1. INTRODUCTION

1.Overview

In this project, we introduce a new convolutional neural network classification algorithm capable of classifying five classes of ships, including cargo, military, carrier, cruise and tanker ships, in inland waterways. 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 framework The purpose of ship classification is to identify different categories of ship images with as accurately as possible. Recent years, lots of researchers have made great efforts for this purpose . The visual sensor is usually significantly affected by the natural factors, such as weather and lighting conditions and so on, which makes it difficult to recognize the ship types.

What's more, wide within-class variation in some types of vessels also makes ship classification more complicated and challenging . Up to now, numerous feature extraction algorithms have been proposed for scene classification, including entropy-based hierarchical discriminant regression, completed local binary patterns (CLBP), multi-linears principal component analysis , hierarchical multi-scales local binary pattern (HMLBP) , histogram of oriented gradients, and multiple feature learning . CLBP is a local texture feature descriptor which captures structure information and extracts local features, e.g., edge feature. Compared with typical LBP, relationship between center point and adjacent points is better described, while the robustness of illumination changes and noise is enhanced. MFL is operated to obtain comprehensive representation of features via fusing the local features and global features. Based on previous work , Rainey and Stastny developed several effective recognition algorithms to separate different ship categories.

2.Purpose

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.

2. LITERATURE SURVEY

2.1 Existing Problem

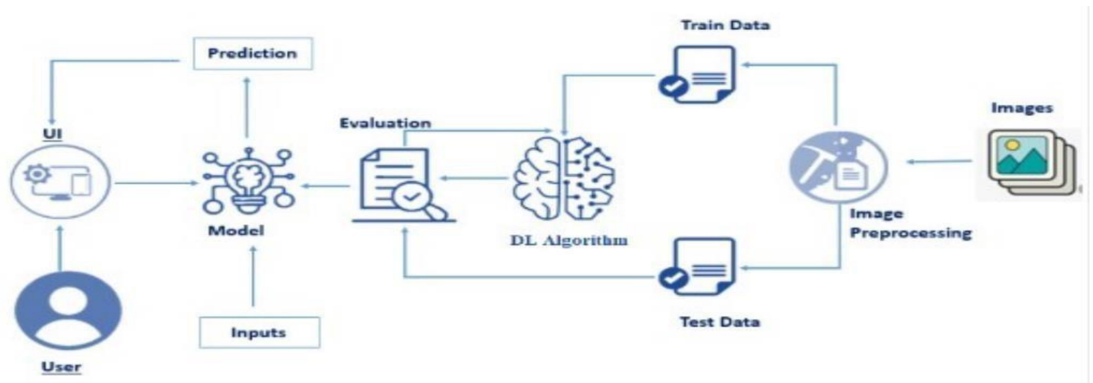
In recent years, deep learning has been used in various applications including the classification of ship targets in inland waterways for enhancing intelligent transport systems. Various researchers introduced different classification algorithms, but they still face the problems of low accuracy and misclassification of other target objects. Hence, there is still a need to do more research on solving the above problems to prevent collisions in inland waterways.

2.2 Proposed Solution

In order to solve the problems for the accuracy of the classification system, we proposed a new classification model. First, based on the pretrained models, the models were fine-tuned with the public dataset we used. Based on their performance, the best model was selected in order to further adjust the performance for high accuracy in classifying ships in inland river waterways. After selecting the best model, the model was adjusted, and classification was conducted based on the modification of the network.

3. THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software Designing

- Anaconda Navigator: Anaconda Navigator is a free and opensource distribution of the Python and R programming languages for data science and machine learningrelated applications. It can be installed on Windows, Linux, and macOS.Conda is an opensource, 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.

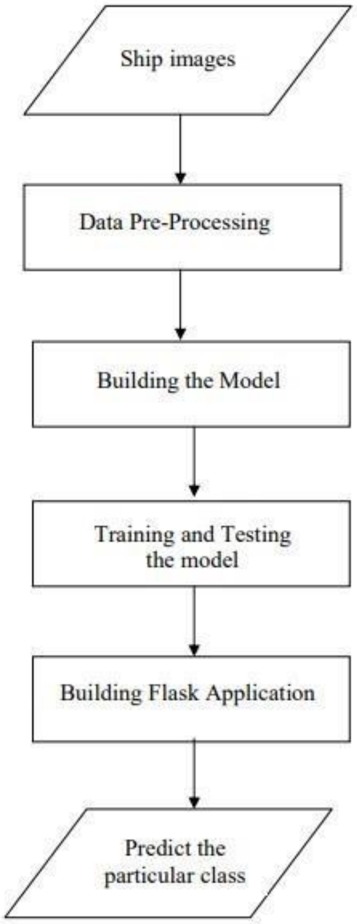
- Python packages:
- NumPy: NumPy is a Python package that stands for 'Numerical Python. It is the core library for scientific computing, which contains a powerful n-dimensional array of objects. 4
- Pandas: pandas is a fast, powerful, flexible, and easy-to-use opensource data analysis and manipulation tool, 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. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
- 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 opensource software library for data flow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks.
- Flask: Web framework used for building Web applications
- Hardware
- Device name: DESKTOP-J5SC39S
- Processor: Intel(R) Pentium(R) Gold G5420 CPU @ 3.80GHz 3.79 GHz System type: 64-bit operating system, x64-based processes.

