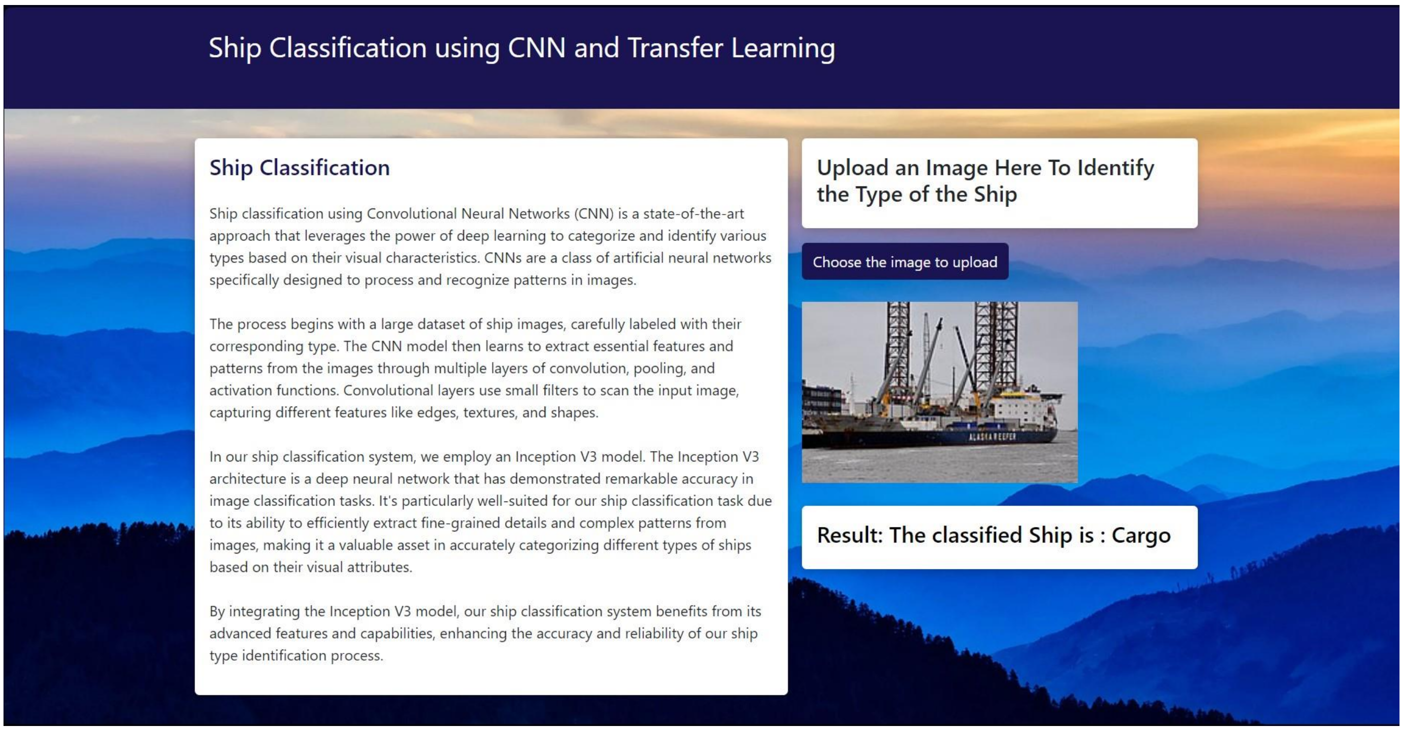
The process begins with a large dataset of ship images, carefully labeled with their corresponding type. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, pooling, and activation functions. Convolutional layers use small filters to scan the input image, capturing different features like edges, textures, and shapes.

In our ship classification system, we employ an Inception V3 model. The Inception V3 architecture is a deep neural network that has demonstrated remarkable accuracy in image classification tasks. It's particularly well-suited for our ship classification task due to its ability to efficiently extract fine-grained details and complex patterns from images, making it a valuable asset in accurately categorizing different types of ships based on their visual attributes.

By integrating the Inception V3 model, our ship classification system benefits from its advanced features and capabilities, enhancing the accuracy and reliability of our ship type identification process.

Upload an Image Here To Identify the Type of the Ship

Now click the on the predict button which appeared.



Ship Classification using CNN and Transfer Learning

Ship Classification

Ship classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various types based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.

The process begins with a large dataset of ship images, carefully labeled with their corresponding type. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, pooling, and activation functions. Convolutional layers use small filters to scan the input image, capturing different features like edges, textures, and shapes.

