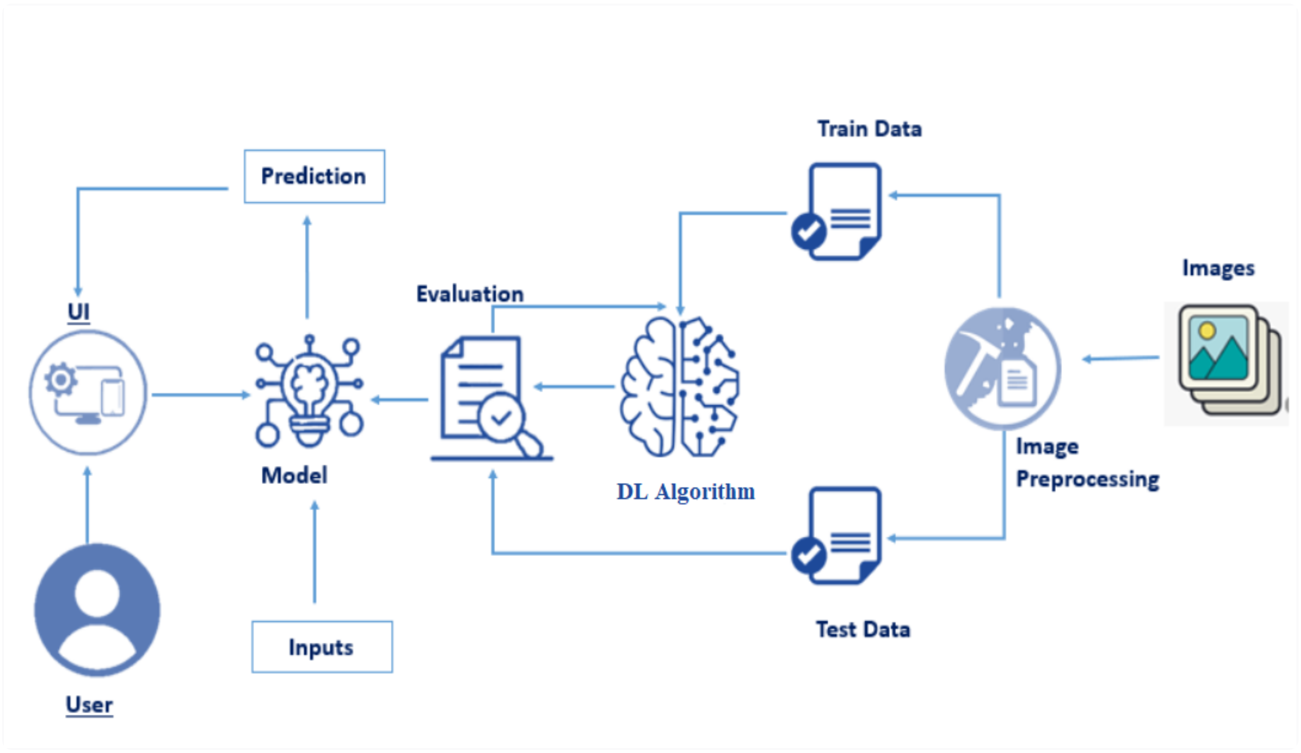
Ship classification using IBM Watson

Ship Classification Using IBM Watson



Submitted by:

Raghumanda.Naga Venkata Siva Prasad
Pin no:19481A03A4

Ship classification using IBM Watson

ProjectDescription :

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 5 categories of ships - Cargo, Carrier, Military, Cruise and Tankers. All the images are present in a single folder so we will first be dividing them into categories with the help of a train.csv file that contains all the filenames of the training images. The model used to train the model is VGG16, 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 frame

Pre Requisites:

1. In order to develop this project we need to install the following software/packages:

Anaconda Navigator :

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning-related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using a Jupyter notebook and Spyder To install the Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

https://youtu.be/5mDYijMfSzs

Python packages:

NumPy: NumPy is a Python package that stands for 'Numerical Python. It is the core library for scientific computing, which contains a powerful n-dimensional array of objects.

Pandas: pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, Ship Classification Using IBM Watson built on top of the Python programming language.

Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

Keras: Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, R, Theano, and PlaidML.

TensorFlow: TensorFlow is just one part of a much bigger, and growing ecosystem of libraries and extensions that help you accomplish your machine learning goals. It is a free and open-source software library for data flow and differentiable programming across a range of tasks.