4. EXPERIMENTAL INVESTIGATIONS

The images need to be organized before proceeding with the project. The original dataset has a single folder known as images. We will be using the train.csv file to fetch the image ID’s of training images. Then we are creating subdirectories with in train folder and move images to them. The dataset images are to be pre-processed before giving to the model. We will create a function that uses the pretrained VGG16 model for predicting custom classes.

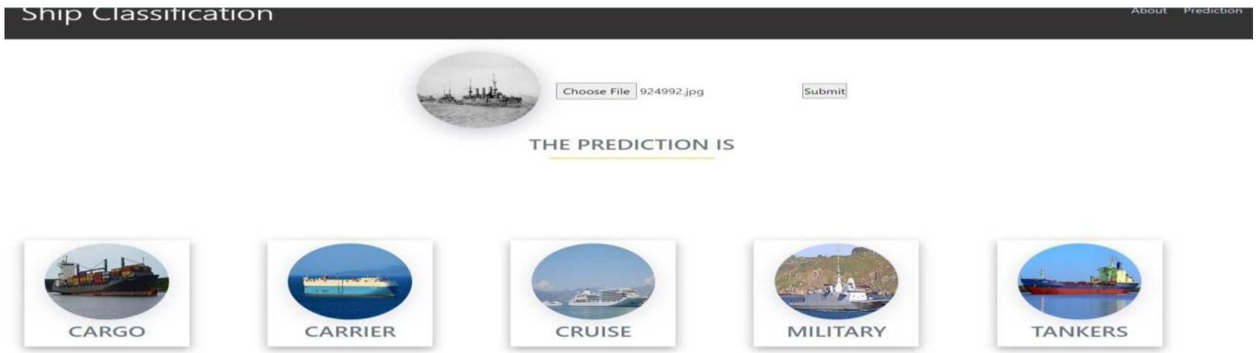
Then we have to test and train the model. After the model is build, we will be integrating it to a web application.

5. FLOWCHARTS

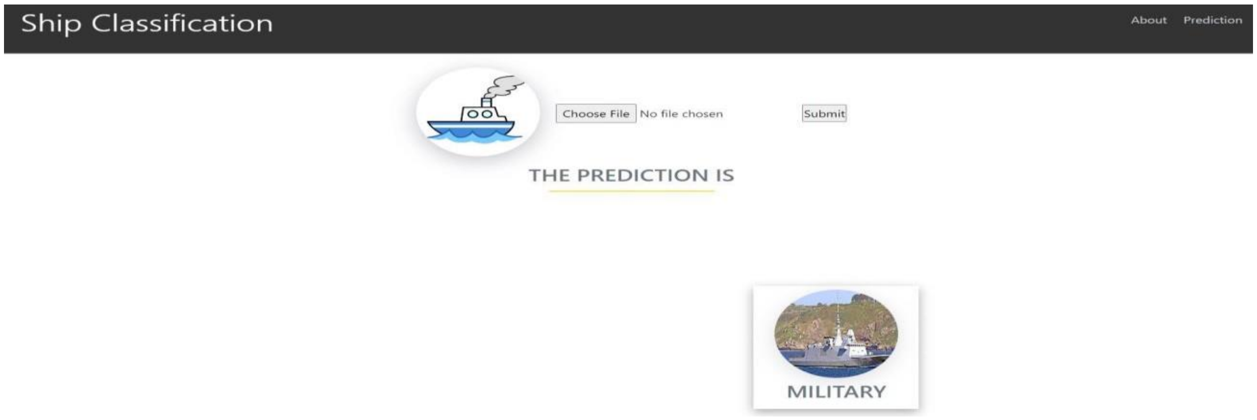

Ship images
Data Pre-Processing
Building the Model
Training and Testing the model
Building Flask Application
Predict the particular class

6. RESULT

INPUT



OUTPUT



7. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Easy to use
- Time efficient
- Cost efficient

DISADVANTAGES:

- Time consuming
- Not so accurate results for small ships.

8. APPLICATIONS

A **ship classification society** or **ship classification organisation** is a non-governmental organization that establishes and maintains technical standards for the construction and operation of ships and offshore structures. Classification societies certify that the construction of a vessel comply with relevant standards and carry out regular surveys in service to ensure continuing compliance with the standards. Currently, more than 50 organizations describe their activities as including marine classification, twelve of which are members of the International Association of Classification Societies.

A **classification certificate** issued by a classification society recognised by the proposed ship register is required for a ship's owner to be able to register the ship and to obtain marine insurance on the ship, and may be required to be produced before a ship's entry into some ports or waterways, and may be of interest to charterers and potential buyers. To avoid liability, classification societies explicitly disclaim responsibility for the safety, fitness for purpose, or seaworthiness of the ship, but is a verification only that the vessel is in compliance with the classification standards of the society issuing the classification certificate.

9. CONCLUSION

This project was about classifying the ship. This project improved the performance of the classification model for classifying ships in inland waterways. The new proposed method achieved high accuracy compared with the other existing algorithms. It was compared with other existing algorithms in classifying different classes of ships in inland waterways, and our proposed method achieved better results compared with the others. 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.

