***ANIMAL SPECIES DETECTION USING CNN***

DATA PROCESSING:

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

import seaborn as sns

import numpy as np

import pandas as pd

import re

import os

import glob

import cv2

DATA PIPELINE:

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import\*

import matplotlib.pyplot as plt

from sklearn.model\_selection import\*

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.feature\_selection import VarianceThreshold

from sklearn.pipeline import make\_pipeline

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.decomposition import TruncatedSVD

from skimage import data, color

from skimage.transform import rescale, resize, downscale\_local\_mean

import matplotlib.image as mpimg

DATA PREDICTION:

from sklearn.linear\_model import\*

from sklearn.preprocessing import\*

from sklearn.ensemble import\*

from sklearn.neighbors import\*

from sklearn import svm

from sklearn.naive\_bayes import\*

import xgboost as xgb

DEEP LEARNING LIBRARIES:

import tensorflow as tf

from tensorflow.keras import Sequential

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

from tensorflow.keras.optimizers import Adam

print(tf.\_\_version\_\_)

GATHERING FILE PATH:

k=0

for i **in** os.listdir('/kaggle/input/african-wildlife/buffalo/'):

if i[-3:] !='txt':

img=mpimg.imread('/kaggle/input/african-wildlife/buffalo/'+i)

plt.imshow(img)

plt.show()

k+=1

if k==3:

break

BUFFALO DATA FILE:

df=[]

label=[]

for i **in** os.listdir('/kaggle/input/african-wildlife/buffalo/'):

if i[-3:] !='txt':

img=mpimg.imread('/kaggle/input/african-wildlife/buffalo/'+i)

img=resize(img,(128,128),anti\_aliasing=True)

tensor=img.reshape(49152)

df.append(tensor)

for i **in** range(len(df)):

label.append(0)

df=pd.DataFrame(df)

label=pd.DataFrame({'label':label})

data=pd.concat([label,df],axis=1)

X=data.drop('label',1)

y=data['label']

xtr1,xts1,ytr1,yts1=train\_test\_split(X,y,test\_size=0.2,random\_state=0)

ELEPHANT DATA FILE:

df=[]

label=[]

for i **in** os.listdir('/kaggle/input/african-wildlife/elephant/'):

if i[-3:] !='txt':

img=mpimg.imread('/kaggle/input/african-wildlife/elephant/'+i)

img=resize(img,(128,128),anti\_aliasing=True)

tensor=img.reshape(49152)

df.append(tensor)

for i **in** range(len(df)):

label.append(1)

df=pd.DataFrame(df)

label=pd.DataFrame({'label':label})

data=pd.concat([label,df],axis=1)

X=data.drop('label',1)

y=data['label']

xtr2,xts2,ytr2,yts2=train\_test\_split(X,y,test\_size=0.2,random\_state=0)

RHINO DATA FILE:

df=[]

label=[]

for i **in** os.listdir('/kaggle/input/african-wildlife/rhino/'):

if i[-3:] !='txt':

img=mpimg.imread('/kaggle/input/african-wildlife/rhino/'+i)

img=resize(img,(128,128),anti\_aliasing=True)

tensor=img.reshape(49152)

df.append(tensor)

for i **in** range(len(df)):

label.append(2)

df=pd.DataFrame(df)

label=pd.DataFrame({'label':label})

data=pd.concat([label,df],axis=1)

X=data.drop('label',1)

y=data['label']

xtr3,xts3,ytr3,yts3=train\_test\_split(X,y,test\_size=0.2,random\_state=0)

ZEBRA DATA FILE:

df=[]

label=[]

for i **in** os.listdir('/kaggle/input/african-wildlife/zebra/'):

if i[-3:] !='txt':

img=mpimg.imread('/kaggle/input/african-wildlife/zebra/'+i)

img=resize(img,(128,128),anti\_aliasing=True)

tensor=img.reshape(49152)

df.append(tensor)

for i **in** range(len(df)):

label.append(3)

df=pd.DataFrame(df)

label=pd.DataFrame({'label':label})

data=pd.concat([label,df],axis=1)

X=data.drop('label',1)

y=data['label']

xtr4,xts4,ytr4,yts4=train\_test\_split(X,y,test\_size=0.2,random\_state=0)

MODEL GENERATION:

model=Sequential()

model.add(Conv2D(64,activation='relu',kernel\_size=(3,3),input\_shape=X\_train[0].shape))

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(64,activation='relu'))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(128,activation='relu'))

model.add(Dense(4,activation='softmax'))

MODEL ACCURACY:

plt.title('Model Accuracy')

plt.plot(hist.history['accuracy'],label='train')

plt.plot(hist.history['val\_accuracy'],label='validation')

plt.legend()

plt.show()

plt.title('Model Loss')

plt.plot(hist.history['loss'],label='train')

plt.plot(hist.history['val\_loss'],label='validation')

plt.legend()

plt.show()

plt.title('Model Accuracy')

plt.plot(hist.history['accuracy'],label='train')

plt.plot(hist.history['val\_accuracy'],label='validation')

plt.legend()

plt.show()

plt.title('Model Loss')

plt.plot(hist.history['loss'],label='train')

plt.plot(hist.history['val\_loss'],label='validation')

plt.legend()

plt.show()

plt.title('Model Accuracy')

plt.plot(hist.history['accuracy'],label='train')

plt.plot(hist.history['val\_accuracy'],label='validation')

plt.legend()

plt.show()

plt.title('Model Loss')

plt.plot(hist.history['loss'],label='train')

plt.plot(hist.history['val\_loss'],label='validation')

plt.legend()

plt.show()