```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "5eca6699",
   "metadata": {},
   "outputs": [],
   "source": [
    "#Importing Libraries\n",
    "\n",
    "#Locating and loading datasets\n",
    "import pathlib\n",
    "from pathlib import Path\n",
    "import os, gc, glob, random\n",
    "from PIL import Image\n",
    "\n",
    "#DataManagement and matrix calculations\n",
    "import pandas as pd\n",
    "import numpy as np\n",
    "\n",
    "#Model Building\n",
    "import tensorflow as tf\n",
    "import keras \n",
    "import keras.backend as K\n",
    "from keras.optimizers import SGD, Adam, Adagrad, RMSprop\n",
    "from keras.applications import *\n",
    "from keras.preprocessing import *\n",
    "from keras.preprocessing.image import ImageDataGenerator\n",
    "from keras.callbacks import EarlyStopping, ModelCheckpoint\n",
    "from keras.models import Sequential\n",
    "from keras.layers import Dense, Conv2D, MaxPool2D, Flatten,
Activation, BatchNormalization,Dropout\n",
    "from keras.models import Model\n",
    "from keras.utils.np_utils import to_categorical\n",
    "from sklearn.model_selection import train_test_split\n",
    "\n",
    "# Data Visualization\n",
    "import matplotlib.pyplot as plt\n",
    "\n",
    "#Loading and testing models\n",
    "from keras.models import load_model\n",
    "from keras.models import model_from_json\n",
    "\n",
    "# Directory operations\n",
    "import os\n",
    "from os import listdir\n",
    "\n",
    "#
==========================================================================
============================================================ # \n",
    "#
==========================================================================
============================================================ #\n",
    "# ==========================================DEFINING THE REQUIRED
FUNCTIONS=============================================================
#\n",
```

```
    "#
==========================================================================
============================================================ #\n",
    "#
==========================================================================
============================================================ #\n",
    "def generateListofFiles(dirName):\n",
    "    \"\"\"This function returns a list with exact paths of files
inside the given directory \"\"\"\n",
    "    listOfFile = os.listdir(dirName)\n",
    "    allFiles = list()\n",
    "    for fol_name in listOfFile:\n",
    "        fullPath = os.path.join(dirName, fol_name)\n",
    "        allFiles.append(fullPath)\n",
    "              \n",
    "    return allFiles\n",
    "\n",
    "def Configure_CNN_Model(output_size):\n",
    "    \"\"\"This function defines the cnn model structure and
configures the layers\"\"\"\n",
    "    K.clear_session()\n",
    "    model = Sequential()\n",
    "    model.add(Dropout(0.4,input_shape=(224, 224, 3)))\n",
    "    \n",
    "    model.add(Conv2D(256, (5, 5),input_shape=(224, 224,
3),activation='relu'))\n",
    "    model.add(MaxPool2D(pool_size=(2, 2)))\n",
    "    #model.add(BatchNormalization())\n",
    "\n",
    "    model.add(Conv2D(128, (3, 3), activation='relu'))\n",
    "    model.add(MaxPool2D(pool_size=(2, 2)))\n",
    "    #model.add(BatchNormalization())\n",
    "\n",
    "    model.add(Conv2D(64, (3, 3), activation='relu'))\n",
    "    model.add(MaxPool2D(pool_size=(2, 2)))\n",
    "    #model.add(BatchNormalization())\n",
    "\n",
    "    model.add(Flatten())\n",
    "    model.add(Dense(512, activation='relu'))\n",
    "    model.add(Dropout(0.3))\n",
    "    model.add(Dense(256, activation='relu'))\n",
    "    model.add(Dropout(0.3))\n",
    "    model.add(Dense(128, activation='relu'))\n",
    "    model.add(Dropout(0.3))\n",
    "\n",
    "    model.add(Dense(output_size, activation='softmax'))\n",
    "    \n",
    "    return model\n",
    "\n",
    "def PrepreocessData(subfolders):\n",
    "    \"\"\"Pre precess the image data in the provided category
list\"\"\"\n",
    "    X_data,Y_data,found = [],[],[]\n",
    "    id_no=0\n",
    "    #itering in all folders under Boats folder\n",
    "    for paths in subfolders:\n",
    "        #setting folder path for each boat type\n",
    "        files = glob.glob (paths + \"/*.jpg\")\n",
```

```
    "        found.append((paths.split('\\\\')[-2],paths.split('\\\\')[-
1]))\n",
    "        \n",
    "        #itering all files under the folder one by one\n",
    "        for myFile in files:\n",
    "            img = Image.open(myFile)\n",
    "            #img.thumbnail((width, height), Image.ANTIALIAS) #
resizes image in-place keeps ratio\n",
    "            img = img.resize((224,224), Image.ANTIALIAS) # resizes
image without ratio\n",
    "            #convert the images to numpy arrays\n",
    "            img = np.array(img)\n",
    "            if img.shape == ( 224, 224, 3):\n",
    "                # Add the numpy image to matrix with all data\n",
    "                X_data.append (img)\n",
    "                Y_data.append (id_no)\n",
    "        id_no+=1\n",
    "\n",
    "    #converting lists to np arrays again\n",
    "    X = np.array(X_data)\n",
    "    Y = np.array(Y_data)\n",
    "\n",
    "    # Print shapes to see if they are correct\n",
    "    print(\"x-shape\",X.shape,\"y shape\", Y.shape)\n",
    "\n",
    "    X = X.astype('float32')/255.0\n",
    "    y_cat = to_categorical(Y_data, len(subfolders))\n",
    "\n",
    "    print(\"X shape\",X,\"y_cat shape\", y_cat)\n",
    "    print(\"X shape\",X.shape,\"y_cat shape\", y_cat.shape)\n",