10. FUTURE SCOPE

In future works, the proposed method will be improved in order to classify the ships in different weather conditions using more advanced technology.

11. BIBLIOGRAPHY

- https://www.mdpi.com/2078-2489/12/8/302/htm
- http://cs229.stanford.edu/proj2017/final-reports/5244159.pdf
- https://www.kaggle.com/code/teeyee314/classification-of-ship-images/log

APPENDIX

App.py

```
import re import numpy as np import os from flask import
Flask, app,request,render_template from tensorflow.keras
import models from tensorflow.keras.models import
load_model from tensorflow.keras.preprocessing import
image from tensorflow.python.ops.gen_array_ops import
concat
from keras.applications.vgg16 import preprocess_input
 #Loading the model  model=load_model(r"vgg16-ship-classification.h5")
 app=Flask(__name__)
#default home page or route  @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")
  @app.route('/result',methods=["GET","POST"]) def res():     if request.method=="POST":
        f=request.files['image']          basepath=os.path.dirname(__file__) #getting the current path
i.e where app.py is present
        #print("current  path",basepath)               filepath=os.path.join(basepath,'uploads',f.filename)
#from anywhere in the system we can give image but we want that image later  to process so we
are saving it to uploads folder for reusing       #print("upload folder is",filepath)       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 mod
 pred = model.predict(img)
        #print(pred)
pred=pred.flatten()        pred
= list(pred)
```

= max(pred)

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

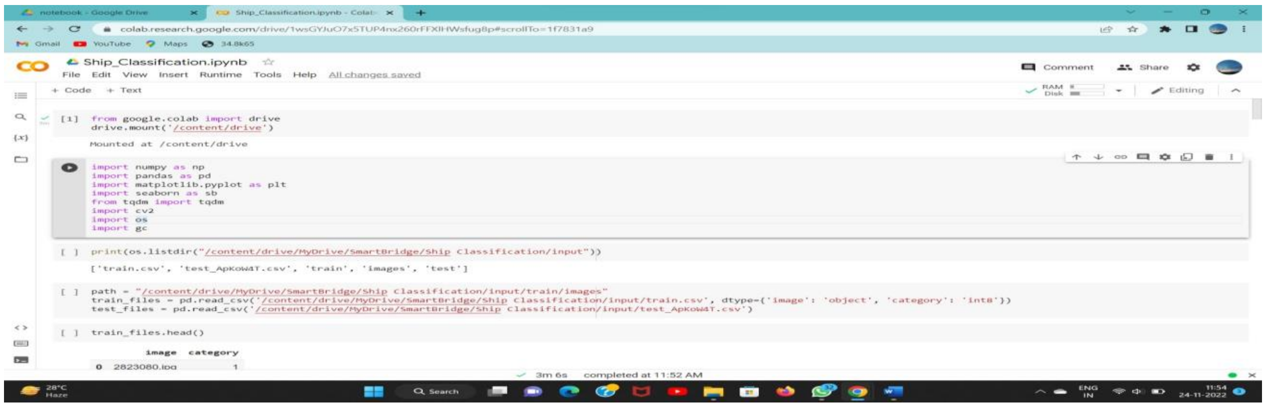
#print(val_dict[pred.index(m)])

result=val_dict[pred.index(m)] print(result)

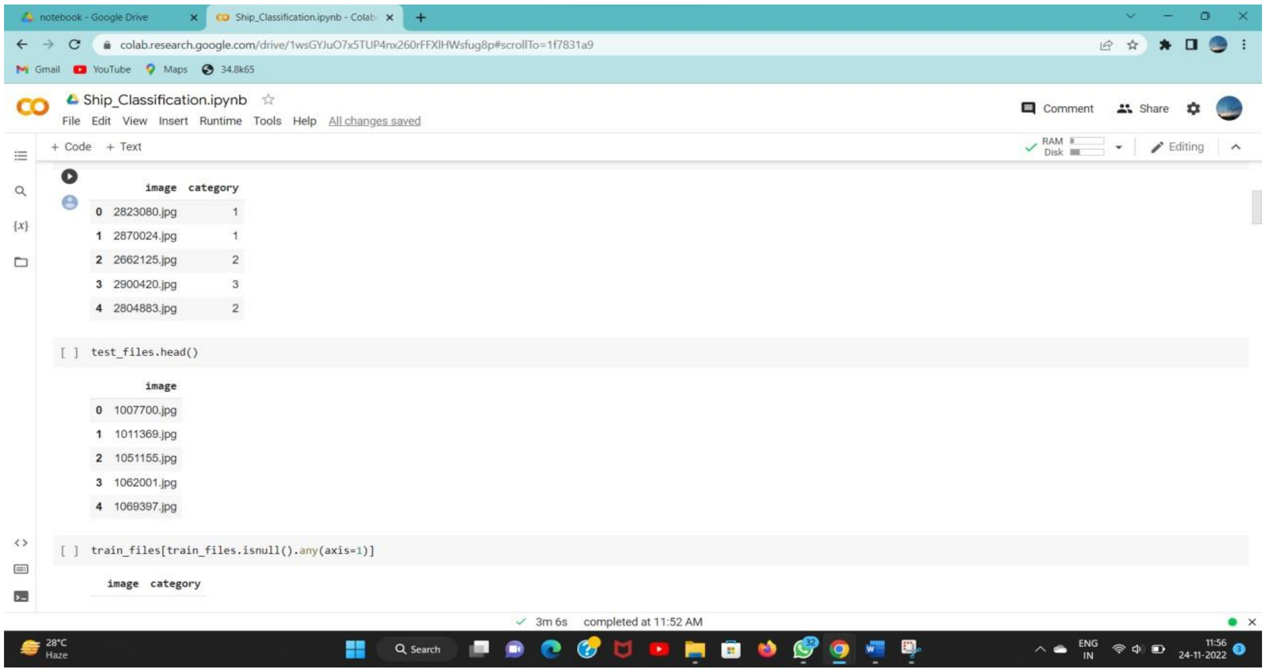
return render_template('prediction.html',prediction=result)