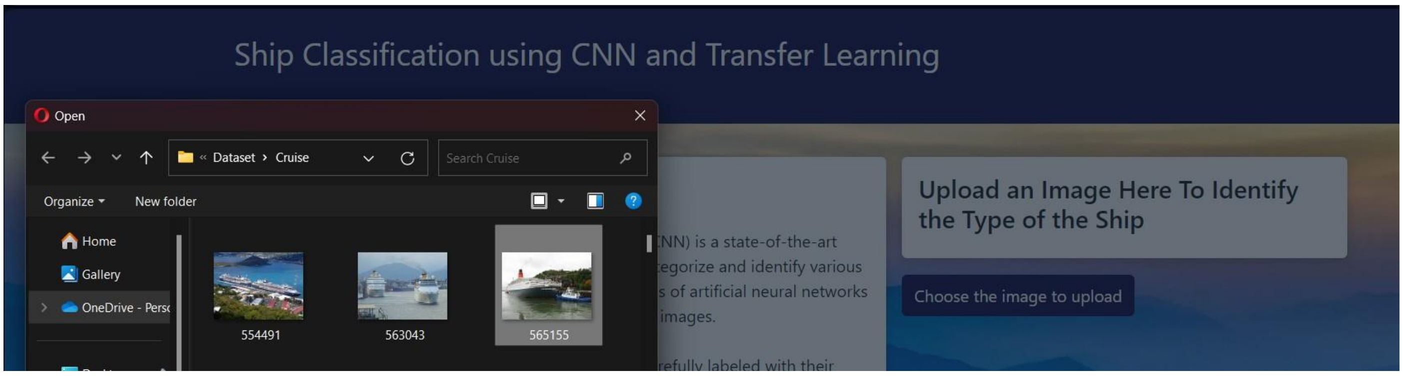
In our ship classification system, we employ an Inception V3 model. The Inception V3 architecture is a deep neural network that has demonstrated remarkable accuracy in image classification tasks. It's particularly well-suited for our ship classification task due to its ability to efficiently extract fine-grained details and complex patterns from images, making it a valuable asset in accurately categorizing different types of ships based on their visual attributes.

By integrating the Inception V3 model, our ship classification system benefits from its advanced features and capabilities, enhancing the accuracy and reliability of our ship type identification process.

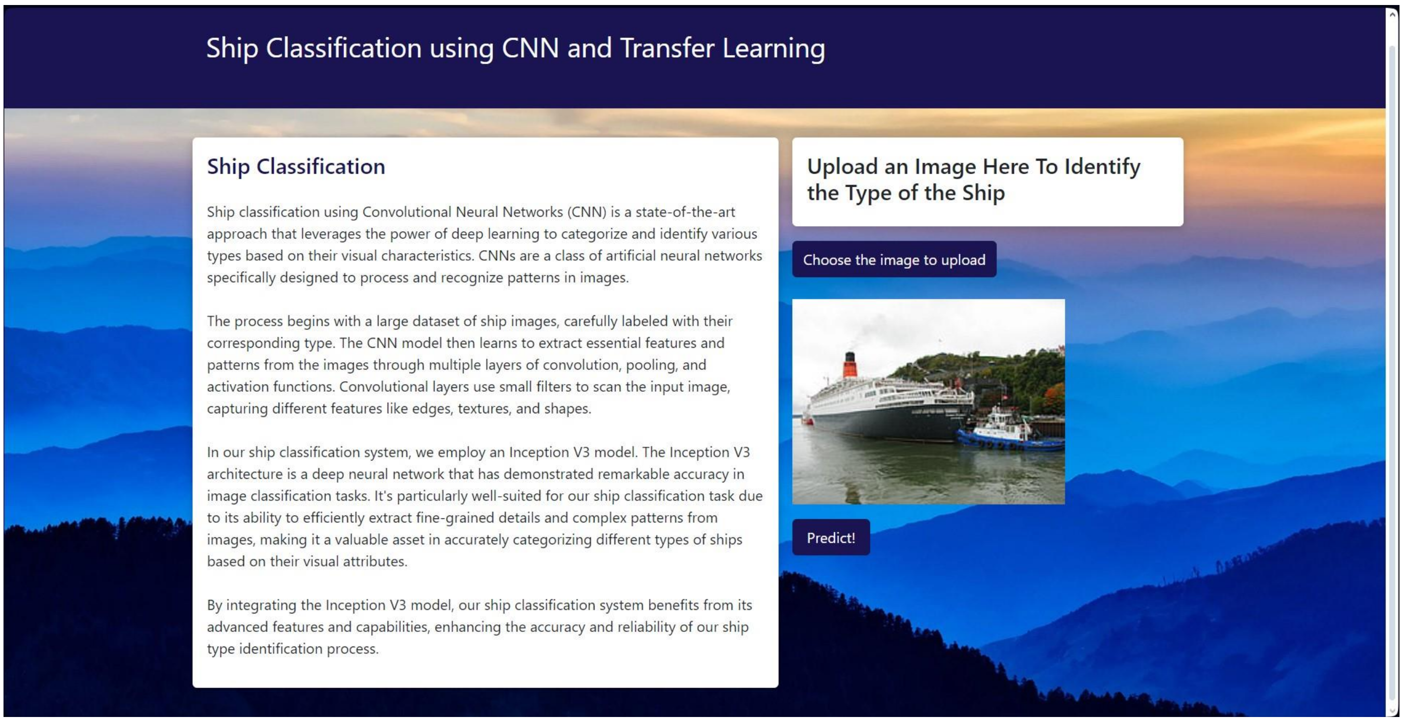
Upload an Image Here To Identify the Type of the Ship