    "\n",
    "    return X_data,Y_data,X,y_cat,found; \n",
    "def splitData():\n",
    "    X_train, X_test, y_train, y_test = train_test_split(X, y_cat,
test_size=0.2)\n",
    "    print(\"The model has \" + str(len(X_train)) + \" inputs\")\n",
    "    return X_train, X_test, y_train, y_test\n",
    "#
========================================================================
========================================================== # \n",
    "#
========================================================================
========================================================== #\n",
    "# ========================================LOADING THE DATA AND
PRE-PROCESSING DATA=====================================================
#\n",
    "#
========================================================================
========================================================== #\n",
    "#
========================================================================
========================================================== #\n",
    "# Augument the datasets with AugumentData.py.\n",
    "# The AugumentData.py will generate many images with the original
dataset to increase the accuracy of the model.\n",
    "\n",
    "\n",
    "# Loading the augumented data form local storage\n",
```

```
    "aug_data_location = \"C:/Users/0xluk/OneDrive/Documents/Digital
Naturalist/augumented data\" \n",
    "Folders = generateListofFiles(aug_data_location)\n",
    "subfolders = []\n",
    "for num in range(len(Folders)):\n",
    "    sub_fols = generateListofFiles(Folders[num])\n",
    "    subfolders+=sub_fols\n",
    "\n",
    "X_data,Y_data,X,y_cat,found= PrepreocessData(subfolders)\n",
    "# Splitting the data to Test and Train\n",
    "X_train, X_test, y_train, y_test = splitData()\n",
    "\n",
    "#
================================================================================
============================================================ # \n",
    "#
================================================================================
============================================================ #\n",
    "# ==========================================================BUILDING THE
CNN MODEL============================================================
#\n",
    "#
================================================================================
============================================================ #\n",
    "#
================================================================================
============================================================ #\n",
    "early_stop_loss = EarlyStopping(monitor='loss', patience=3,
verbose=1)\n",
    "early_stop_val_acc = EarlyStopping(monitor='val_accuracy',
patience=3, verbose=1)\n",
    "model_callbacks=[early_stop_loss, early_stop_val_acc]\n",
    "\n",
    "model = Configure_CNN_Model(6)\n",

"model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.001),m
etrics=['accuracy'])\n",
    "weights = model.get_weights()\n",
    "model.set_weights(weights)\n",
    "\n",
    "#
================================================================================
============================================================ # \n",
    "#
================================================================================
============================================================ #\n",
    "# ==========================================================PREDECTING
IMAGE CLASSES============================================================ #\n",
    "#
================================================================================
============================================================ #\n",
    "#
================================================================================
============================================================ #\n",
    "image_number = random.randint(0,len(X_test))\n",
    "predictions = model.predict([X_test[image_number].reshape(1,
224,224,3)])\n",
    "\n",
```

```
    "for idx, result, x in zip(range(0,6), found, predictions[0]):\n",
    "   print(\"Label: {}, Type : {}, Species : {} , Score :
{}%\".format(idx, result[0],result[1], round(x*100,3)))\n",
    "\n",
    "\n",
    "#predicting the class with max probability\n",
    "ClassIndex=np.argmax(model.predict([X_test[image_number].reshape(1,
224,224,3)]),axis=1)\n",
    "print(found[ClassIndex[0]])\n",
    "#
========================================================================
========================================================== # \n",
    "#
========================================================================
========================================================== #\n",
    "# ==========================================================SAVING THE
MODEL LOCALLY========================================================== #\n",
    "#
========================================================================
========================================================== #\n",
    "#
========================================================================
========================================================== #\n",
    "model_json = model.to_json() #indent=2\n",
    "with open(\"DigitalNaturalist.json\", \"w\") as json_file:\n",
    "    json_file.write(model_json)\n",
    "\n",
    "# serialize weights to H5\n",
    "\n",
    "model.save_weights(\"DigitalNaturalist.h5\")\n",
    "print(\"Saved model to disk\")\n",
    "\n",
    "#CNN model tested with 86% accuracy"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.9.13"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```