""" Running our application """ if __name__ == "__main__":

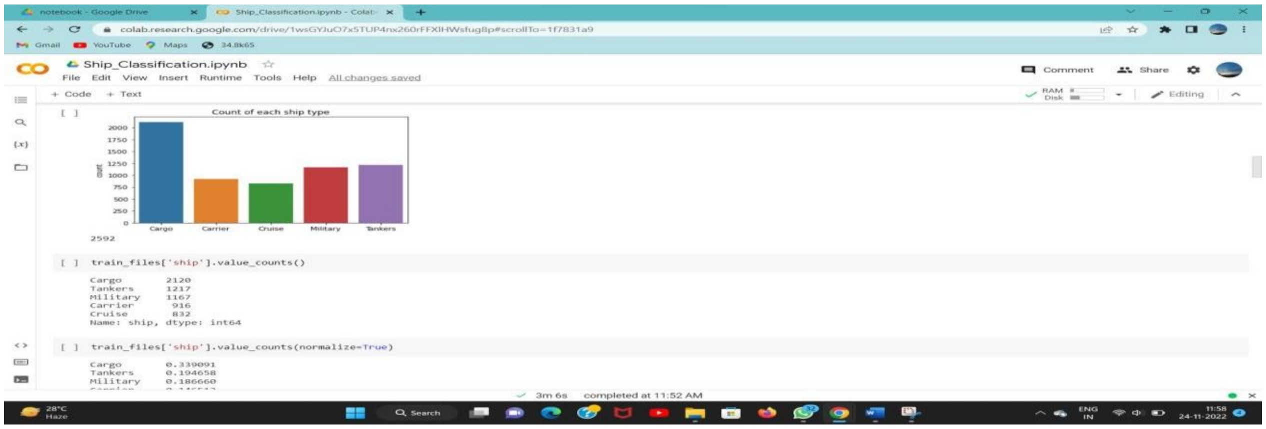
app.run()



```
[1] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sb
    from tqdm import tqdm
    import cv2
    import os
    import gc

[ ] print(os.listdir("/content/drive/MyDrive/SmartBridge/Ship Classification/input"))

    ['train.csv', 'test_ApKoW4T.csv', 'train', 'images', 'test']

[ ] path = "/content/drive/MyDrive/SmartBridge/Ship Classification/input/train/images"
    train_files = pd.read_csv('/content/drive/MyDrive/SmartBridge/Ship Classification/input/train.csv', dtype={'image': 'object', 'category': 'int8'})
    test_files = pd.read_csv('/content/drive/MyDrive/SmartBridge/Ship Classification/input/test_ApKoW4T.csv')

[ ] train_files.head()

         image  category
    0  2823080.jpg     1
```



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

[ ] test_files.head()

         image
    0  1007700.jpg
    1  1011369.jpg
    2  1051155.jpg
    3  1062001.jpg
    4  1069397.jpg

[ ] train_files[train_files.isnull().any(axis=1)]