Result: The classified Ship is : Cargo

As we can see the ship has been categorized as a Cargo ship correctly by the flask application running on the inception v3 model.


Ship Classification using CNN and Transfer Learning
Open
Dataset › Cruise
Search Cruise
Organize
New folder
Home
Gallery
OneDrive - Pers
554491
563043
565155
Upload an Image Here To Identify the Type of the Ship
Choose the image to upload

For our second test we are choosing a ship image belonging to the cruise category.


Ship Classification using CNN and Transfer Learning

Ship Classification

Ship classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various types based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.

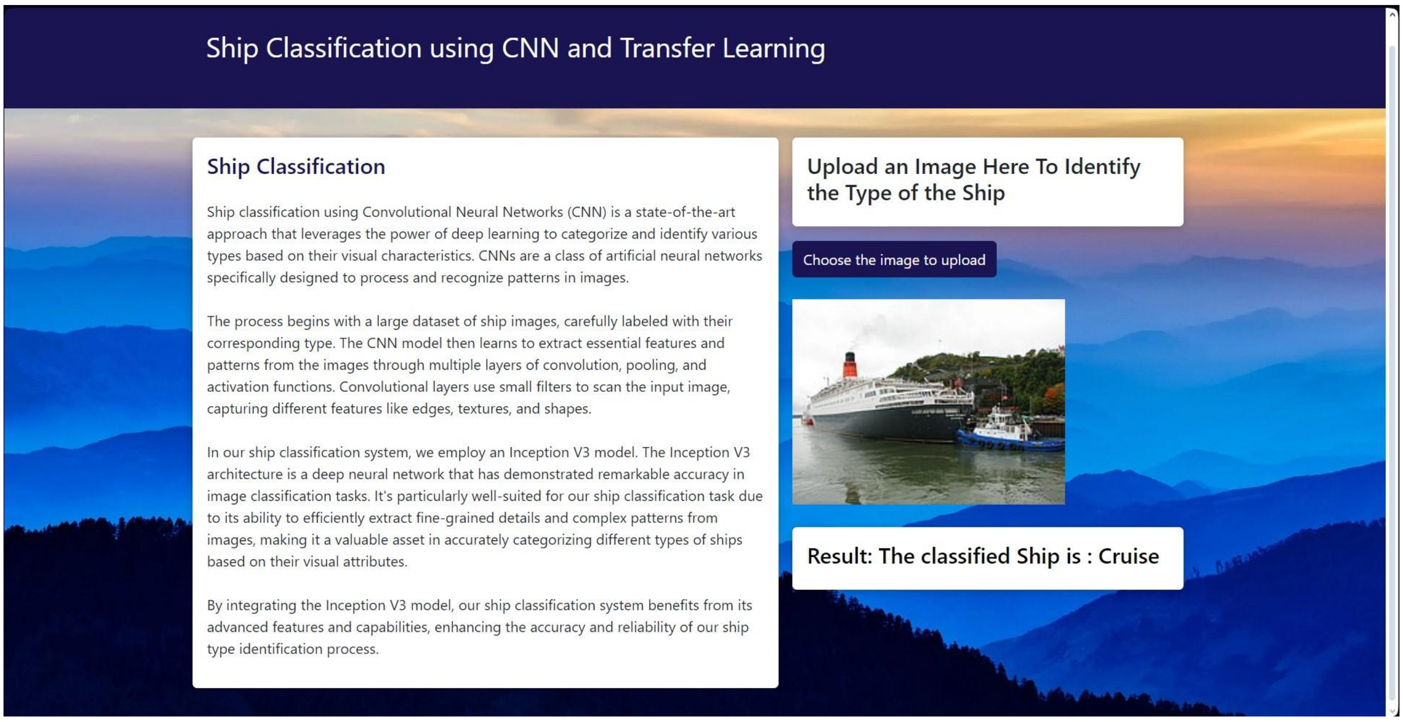
The process begins with a large dataset of ship images, carefully labeled with their corresponding type. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, pooling, and activation functions. Convolutional layers use small filters to scan the input image, capturing different features like edges, textures, and shapes.