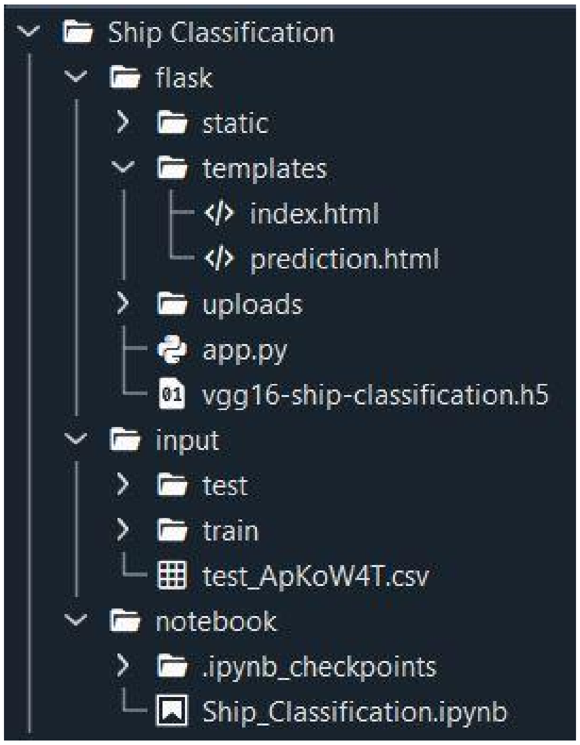
Flask: Web framework used for building Web applications

If you are using anaconda navigator, follow the below steps to download the required packages:

? Open anaconda prompt.
? Type “pip install OpenCV-python” and click enter.
? Type “pip install keras==2.2.4” and click enter.
? Type “pip install tensorflow==2.0” and click enter.(Make sure you are work on python 64 bit)
? Type “pip install Flask” and click enter.

Project Structure:

Ship classification using IBM Watson



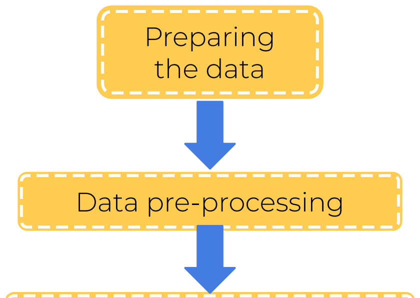
```
Ship Classification
  flask
    static
    templates
      index.html
      prediction.html
    uploads
    app.py
    vgg16-ship-classification.h5
  input
    test
    train
    test_ApKoW4T.csv
  notebook
    .ipynb_checkpoints
    Ship_Classification.ipynb
```

● The input folder contains two folders for train and test images, each of them having images of different categories of ships, arranged folder-wise. Ship Classification Using IBM Watson
● Flask folder has all the files necessary to build the flask application.
● static folder has the images, style sheets and scripts that are needed in building the web page.
● templates folder has the HTML pages.
● uploads folder has the uploads made by the user.
● app.py is the python script for server side computing.
● .h5 file is the model file which is saved after model building.
● Ship_Classification.ipynb is the notebook on which code is executed.

Project Objectives:

By the end of this project you'll understand:
● Preprocess the images.
● Training VGG16 model with custom data.
● How pre-trained models will be useful in object classification.
● How to evaluate the model.
● Building a web application using the Flask framework.

Project Flow:



Ship classification using IBM Watson

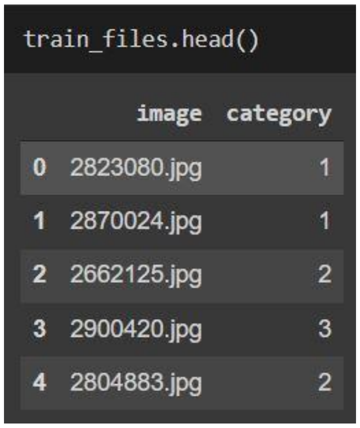
Preparing the data IBM Watson Data pre-processing Building the model Building Flask application Training and testing the model Ship Classification Using IBM Watson

Preparing The Data:

I downloaded the dataset from the following link.

https://www.kaggle.com/code/abdullahhaxsh/ship-classifier-using-cnn/data

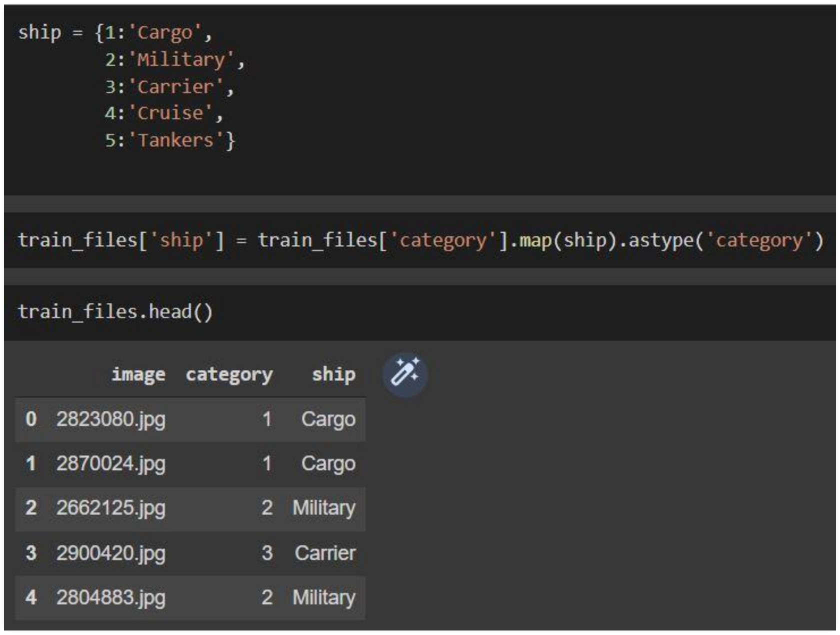
Categorize Train Images:



```
train_files.head()
```

	image	category
0	2823080.jpg	1
1	2870024.jpg	1
2	2662125.jpg	2
3	2900420.jpg	3
4	2804883.jpg	2