       image  category
```



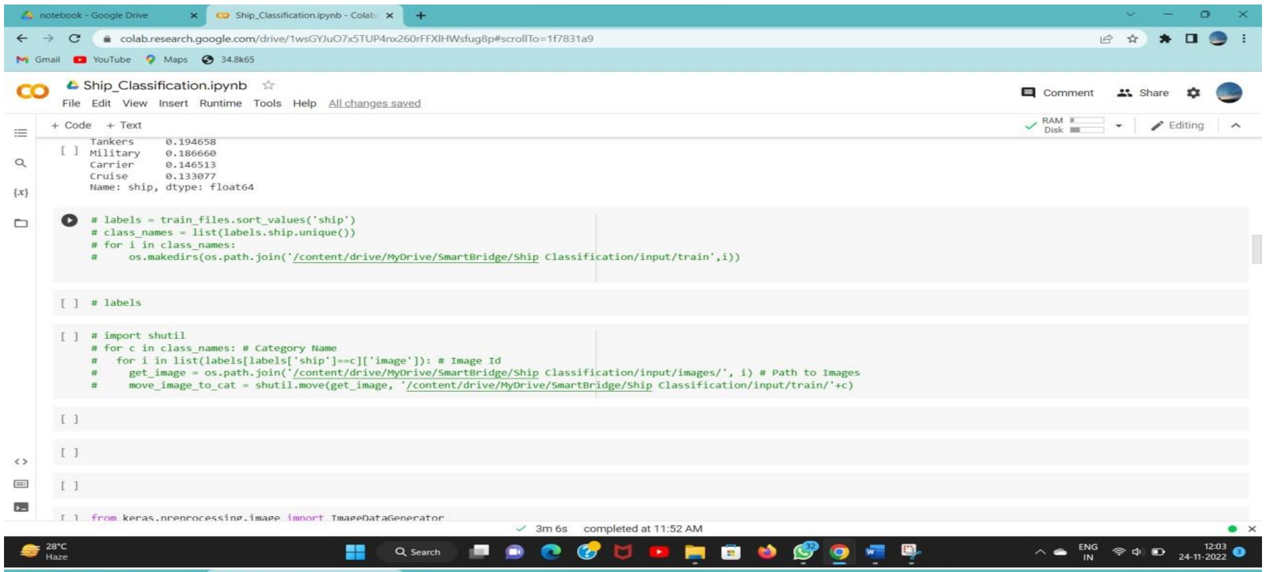
```
2592
```

```
train_files['ship'].value_counts()
```

```
Cargo       2120
Tankers     1217
Military    1167
Carrier      916
Cruise       832
Name: ship, dtype: int64
```

```
train_files['ship'].value_counts(normalize=True)
```

```
Cargo       0.339091
Tankers     0.194658
Military    0.186660
```



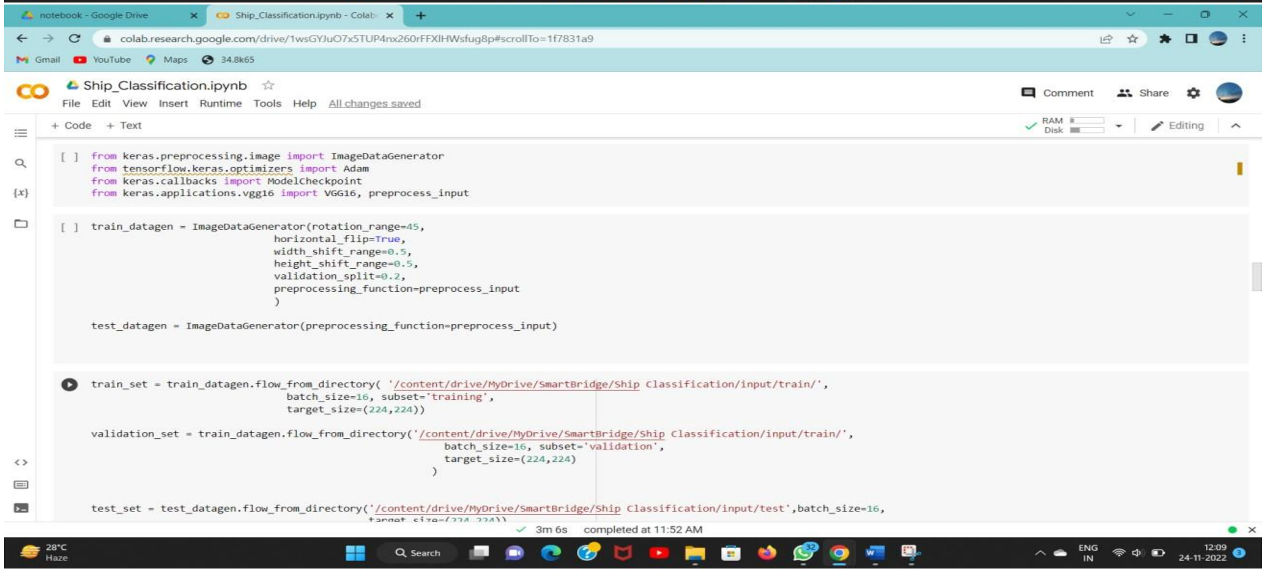
```
Tankers     0.194658
Military    0.186660
Carrier     0.146513
Cruise      0.133077
Name: ship, dtype: float64
```