In our ship classification system, we employ an Inception V3 model. The Inception V3 architecture is a deep neural network that has demonstrated remarkable accuracy in image classification tasks. It's particularly well-suited for our ship classification task due to its ability to efficiently extract fine-grained details and complex patterns from images, making it a valuable asset in accurately categorizing different types of ships based on their visual attributes.

By integrating the Inception V3 model, our ship classification system benefits from its advanced features and capabilities, enhancing the accuracy and reliability of our ship type identification process.

Upload an Image Here To Identify the Type of the Ship

Click on the predict button when it appears


Ship Classification using CNN and Transfer Learning

Ship Classification

Ship classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various types based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.

The process begins with a large dataset of ship images, carefully labeled with their corresponding type. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, pooling, and activation functions. Convolutional layers use small filters to scan the input image, capturing different features like edges, textures, and shapes.

In our ship classification system, we employ an Inception V3 model. The Inception V3 architecture is a deep neural network that has demonstrated remarkable accuracy in image classification tasks. It's particularly well-suited for our ship classification task due to its ability to efficiently extract fine-grained details and complex patterns from images, making it a valuable asset in accurately categorizing different types of ships based on their visual attributes.

By integrating the Inception V3 model, our ship classification system benefits from its advanced features and capabilities, enhancing the accuracy and reliability of our ship type identification process.

Upload an Image Here To Identify the Type of the Ship

Result: The classified Ship is : Cruise

The ship is correctly classified as a cruise ship

ADVANTAGES & DISADVANTAGES

Advantages

1. Enhanced Maritime Safety: The automated ship classification system can contribute to safer maritime operations by quickly identifying ships and their purposes, aiding in navigation and accident prevention.

2. Improved Security: It can bolster maritime security by assisting in the identification of military vessels and potentially suspicious ships, aiding in border control and security efforts.

3. Efficient Port Operations: Port authorities can optimize cargo handling and storage, reducing turnaround times and improving logistics based on ship type.

4. Environmental Impact Assessment: The system can be used for assessing and regulating the environmental impact of ships, especially concerning emissions and pollution, contributing to a greener maritime industry.

5. Risk Assessment for Insurance: Accurate ship classification helps insurance companies evaluate risks and set appropriate premiums for insuring ships and cargo.

6. Market Analysis: It provides insights into trade dynamics, economic trends, and shipping industry health, aiding in market analysis and investment decisions.

7. Search and Rescue: During maritime emergencies, the system can help identify ships, expediting search and rescue operations.

8. Fisheries and Conservation: It can support fisheries management and marine conservation efforts by tracking and monitoring fishing vessels.

Disadvantages

1. Data Privacy Concerns: Handling ship images may raise privacy concerns, especially if sensitive or personal information is inadvertently captured during classification.

2. Data Quality: The accuracy of ship classification heavily relies on the quality of the dataset. Incomplete or biased data may lead to misclassifications.

3. Maintenance and Updates: Keeping the system up-to-date, including retraining the model and ensuring software security, can be resource-intensive.

4. Initial Development Costs: Building and deploying the system, including training the deep learning model, can require significant initial investments.

5. Complexity: Developing a robust ship classification system using deep learning and integration with other technologies can be complex, requiring specialized skills.

6. Security Risks: Storing and transmitting ship images may present security risks if not properly protected from cyber threats.

7. Resource Intensive: Deep learning models can be computationally intensive, necessitating powerful hardware for efficient processing.

8. Limited Accuracy: Even with deep learning, ship classification may not always achieve 100% accuracy, leading to occasional misclassifications.

9. Regulatory Compliance: Adhering to maritime and data privacy regulations may pose challenges and require ongoing monitoring and adjustments.

