LA ESTRUCTURA CNN YA ESTA, SE QUIERE UTILIZAR PERO CON UN SET DE IMÁGENES A ENTRENAR.

ES DECIR, SI UNA ESTRUCTURA ES MUY BUENA SE REUTILIZA, Y SE LE ASIGNA UN GRUPO DE IMÁGENES PARA ENTRENAR, NO SE LE ‘AGREGA’ MAS IMÁGENES, AL MODELO SE LE REENTRENA CON UN NUEVO SET DE IMÁGENES.

SE REUTILIZARA EL MODELO DE REDES NEURALES CONVOLUCIONALES, YA ESTA DEFINIDAS LAS CAPAS

1. SE CARGA UN SET DE IMÁGENES
2. SE LE ASIGNARA SU TARGET
3. SE REENTRENARA
4. EL MODELO NUEVO FUNCIONARA UNICAMENTE PARA LAS NUEVAS IMÁGENES
5. SE DESCARTARA EL ENTRENAMIENTO DE IMÁGENES QUE TENIA
6. LO QUE SE APROVECHA ES QUE ESTE MODELO YA ESTA DEFINIDO QUE LAYERS UTILIZAR

import numpy as np

import cv2

import tensorflow as tf

import tensorflow\_hub as hub

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.models import Sequential

CARGAR IMÁGENES

**IMAGE\_SHAPE** = (224, 224)

INPUT\_SHAPE = (**224, 224**, 3)

dataset\_url = <https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz>

data\_dir = tf.keras.utils.get\_file('flower\_photos', origin=dataset\_url,  cache\_dir='.', untar=True)

Se crea directorios con los grupos de fotos

datasets

flower\_photos

daisy

dandelion

roses

sunflowers

tulips

CARGUE IMÁGENES EN MEMORIA

import pathlib

data\_dir = pathlib.Path(data\_dir)

flowers\_images\_dict = {

    'roses': list(data\_dir.glob('roses/\*')),

    'daisy': list(data\_dir.glob('daisy/\*')),

    'dandelion': list(data\_dir.glob('dandelion/\*')),

    'sunflowers': list(data\_dir.glob('sunflowers/\*')),

    'tulips': list(data\_dir.glob('tulips/\*')),

}

# Se crea el label de las flores, y se les asigna categoria

# Si es rosa sera 0, si es daisy sera 1

flowers\_labels\_dict = {

    'roses': 0,

    'daisy': 1,

    'dandelion': 2,

    'sunflowers': 3,

    'tulips': 4,

}

# RECORRA Y CARGUE

X, y = [], []

for flower\_name, images in flowers\_images\_dict.items():

    for image in images:

        img = cv2.imread(str(image))

        resized\_img = cv2.resize(img,**IMAGE\_SHAPE**)

        X.append(resized\_img)

        y.append(flowers\_labels\_dict[flower\_name])

X = np.array(X)

y = np.array(y)

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=0)

X\_train\_scaled = X\_train / 255

X\_test\_scaled = X\_test / 255

# TOME MODELO PRE-ENTRENADO Y REENTRENE CON NUEVAS IMÁGENES

PRE\_model = "https://tfhub.dev/google/tf2-preview/mobilenet\_v2/feature\_vector/4"

MODEL2 = hub.KerasLayer(PRE\_model, input\_shape=**INPUT\_SHAPE**, trainable=False)

categorias = 5 # Son 5 categorias de imágenes en este caso

NEW\_MODEL = tf.keras.Sequential([ MODEL2, tf.keras.layers.Dense(categorias)

])

NEW\_MODEL.compile(

  optimizer="adam",

  loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

  metrics=['acc'])

NEW\_MODEL.fit(X\_train\_scaled, y\_train, epochs=5)

SE UTILIZARA SOLO 5 EPOCHS, Y SERA EFECTIVO PORQUE YA ESTA BIEN DEFINIDO COMO RECONOCER IMÁGENES

model.evaluate(X\_test\_scaled,y\_test)

Output

[0.4112617, 0.857298]

Si es bueno, lo reentreno con toda la data, sin separar en train / test, o le aumenta epochs.

NEW\_MODEL.fit(X, y, epochs=10)

# Compruebe que funciona, con predicción

import PIL.Image as Image

imagen = Image.open("imagen\_flor.jpg").resize(**IMAGE\_SHAPE**)

imagen = np.array(imagen)/255.0

result = NEW\_MODEL.predict(imagen[np.newaxis, ...])

categoria = np.argmax(result)

categoría

Saldrá la categoría de 0 – 4, de acuerdo a lo que se especifico

flowers\_labels\_dict = {

    'roses': 0,

    'daisy': 1,

    'dandelion': 2,

    'sunflowers': 3,

    'tulips': 4,

}