```
# 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))
```

```
# labels
```

```
# 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)
```

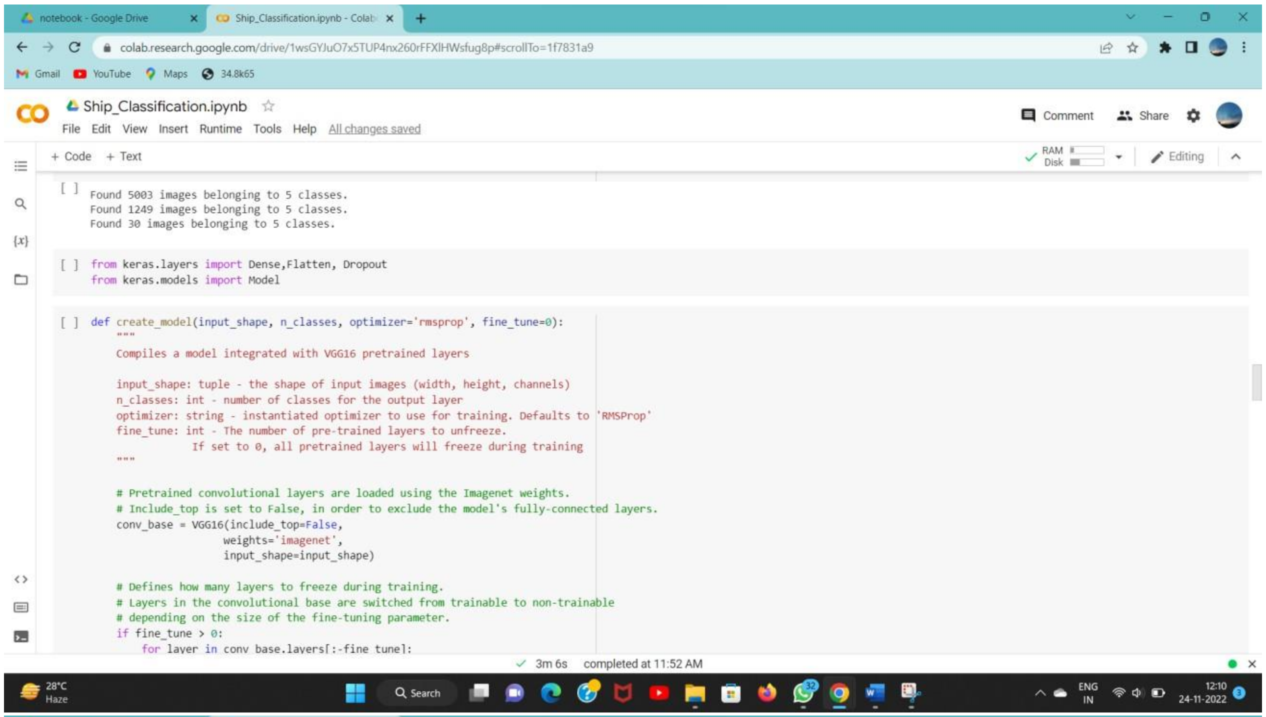
```
from keras.preprocessing.image import ImageDataGenerator
```



```
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from keras.applications.vgg16 import VGG16, preprocess_input
```

```
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)
```

```
train_set = train_datagen.flow_from_directory( '/content/drive/MyDrive/SmartBridge/Ship Classification/input/train/',
                               batch_size=16, subset='training',
                               target_size=(224,224))

validation_set = train_datagen.flow_from_directory('/content/drive/MyDrive/SmartBridge/Ship Classification/input/train/',
                                                        batch_size=16, subset='validation',
                                                        target_size=(224,224)
                                                      )


test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/SmartBridge/Ship Classification/input/test',batch_size=16,
                                            target_size=(224,224))
```

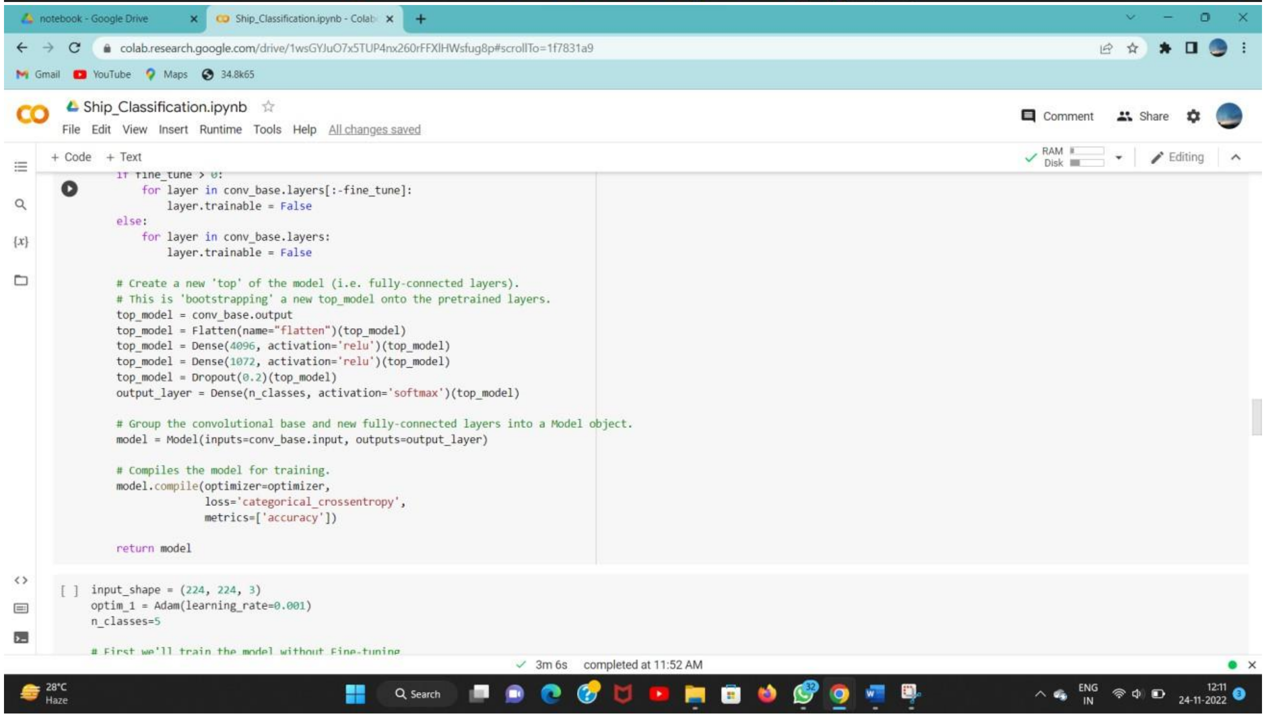

notebook - Google Drive
Ship_Classification.ipynb - Colab
colab.research.google.com/drive/1wsGYJuO7x5TUP4nx260rFFXlHWsfug8p#scrollTo=1f7831a9
Gmail
YouTube
Maps
34.8k65
Ship_Classification.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
All changes saved
Comment
Share
+ Code
+ Text
RAM
Disk
Editing

