```
"""
    Created on Mon Oct 31 10:48:24 2022
    @author: Gokul R
    """
    #Importing Libraries

    #Locating and loading datasets
    import pathlib
    from pathlib import Path
    import os, gc, glob, random
    from PIL import Image

    #DataManagement and matrix calculations
    import pandas as pd
    import numpy as np

    #Model Building
    import tensorflow as tf
    import keras
    import keras.backend as K
    from keras.optimizers import SGD, Adam, Adagrad, RMSprop
    from keras.applications import *
    from keras.preprocessing import *
    from keras.preprocessing.image import ImageDataGenerator
    from keras.callbacks import EarlyStopping, ModelCheckpoint
    from keras.models import Sequential
    from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Activation, BatchNormalization,Dropout
    from keras.models import Model
    from keras.utils.np_utils import to_categorical
    from sklearn.model_selection import train_test_split

    # Data Visualization
    import matplotlib.pyplot as plt

    #Loading and testing models
    from keras.models import load_model
    from keras.models import model_from_json

    # Directory operations
    import os
    from os import listdir

    #
    ================================================================================================
    ============================== #
```

```
#
================================================================================================================
================================ #
# ========================================DEFINING THE REQUIRED
FUNCTIONS============================================================ #
#
================================================================================================================
================================ #
#
================================================================================================================
================================ #
def generateListofFiles(dirName):
"""This function returns a list with exact paths of files inside the given directory """
listOfFile = os.listdir(dirName)
allFiles = list()
for fol_name in listOfFile:
fullPath = os.path.join(dirName, fol_name)
allFiles.append(fullPath)

return allFiles

def Configure_CNN_Model(output_size):
"""This function defines the cnn model structure and configures the layers"""
K.clear_session()
model = Sequential()
model.add(Dropout(0.4,input_shape=(224, 224, 3)))

model.add(Conv2D(256, (5, 5),input_shape=(224, 224, 3),activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
#model.add(BatchNormalization())

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
#model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
#model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
```

```
model.add(Dense(output_size, activation='softmax'))

return model

def PrepreocessData(subfolders):
"""Pre precess the image data in the provided category list"""
X_data,Y_data,found = [],[],[]
id_no=0
#itering in all folders under Boats folder
for paths in subfolders:
#setting folder path for each boat type
files = glob.glob (paths + "/*.jpg")
found.append((paths.split('\\')[-2],paths.split('\\')[-1]))

#itering all files under the folder one by one
for myFile in files:
img = Image.open(myFile)
#img.thumbnail((width, height), Image.ANTIALIAS) # resizes image in-place keeps ratio
img = img.resize((224,224), Image.ANTIALIAS) # resizes image without ratio
#convert the images to numpy arrays
img = np.array(img)
if img.shape == ( 224, 224, 3):
# Add the numpy image to matrix with all data
X_data.append (img)
Y_data.append (id_no)
id_no+=1

#converting lists to np arrays again
X = np.array(X_data)
Y = np.array(Y_data)

# Print shapes to see if they are correct
print("x-shape",X.shape,"y shape", Y.shape)

X = X.astype('float32')/255.0
y_cat = to_categorical(Y_data, len(subfolders))

print("X shape",X,"y_cat shape", y_cat)
print("X shape",X.shape,"y_cat shape", y_cat.shape)

return X_data,Y_data,X,y_cat,found;
def splitData():
X_train, X_test, y_train, y_test = train_test_split(X, y_cat, test_size=0.2)
print("The model has " + str(len(X_train)) + " inputs")
return X_train, X_test, y_train, y_test
#
```

```
# ================================================================================================================
# ================================ #
#
# ================================================================================================================
# ================================ #
# ==========================================LOADING THE DATA AND PRE-PROCESSING
DATA================================================== #
#
# ================================================================================================================
# ================================ #
#
# ================================================================================================================
# ================================ #
# Augument the datasets with AugumentData.py.
# The AugumentData.py will generate many images with the original dataset to increase the accuracy of the model.

# Loading the augumented data form local storage
aug_data_location = "C:/Users/0xluk/OneDrive/Documents/Digital Naturalist/augumented data"
Folders = generateListofFiles(aug_data_location)
subfolders = []
for num in range(len(Folders)):
sub_fols = generateListofFiles(Folders[num])
subfolders+=sub_fols

X_data,Y_data,X,y_cat,found= PrepreocessData(subfolders)
# Splitting the data to Test and Train
X_train, X_test, y_train, y_test = splitData()

#
# ================================================================================================================
# ================================ #
#
# ================================================================================================================
# ================================ #
# ======================================================BUILDING THE CNN
MODEL========================================================== #
#
# ================================================================================================================
# ================================ #
#
# ================================================================================================================
# ================================ #
early_stop_loss = EarlyStopping(monitor='loss', patience=3, verbose=1)
early_stop_val_acc = EarlyStopping(monitor='val_accuracy', patience=3, verbose=1)
model_callbacks=[early_stop_loss, early_stop_val_acc]
```

```
model = Configure_CNN_Model(6)
model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.001),metrics=['accuracy'])
weights = model.get_weights()
model.set_weights(weights)

#
========================================================================================================
============================== #
#
========================================================================================================
============================== #
# ==========================================================PREDECTING IMAGE
CLASSES================================================== #
#
========================================================================================================
============================== #
#
========================================================================================================
============================== #
image_number = random.randint(0,len(X_test))
predictions = model.predict([X_test[image_number].reshape(1, 224,224,3)])

for idx, result, x in zip(range(0,6), found, predictions[0]):
print("Label: {}, Type : {}, Species : {} , Score : {}%".format(idx, result[0],result[1], round(x*100,3)))

#predicting the class with max probability
ClassIndex=np.argmax(model.predict([X_test[image_number].reshape(1, 224,224,3)]),axis=1)
print(found[ClassIndex[0]])
#
========================================================================================================
============================== #
#
========================================================================================================
============================== #
# ==========================================================SAVING THE MODEL
LOCALLY================================================== #
#
========================================================================================================
============================== #
#
========================================================================================================
============================== #
model_json = model.to_json() #indent=2
with open("DigitalNaturalist.json", "w") as json_file:
json_file.write(model_json)
```

```
# serialize weights to H5

model.save_weights("DigitalNaturalist.h5")

print("Saved model to disk")

#CNN model tested with 86% accuracy
```