import tensorflow\_datasets as tfds

import tensorflow as tf

from tensorflow.keras.utils import to\_categorical

## Loading images and labels

(train\_ds, train\_labels), (test\_ds, test\_labels) = tfds.load("tf\_flowers",

split=["train[:70%]", "train[:30%]"], ## Train test split

batch\_size=-1,

as\_supervised=True, # Include labels

)

## check existing image size

train\_ds[0].shape

## Resizing images

train\_ds = tf.image.resize(train\_ds, (150, 150))

test\_ds = tf.image.resize(test\_ds, (150, 150))

train\_labels

## Transforming labels to correct format

train\_labels = to\_categorical(train\_labels, num\_classes=5)

test\_labels = to\_categorical(test\_labels, num\_classes=5)

train\_labels[0]

from tensorflow.keras.applications.vgg16 import VGG16

from tensorflow.keras.applications.vgg16 import preprocess\_input

train\_ds[0].shape

## Loading VGG16 model

base\_model = VGG16(weights="imagenet", include\_top=False, input\_shape=train\_ds[0].shape)

## will not train base mode

# Freeze Parameters in model's lower convolutional layers

base\_model.trainable = False

## Preprocessing input

train\_ds = preprocess\_input(train\_ds)

test\_ds = preprocess\_input(test\_ds)

## model details

base\_model.summary()

#add our layers on top of this model

from tensorflow.keras import layers, models

flatten\_layer = layers.Flatten()

dense\_layer\_1 = layers.Dense(50, activation='relu')

dense\_layer\_2 = layers.Dense(20, activation='relu')

prediction\_layer = layers.Dense(5, activation='softmax')

model = models.Sequential([

base\_model,

flatten\_layer,

dense\_layer\_1,

dense\_layer\_2,

prediction\_layer

])

from tensorflow.keras.callbacks import EarlyStopping

model.compile(

optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'],

)

es = EarlyStopping(monitor='val\_accuracy', mode='max', patience=5, restore\_best\_weights=True)

history=model.fit(train\_ds, train\_labels, epochs=50, validation\_split=0.2, batch\_size=32, callbacks=[es])

los,accurac=model.evaluate(test\_ds,test\_labels)

print("Loss: ",los,"Accuracy: ", accurac)

import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])

plt.title('ACCURACY')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train'],loc='upper left')

plt.show()

import numpy as np

import pandas as pd

y\_pred = model.predict(test\_ds)

y\_classes = [np.argmax(element) for element in y\_pred]

#to\_categorical(y\_classes, num\_classes=5)

#to\_categorical(test\_labels, num\_classes=5)

print(y\_classes[:10])

print("\nTest")

print(test\_labels[:10])