```
Found 5003 images belonging to 5 classes.
Found 1249 images belonging to 5 classes.
Found 30 images belonging to 5 classes.
```

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

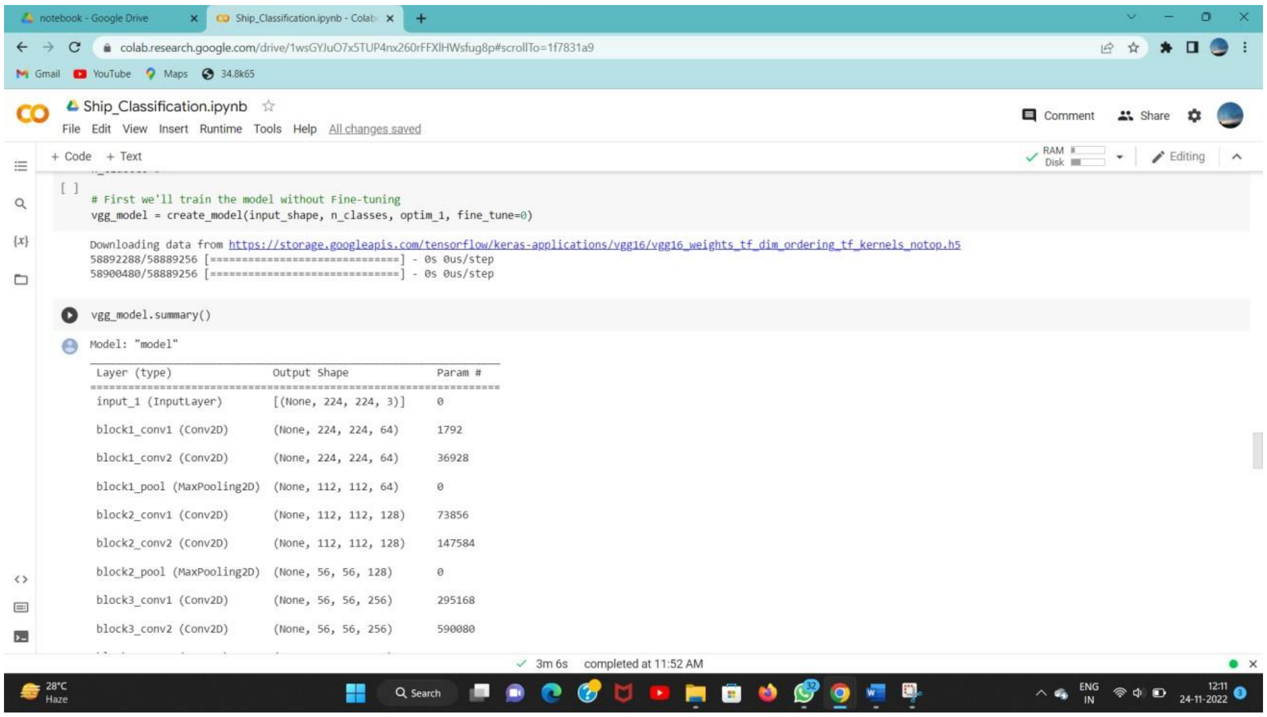
```python
def create_model(input_shape, n_classes, optimizer='rmsprop', fine_tune=0):
    """
    Compiles a model integrated with VGG16 pretrained layers

    input_shape: tuple - the shape of input images (width, height, channels)
    n_classes: int - number of classes for the output layer
    optimizer: string - instantiated optimizer to use for training. Defaults to 'RMSProp'
    fine_tune: int - The number of pre-trained layers to unfreeze.
                If set to 0, all pretrained layers will freeze during training
    """

    # Pretrained convolutional layers are loaded using the Imagenet weights.
    # Include_top is set to False, in order to exclude the model's fully-connected layers.
    conv_base = VGG16(include_top=False,
                      weights='imagenet',
                      input_shape=input_shape)

    # Defines how many layers to freeze during training.
    # Layers in the convolutional base are switched from trainable to non-trainable
    # depending on the size of the fine-tuning parameter.
    if fine_tune > 0:
        for layer in conv_base.layers[:-fine_tune]:
```

✓ 3m 6s completed at 11:52 AM
28°C
Haze
Search
ENG
IN
12:10
24-11-2022


notebook - Google Drive
Ship_Classification.ipynb - Colab
colab.research.google.com/drive/1wsGYJuO7x5TUP4nx260rFFXlHWsfug8p#scrollTo=1f7831a9
Gmail
YouTube
Maps
34.8k65
Ship_Classification.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
All changes saved
Comment
Share
+ Code
+ Text
RAM
Disk
Editing

