#importing the necessary libraries required to run the program

import matplotlib.pyplot as plt #data visualisation tool

import seaborn as sns

import keras

from keras.modelDatas import Sequential

from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout

from keras.preprocessing.image import ImageDataGenerator

from keras.optimizers import Adam

from sklearn.metrics import classification\_report,confusion\_matrix

import tensorflow as tf

import torch

import torchvision

import cv2 #image recognition tool

import os

import numpy as np

##  Load the dataset

gamelabels = ['isRugby', 'isSoccer']

img\_size = 224

def get\_data(data\_dir):

    data = []

    for gamelabel in gamelabels:

        path = os.path.join(data\_dir, gamelabel)

        class\_num = gamelabels.index(gamelabel)

        for img in os.listdir(path):

            try:

                img\_arr = cv2.imread(os.path.join(path, img))[...,::-1]

                resized\_arr = cv2.resize(img\_arr, (img\_size, img\_size))

                data.append([resized\_arr, class\_num])

            except Exception as e:

                print(e)

    return np.array(data)

##Fetching the data;

trainData = get\_data('../input/traintestsports/Main/train')

valData = get\_data('../input/traintestsports/Main/test')

#Plot the graph

l = []

for i in trainData:

    if(i[1] == 0):

        l.append("Rugby")

    else:

        l.append("Soccer")

sns.set\_style('darkgrid')

sns.countplot(l)

#Randomly grab one rugby  image

plt.figure(figsize = (5,5))

plt.imshow(trainData[1][0])

plt.title(gamelabels[trainData[0][1]])

#Randomly grab a socccer image

plt.figure(figsize = (5,5))

plt.imshow(trainData[-1][0])

plt.title(gamelabels[trainData[-1][1]])

#Then data augemnetaion

x\_train = []

y\_train = []

x\_val = []

y\_val = []

for feature, label in trainData:

  x\_train.append(feature)

  y\_train.append(label)

for feature, label in valData:

  x\_val.append(feature)

  y\_val.append(label)

# Normalize the data

x\_train = np.array(x\_train) / 255

x\_val = np.array(x\_val) / 255

x\_train.reshape(-1, img\_size, img\_size, 1)

y\_train = np.array(y\_train)

x\_val.reshape(-1, img\_size, img\_size, 1)

y\_val = np.array(y\_val)

#Proceed with augmenation

datagenerated = ImageDataGenerator(

        featurewise\_center=False,

        samplewise\_center=False,

        featurewise\_std\_normalization=False,

        samplewise\_std\_normalization=False,

        zca\_whitening=False,

        rotation\_range = 30,

        zoom\_range = 0.2,

        width\_shift\_range=0.1,

        height\_shift\_range=0.1,

        horizontal\_flip = True,

        vertical\_flip=False)

datagenerated.fit(x\_train)

#Defining the modelData

modelData = Sequential()

modelData.add(Conv2D(32,3,padding="same", activation="relu", input\_shape=(224,224,3)))

modelData.add(MaxPool2D())

modelData.add(Conv2D(32, 3, padding="same", activation="relu"))

modelData.add(MaxPool2D())

modelData.add(Conv2D(64, 3, padding="same", activation="relu"))

modelData.add(MaxPool2D())

modelData.add(Dropout(0.4))

modelData.add(Flatten())

modelData.add(Dense(128,activation="relu"))

modelData.add(Dense(2, activation="softmax"))

modelData.summary()

#Compiling the model

opt = Adam(lr=0.000001)

modelData.compile(optimizer = opt , loss = tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True) , metrics = ['accuracy'])

historyData = modelData.fit(x\_train,y\_train,epochs = 400 , validation\_data = (x\_val, y\_val))

#Results evaluations

acc = historyData.historyData['accuracy']

val\_acc = historyData.historyData['val\_accuracy']

loss = historyData.historyData['loss']

val\_loss = historyData.historyData['val\_loss']

epochs\_range = range(500)

plt.figure(figsize=(15, 15))

plt.subplot(2, 2, 1)

plt.plot(epochs\_range, acc, label='Training Accuracy')

plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(2, 2, 2)

plt.plot(epochs\_range, loss, label='Training Loss')

plt.plot(epochs\_range, val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

#show the resulting predcitions of data

modelPredictions = modelData.predict\_classes(x\_val)

modelPredictions = modelPredictions.reshape(1,-1)[0]

print(classification\_report(y\_val, modelPredictions, target\_names = ['Rugby (Class 0)','Soccer (Class 1)']))