Ship classification using IBM Watson

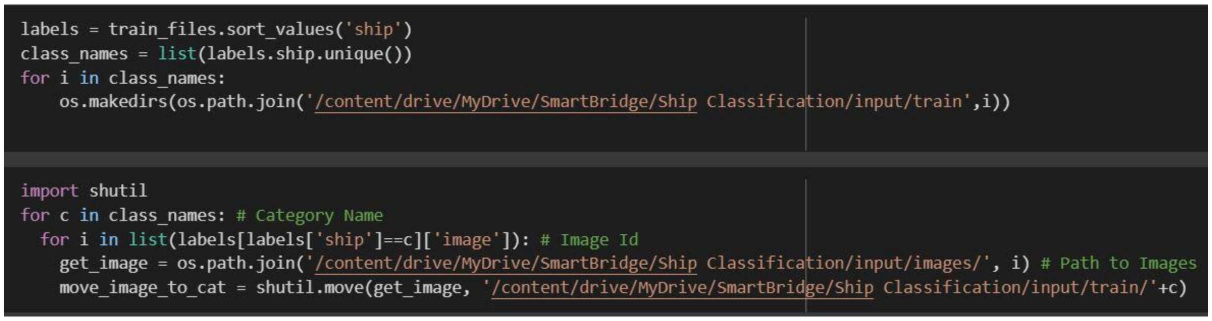


```
ship = {1:'Cargo',
        2:'Military',
        3:'Carrier',
        4:'Cruise',
        5:'Tankers'}
```

```
train_files['ship'] = train_files['category'].map(ship).astype('category')
```

```
train_files.head()
```

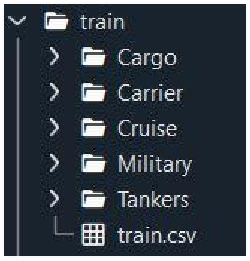
	image	category	ship
0	2823080.jpg	1	Cargo
1	2870024.jpg	1	Cargo
2	2662125.jpg	2	Military
3	2900420.jpg	3	Carrier
4	2804883.jpg	2	Military



```
labels = train_files.sort_values('ship')
class_names = list(labels.ship.unique())
for i in class_names:
    os.makedirs(os.path.join('/content/drive/MyDrive/SmartBridge/Ship Classification/input/train',i))
```

```
import shutil
for c in class_names: # Category Name
  for i in list(labels[labels['ship']==c]['image']): # Image Id
    get_image = os.path.join('/content/drive/MyDrive/SmartBridge/Ship Classification/input/images/', i) # Path to Images
    move_image_to_cat = shutil.move(get_image, '/content/drive/MyDrive/SmartBridge/Ship Classification/input/train/'+c)
```

The above code will create sub-directories within the train folder and move the images into them. The final folder structure of train will look like:

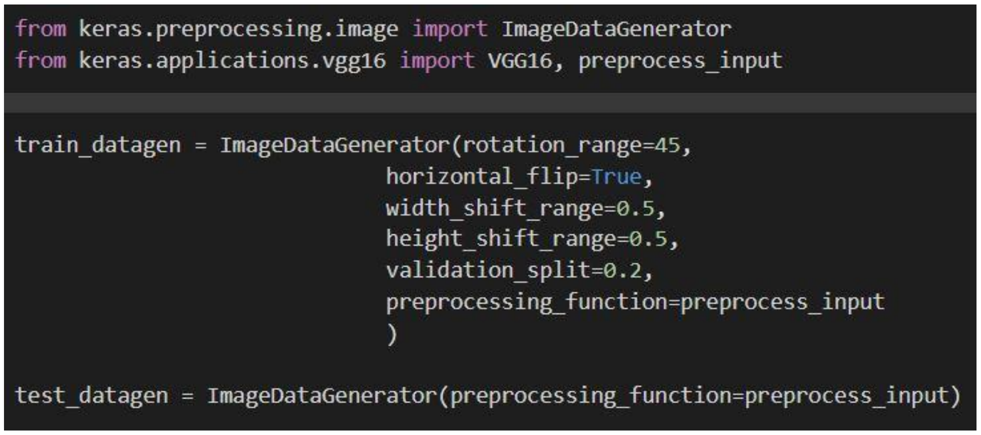


```
train
  Cargo
  Carrier
  Cruise
  Military
  Tankers
  train.csv
```

Data Pre-Processing:

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

Ship classification using IBM Watson



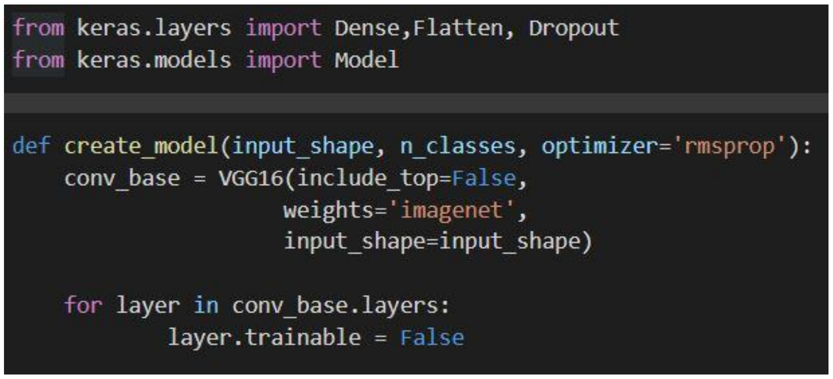
```python
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16, preprocess_input
```

```python
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)
```

• Specify the path of both the folders in flow_from_directory method. We are importing the images in 224*224 pixels.

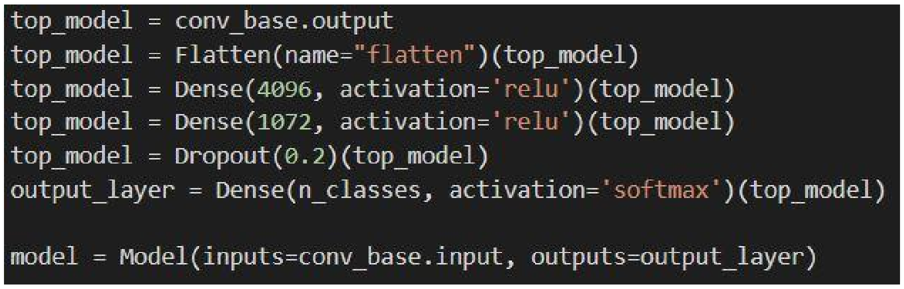
Building The Model:

• First of all we have to import libraries and define a function for the model.

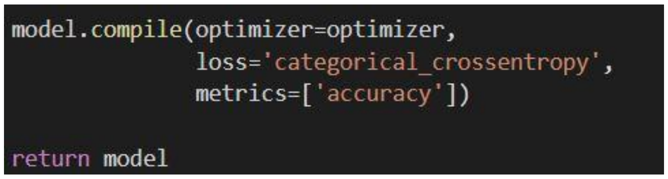


```python
from keras.layers import Dense,Flatten, Dropout
from keras.models import Model
```

```python
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
```



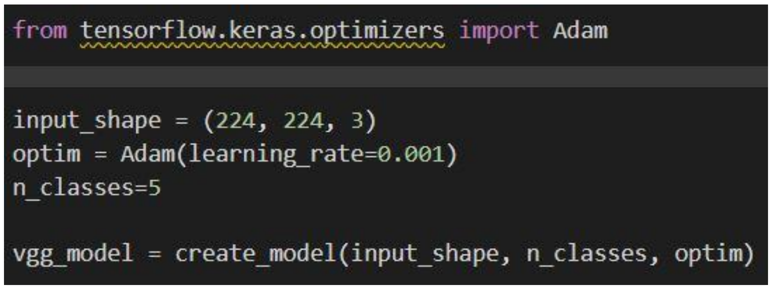
```python
top_model = conv_base.output
top_model = Flatten(name="flatten")(top_model)
top_model = Dense(4096, activation='relu')(top_model)
top_model = Dense(1072, activation='relu')(top_model)
top_model = Dropout(0.2)(top_model)
output_layer = Dense(n_classes, activation='softmax')(top_model)

model = Model(inputs=conv_base.input, outputs=output_layer)
```



```python
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

return model
```

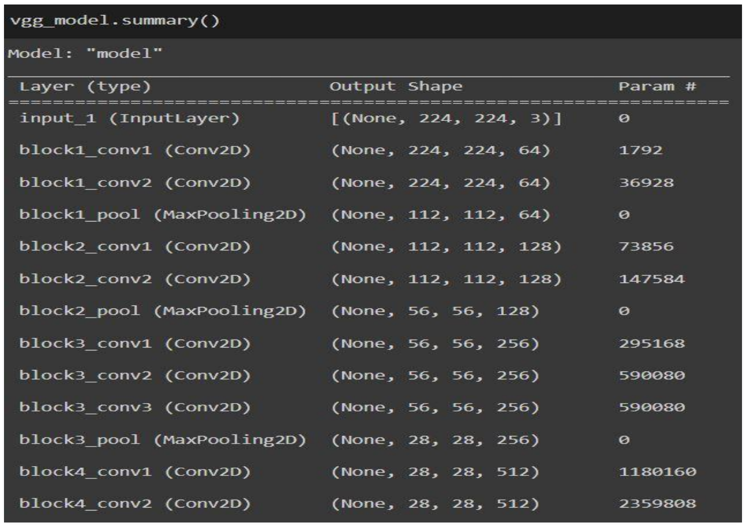
Ship classification using IBM Watson

• Call the summary() function to have a look at the model summary:

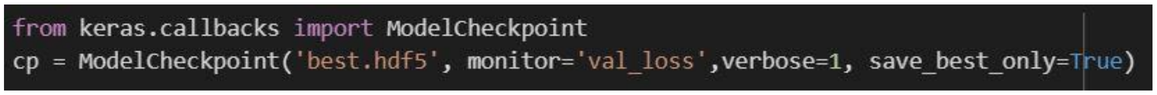


```
from tensorflow.keras.optimizers import Adam

input_shape = (224, 224, 3)
optim = Adam(learning_rate=0.001)
n_classes=5

vgg_model = create_model(input_shape, n_classes, optim)
```

• We need to create a checkpoint to track the validation loss of the model in order to save the best weights.



```
vgg_model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
```



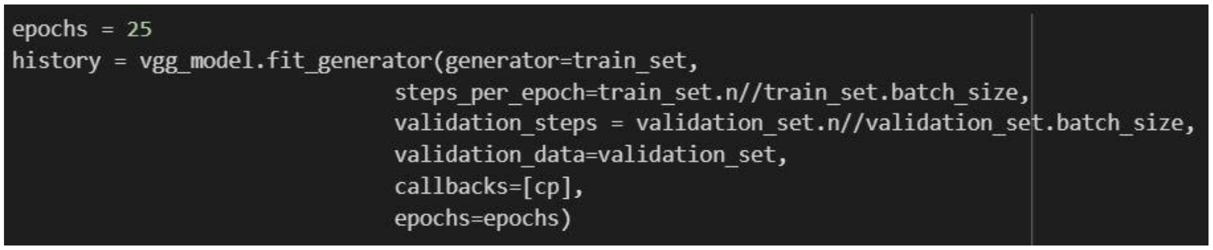
```
from keras.callbacks import ModelCheckpoint
cp = ModelCheckpoint('best.hdf5', monitor='val_loss',verbose=1, save_best_only=True)
```

r

Ship classification using IBM Watson

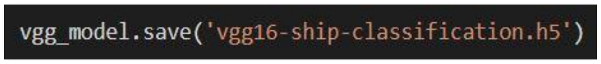
Training And Testing The Model:

• To train the model while monitoring validation loss.We will be training the model for 25 epochs using the fit_generator() function and save the model:



```
epochs = 25
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=epochs)
```

• We have to test the model with the custom inputs.First, specify the path of the image to be tested. Then, preprocess the image and perform predictions.



```
vgg_model.save('vgg16-ship-classification.h5')
```

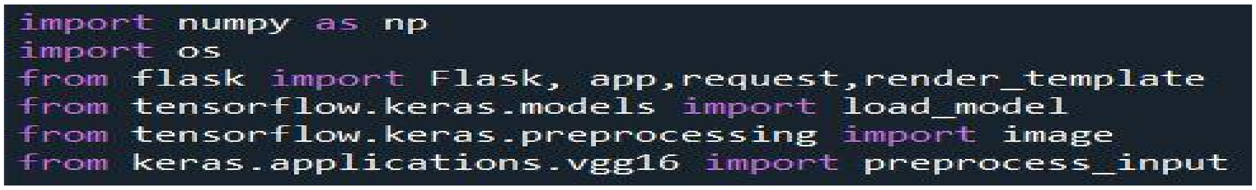
Building Flask Application:

● After the model is built, we will be integrating it to a web application so that normal users can also use it. The new users need to initially register in the portal. After registration users can login to browse the images to detect the category of ships.

Build A Python Application:

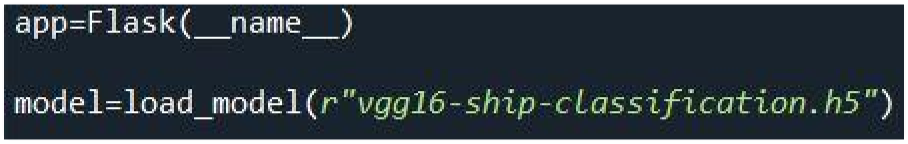
For creating a python application,we have to follow the following steps:

Step 1: Load the required packages.



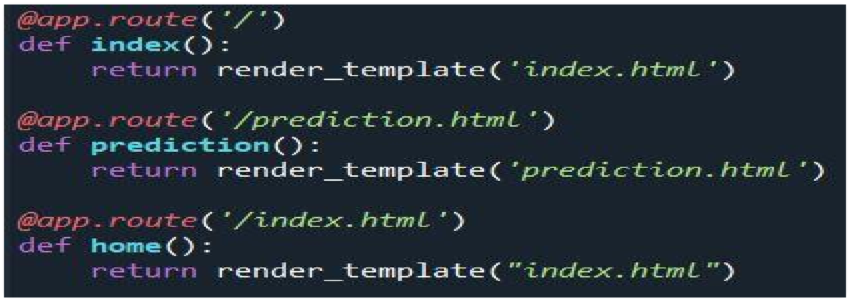
```
import numpy as np
import os
from flask import Flask, app,request,render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
```

Step 2: Initialize the flask app and load the model.