```python
    if fine_tune > 0:
        for layer in conv_base.layers[:-fine_tune]:
            layer.trainable = False
    else:
        for layer in conv_base.layers:
            layer.trainable = False

    # Create a new 'top' of the model (i.e. fully-connected layers).
    # This is 'bootstrapping' a new top_model onto the pretrained layers.
    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)

    # Group the convolutional base and new fully-connected layers into a Model object.
    model = Model(inputs=conv_base.input, outputs=output_layer)

    # Compiles the model for training.
    model.compile(optimizer=optimizer,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

```python
input_shape = (224, 224, 3)
optim_1 = Adam(learning_rate=0.001)
n_classes=5

# First we'll train the model without Fine-tuning
```

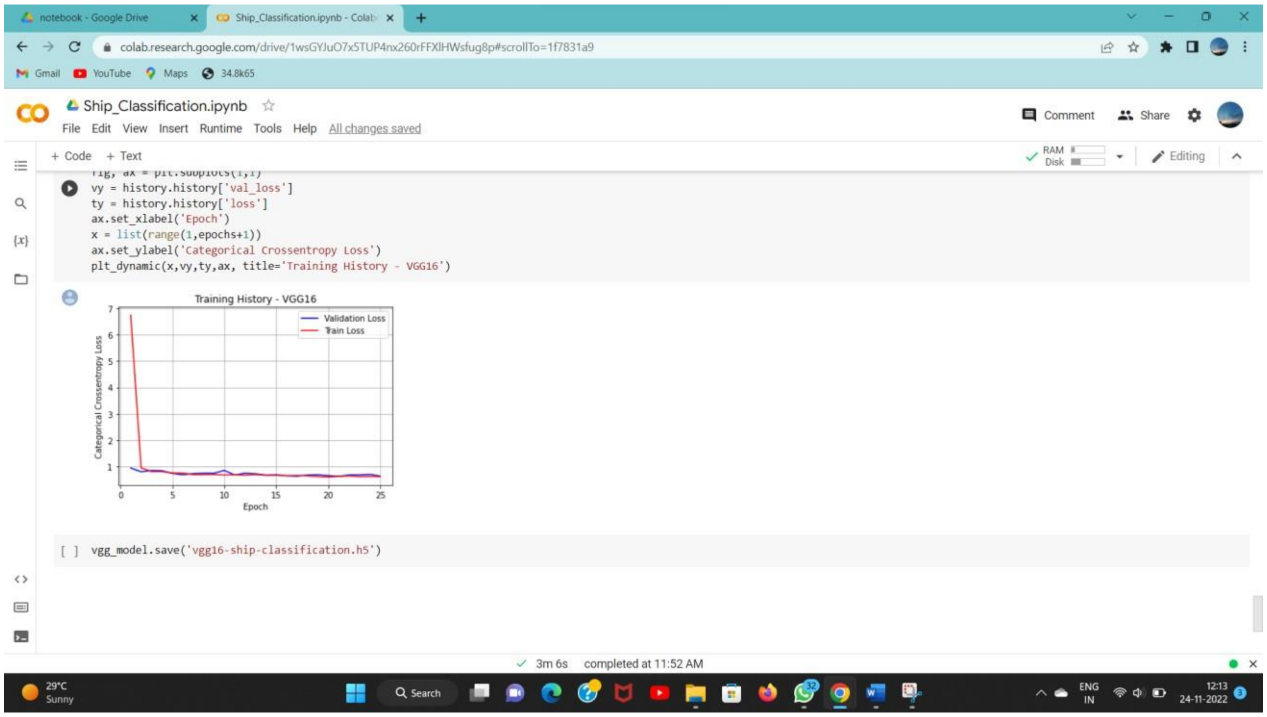

notebook - Google Drive
Ship_Classification.ipynb - Colab
colab.research.google.com/drive/1wsGYJuO7x5TUP4nx260rFFXlHWsfug8p#scrollTo=1f7831a9
Gmail
YouTube
Maps
34.8k65
Ship_Classification.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
✓ 3m 6s completed at 11:52 AM
28°C
Haze
Search
ENG
IN
12:11
24-11-2022

```
# First we'll train the model without Fine-tuning
vgg_model = create_model(input_shape, n_classes, optim_1, fine_tune=0)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
58900480/58889256 [==============================] - 0s 0us/step
```

```
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
```


notebook - Google Drive
Ship_Classification.ipynb - Colab
colab.research.google.com/drive/1wsGYJuO7x5TUP4nx260rFFXlHWsfug8p#scrollTo=1f7831a9
Gmail
YouTube
Maps
34.8k65
Ship_Classification.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
Training History - VGG16
Validation Loss
Train Loss
Categorical Crossentropy Loss
1
2
3
4
5
6
7
0
5
10
15
20
25
Epoch
✓ 3m 6s completed at 11:52 AM
29°C
Sunny
Search
ENG
IN
12:13
24-11-2022

```
fig, ax = plt.subplots(1,1)
vy = history.history['val_loss']
ty = history.history['loss']
ax.set_xlabel('Epoch')
x = list(range(1,epochs+1))
ax.set_ylabel('Categorical Crossentropy Loss')
plt_dynamic(x,vy,ty,ax, title='Training History - VGG16')
```

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