Overall, the ship classification project offers numerous benefits, but it also comes with challenges related to data privacy, data quality, ongoing maintenance, and the complexity of deep learning technology. Careful planning and management are crucial to realizing the advantages while mitigating the disadvantages.

CONCLUSION

In conclusion, the ship classification project represents a valuable and innovative endeavor in the maritime industry, with the potential to bring about significant advancements in safety, security, logistics, and environmental sustainability. By automating the process of ship classification using deep learning, specifically the Inception v3 model, the project addresses critical challenges that have long plagued the industry.

The advantages of this project are multifaceted, offering enhanced maritime safety, improved security, streamlined port operations, comprehensive environmental impact assessment, risk assessment for insurance, market analysis insights, and support for search and rescue operations, fisheries management, and marine conservation.

However, it is important to recognize the project's associated disadvantages and challenges. These include concerns about data privacy, data quality, ongoing maintenance, initial development costs, complexity, security risks, computational resources, and the potential for occasional misclassifications.

In light of these factors, successful implementation of the ship classification system hinges on meticulous planning, data quality assurance, robust security measures, and ongoing maintenance to keep the system up-to-date and accurate. Moreover, adherence to regulatory and privacy standards is of paramount importance to ensure ethical and responsible use of data.

The ship classification project embodies the future of maritime operations, where cutting-edge technology contributes to efficiency, safety, and environmental responsibility. By balancing the advantages with the challenges, and through vigilant management, this project holds the promise of positively impacting the maritime industry and the broader global community it serves.

FUTURE SCOPE

The ship classification project holds significant potential for future developments and enhancements, contributing to the ongoing evolution of the maritime industry and related domains. Here are some future scope considerations:

1. Expanded Classification Categories: Beyond cargo carriers, military tankers, and cruise ships, the system can be extended to classify a wider range of ship types, including container ships, fishing vessels, and more, providing greater granularity in ship identification.

2. Enhanced Accuracy: Continuous improvement in deep learning models and data quality can lead to even higher accuracy in ship classification, reducing the margin for misclassifications.

3. Real-time Monitoring: Integration with real-time data sources, such as AIS (Automatic Identification System) data, satellite imagery, and radar, can enable live ship tracking and classification, improving maritime situational awareness.

4. Global Implementation: Expanding the system for global use, with multilingual support and adaptation to diverse maritime regulations, would make it a valuable tool for international maritime operations.

5. Environmental Impact Assessment: Advanced image analysis and machine learning algorithms can be employed to assess environmental factors, such as emissions, fuel efficiency, and adherence to environmental regulations, thereby contributing to sustainable shipping practices.

6. Automated Alerts and Notifications: Implementing alert systems based on ship classification results can facilitate immediate responses to potential security threats, accidents, or environmental incidents.

7. Integration with Autonomous Shipping: As autonomous shipping technologies continue to develop; ship classification systems can play a crucial role in identifying and monitoring autonomous vessels for safety and compliance.

8. Customization for Specific Use Cases: Tailoring the system for specific maritime sectors, such as oil and gas, fisheries, or cruise tourism, can enhance its relevance and utility in these industries.

9. Collaboration with Maritime Authorities: Close collaboration with maritime authorities and organizations can lead to standardized practices and regulatory compliance in ship classification, ensuring widespread adoption and adherence to industry standards.

10. Data Sharing and Transparency: Promoting data sharing and transparency in ship classification results can foster trust and cooperation among maritime stakeholders, improving overall industry efficiency.

The future scope of the ship classification project is promising, with opportunities for continuous advancement, innovation, and broader application, ultimately shaping a safer, more secure, and environmentally responsible maritime industry.

Source Code

The drive link for the ipynb file with the ship classification model

https://drive.google.com/file/d/11uKa-gGdJnd-3nfk4bwF0YvAJD0VZuVQ/view?usp=drive_link

The github link for the ipynb file

https://github.com/smartinternz02/SI-GuidedProject-600206-1697472221/blob/main/4.%20Development%20Phase/Ship%20Classification.ipynb

The github link for the flask source code

https://github.com/smartinternz02/SI-GuidedProject-600206-1697472221/tree/main/4.%20Development%20Phase

The demo video drive link

https://drive.google.com/file/d/13qIC8bbzERUc-LigA7946mGLdtcH0RcF/view?usp=drive_link

The demo video github link

https://github.com/smartinternz02/SI-GuidedProject-600206-1697472221/blob/main/4.%20Development%20Phase/Demo%20Video.mp4