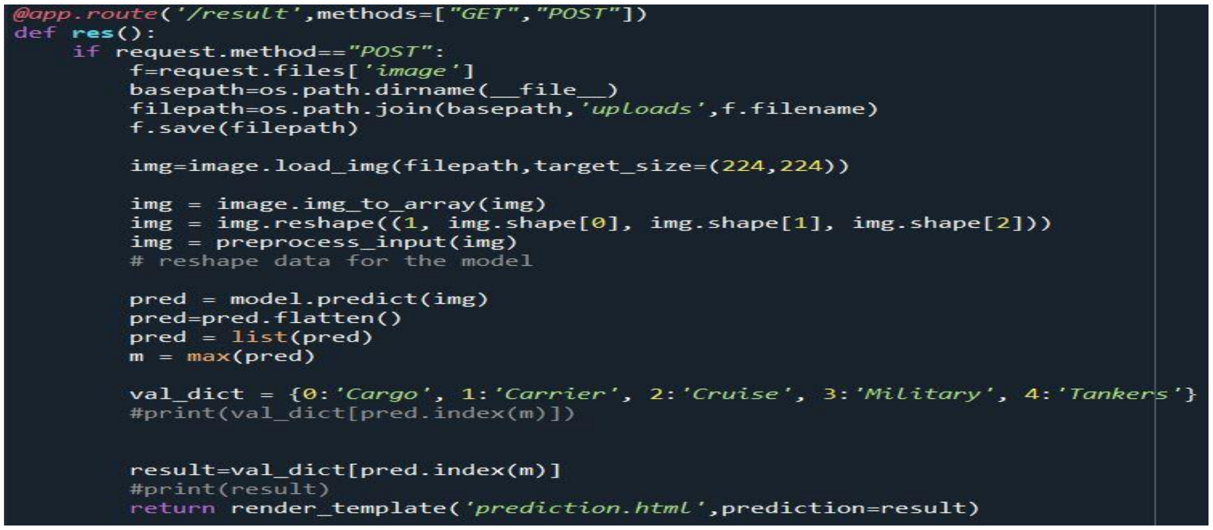
```
app=Flask(__name__)

model=load_model(r"vgg16-ship-classification.h5")
```

Step 3: Configure html pages



```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/prediction.html')
def prediction():
    return render_template('prediction.html')

@app.route('/index.html')
def home():
    return render_template("index.html")
```

Step 4: Pre-process the frame and run



```
@app.route('/result',methods=["GET","POST"])
def res():
    if request.method=="POST":
        f=request.files['image']
        basepath=os.path.dirname(__file__)
        filepath=os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)

        img=image.load_img(filepath,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)
        # reshape data for the model

        pred = model.predict(img)
        pred=pred.flatten()
        pred = list(pred)
        m = max(pred)

        val_dict = {0:'Cargo', 1:'Carrier', 2:'Cruise', 3:'Military', 4:'Tankers'}
        #print(val_dict[pred.index(m)])


        result=val_dict[pred.index(m)]
        #print(result)
        return render_template('prediction.html',prediction=result)
```

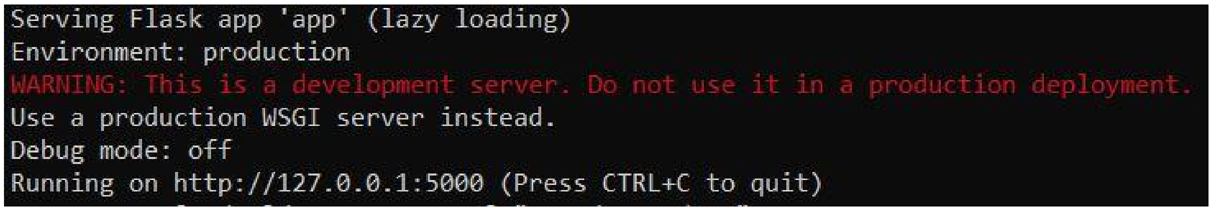
Then we have to run the flask application using the run method. By default the flask runs on port 5000. If the port is to be changed, an argument can be passed and the port can be modified

. Build The HTML Page And Execute:

To build the HTML page and execute.we have to follow the below steps:

Step 1: Run the application

Ship classification using IBM Watson

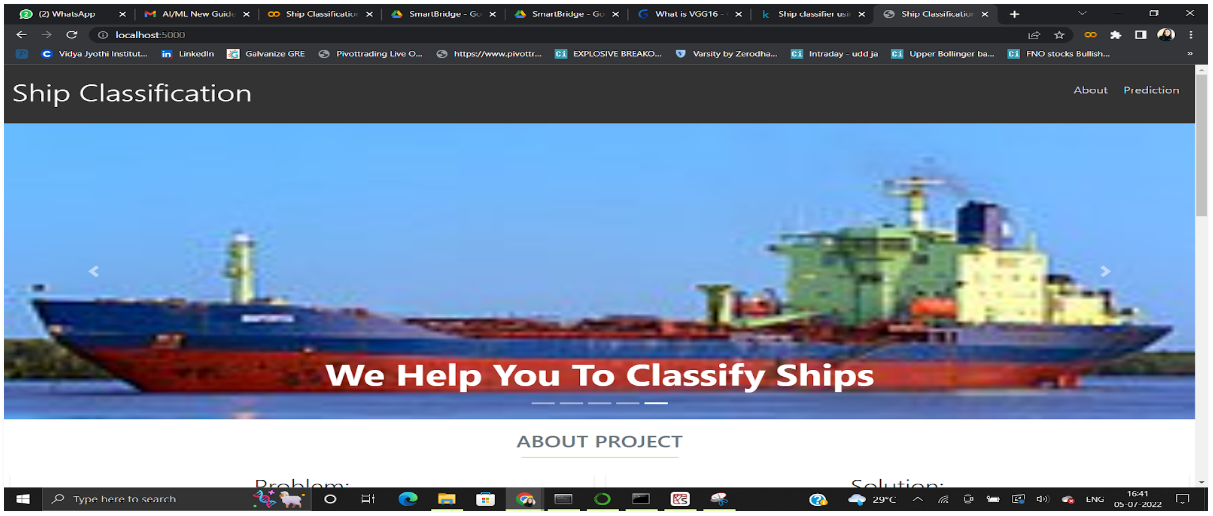


```
Serving Flask app 'app' (lazy loading)
Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
Debug mode: off
Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

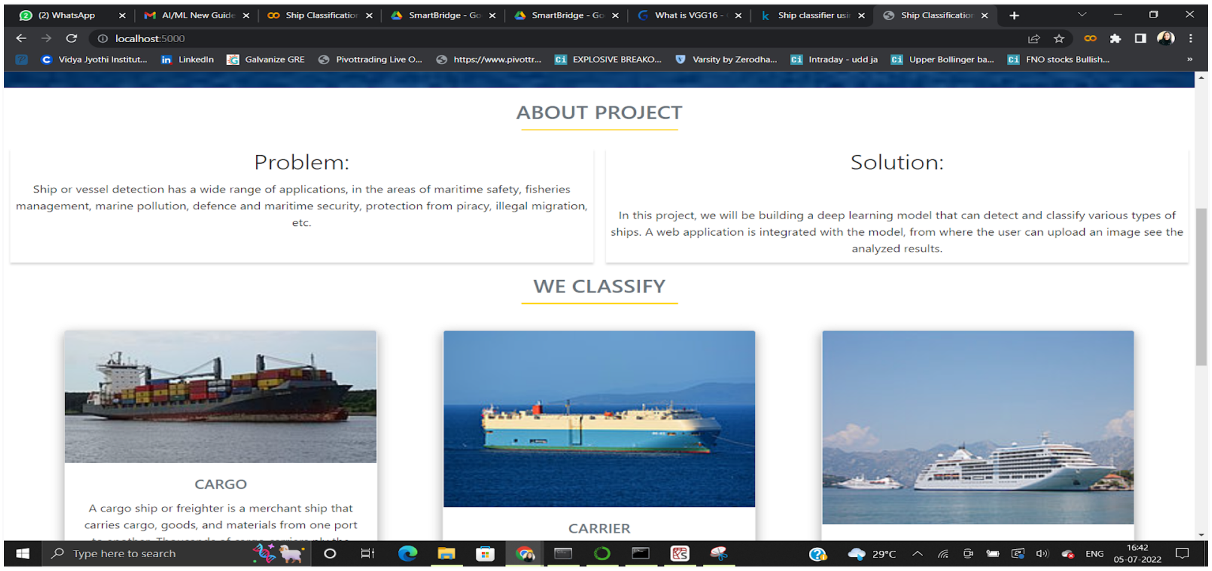
In the anaconda prompt, navigate to the folder in which the flask app is present. When the python file is executed the localhost is activated on port 5000 and can be accessed through it.

Step 2: Open the browser and navigate to localhost:5000 to check your application

The home page looks like this. You can click on login or register.



Ship classification using IBM Watson



ABOUT PROJECT

Problem:

Ship or vessel detection has a wide range of applications, in the areas of maritime safety, fisheries management, marine pollution, defence and maritime security, protection from piracy, illegal migration, etc.

Solution:

In this project, we will be building a deep learning model that can detect and classify various types of ships. A web application is integrated with the model, from where the user can upload an image see the analyzed results.

WE CLASSIFY

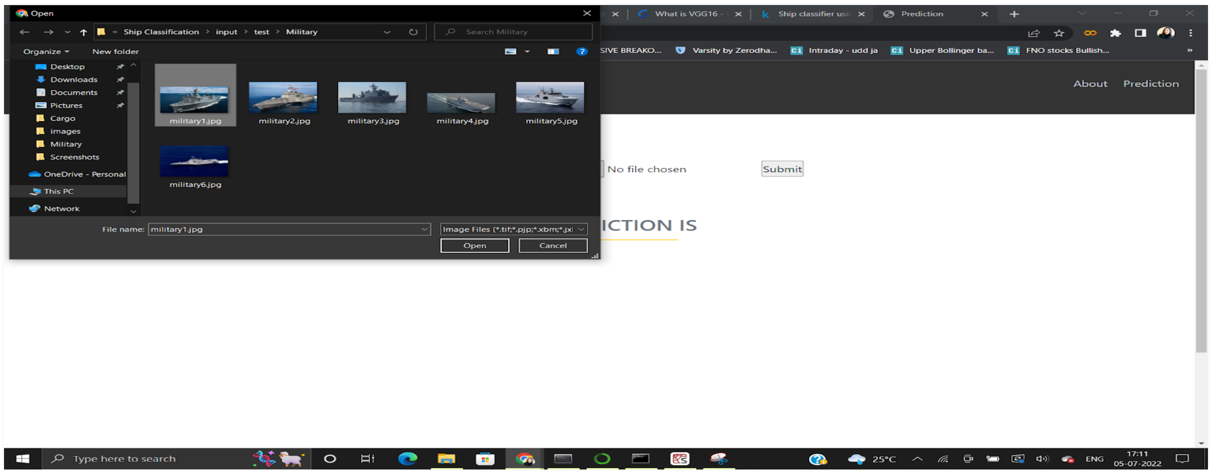
CARGO

A cargo ship or freighter is a merchant ship that carries cargo, goods, and materials from one port

CARRIER

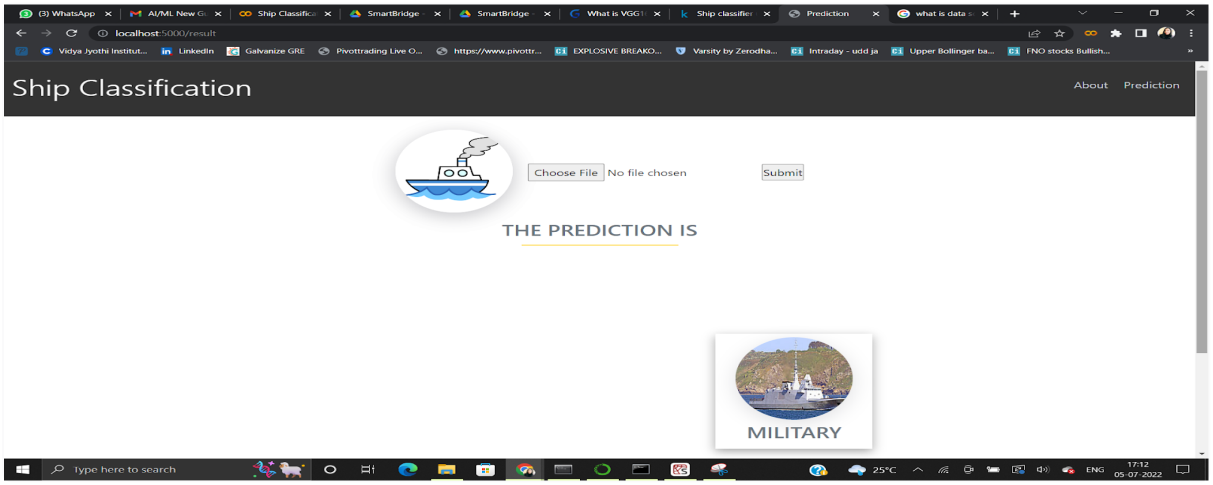
After clicking on the predict button you will be redirected to the prediction page where you can browse the images

Input 1:



Output 1:

Ship classification using IBM Watson



IBM Watson:

For Account Creation,follow the given link :

https://youtu.be/QuTDhYeJh0k

Training model on IBM watson

https://youtu.be/BzouqMGJ41k

I have done local deployement from the jupiter note book.And modify the code In IBM watson studio.I got scoring end point and python code by modifing the code in IBM watson studio.After that I have converted the app from local deployment to public deployment.Then everyone will use it to classify the ships.

CONCLUSION:

This paper introduced a new classification model's architecture, which is based on improving the ResNet152 architecture. This improved the performance of the classification model for classifying ships. Initially, the pretrained models used were AlexNet, VGG16, ResNet, Inception V3 and GoogleNet for the game of deep learning sea ship public dataset, which consisted of five classes: cargo, military, carrier, cruise and tanker ships. Based on their performance, the best model was selected for further improvement. Additionally, for proper image pre-processing, a comparison of accuracy for noisy and low-contrast images will be used along with the addition of the Jaccard index to compare the accuracy of the classification.