// PARCIAL 9 - DISEÑO CON uP y uC. 2024-1.

// NOMBRE: Diego Andrés García Díaz.

// CÓDIGO: 2195533.

// -----------------------------------------------------------------------------

# -\*- coding: utf-8 -\*-

"""Copia de parcial9\_FINAL.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1chf0nXPHLhCwIVbzXX88qHZm\_HoG-3Hq

"""

!pip install tensorflowjs

# Se crean las carpetas en las que se almacenaran las diferentes imágenes de los

# componentes electrónicos.

from google.colab import drive

drive.mount('/content/drive')

# La ruta local a la carpeta en Google Drive

folder\_path = '/content/drive/MyDrive/Components'

!mkdir -p {folder\_path}/Condensadores\_Ceramicos

!mkdir -p {folder\_path}/Condensadores\_Electroliticos

!mkdir -p {folder\_path}/Condensadores\_Poliester

!mkdir -p {folder\_path}/Diodos\_Led

!mkdir -p {folder\_path}/Diodos\_Rectificadores

!mkdir -p {folder\_path}/Diodos\_Zener

!mkdir -p {folder\_path}/Resistencias\_1\_2\_W

!mkdir -p {folder\_path}/Resistencias\_1\_4\_W

!mkdir -p {folder\_path}/Resistencias\_1\_W

!mkdir -p {folder\_path}/Resistencias\_5\_W

# Luego de que se crean las carpetas, se deben cargar por medio de un archivo .zip

# que luego serán descomprimidos para poder usar las imágenes de cada componente.

# Commented out IPython magic to ensure Python compatibility.

# La siguiente parte de código descomprime los archivos .zip contenidos en cada carpeta

# anteriormente creada.

from google.colab import drive

drive.mount('/content/drive')

# Ruta base a la carpeta en Google Drive

base\_path = '/content/drive/MyDrive/Components'

# Lista de carpetas y archivos .zip correspondientes

folders\_and\_zips = [

("Condensadores\_Electroliticos", "Condensadores\_Electroliticos.zip"),

("Condensadores\_Poliester", "Condensadores\_Poliester.zip"),

("Condensadores\_Ceramicos", "Condensadores\_Ceramicos.zip"),

("Diodos\_Rectificadores", "Diodos\_Rectificadores.zip"),

("Diodos\_Zener", "Diodos\_Zener.zip"),

("Diodos\_Led", "Diodos\_Led.zip"),

("Resistencias\_1\_2\_W", "Resistencias\_de\_1\_2\_W.zip"),

("Resistencias\_1\_4\_W", "Resistencias\_de\_1\_4\_W.zip"),

("Resistencias\_1\_W", "Resistencias\_de\_1\_W.zip"),

("Resistencias\_5\_W", "Resistencias\_de\_5\_W.zip")

]

# Descomprimir cada archivo .zip en su respectiva carpeta

for folder, zip\_file in folders\_and\_zips:

folder\_path = f"{base\_path}/{folder}"

zip\_file\_path = f"{folder\_path}/{zip\_file}"

# %cd {folder\_path}

!unzip {zip\_file\_path}

# %cd ..

# Borrar archivos zip luego se ser descomprimidos

from google.colab import drive

drive.mount('/content/drive')

# Ruta base a la carpeta en Google Drive

base\_path = '/content/drive/MyDrive/Components'

# Lista de carpetas y archivos .zip correspondientes

folders\_and\_zips = [

("Condensadores\_Electroliticos", "Condensadores\_Electroliticos.zip"),

("Condensadores\_Poliester", "Condensadores\_Poliester.zip"),

("Condensadores\_Ceramicos", "Condensadores\_Ceramicos.zip"),

("Diodos\_Rectificadores", "Diodos\_Rectificadores.zip"),

("Diodos\_Zener", "Diodos\_Zener.zip"),

("Diodos\_Led", "Diodos\_Led.zip"),

("Resistencias\_1\_2\_W", "Resistencias\_de\_1\_2\_W.zip"),

("Resistencias\_1\_4\_W", "Resistencias\_de\_1\_4\_W.zip"),

("Resistencias\_1\_W", "Resistencias\_de\_1\_W.zip"),

("Resistencias\_5\_W", "Resistencias\_de\_5\_W.zip")

]

# Se eliminan los archivos .zip

for folder, zip\_file in folders\_and\_zips:

zip\_file\_path = f"{base\_path}/{folder}/{zip\_file}"

!rm -rf {zip\_file\_path}

# Se muestran cuantas imagenes hay en cada carpeta

from google.colab import drive

drive.mount('/content/drive')

# Ruta base a la carpeta en Google Drive

base\_path = '/content/drive/MyDrive/Components'

# Lista de carpetas correspondientes

folders = [

"Condensadores\_Electroliticos",

"Condensadores\_Poliester",

"Condensadores\_Ceramicos",

"Diodos\_Rectificadores",

"Diodos\_Zener",

"Diodos\_Led",

"Resistencias\_1\_2\_W",

"Resistencias\_1\_4\_W",

"Resistencias\_1\_W",

"Resistencias\_5\_W"

]

# Mostrar cuántas imágenes hay en cada carpeta

for folder in folders:

folder\_path = f"{base\_path}/{folder}"

print(f"{folder}:")

!ls {folder\_path} | wc -l

# Se crean las carpetas para hacer el respectivo set de datos

from google.colab import drive

drive.mount('/content/drive')

# Ruta base a la carpeta en Google Drive

base\_path = '/content/drive/MyDrive/Components'

# Crear la carpeta principal 'data' y las subcarpetas para cada conjunto de datos

!mkdir -p {base\_path}/data5/Condensadores\_Electroliticos

!mkdir -p {base\_path}/data5/Condensadores\_Ceramicos

!mkdir -p {base\_path}/data5/Condensadores\_Poliester

!mkdir -p {base\_path}/data5/Diodos\_Rectificadores

!mkdir -p {base\_path}/data5/Diodos\_Zener

!mkdir -p {base\_path}/data5/Diodos\_Led

!mkdir -p {base\_path}/data5/Resistencias\_1\_2\_W

!mkdir -p {base\_path}/data5/Resistencias\_1\_4\_W

!mkdir -p {base\_path}/data5/Resistencias\_1\_W

!mkdir -p {base\_path}/data5/Resistencias\_5\_W

print("Carpetas creadas en Google Drive en la ruta:", base\_path + "/datos")

# Se copian las imágenes y se les limita la cantidad de imagenes, esto con el fin de

# tener el mismo número de imágenes en cada carpeta.

from google.colab import drive

import os

import shutil

drive.mount('/content/drive')

# Ruta base a la carpeta en Google Drive

base\_path = '/content/drive/MyDrive/Components'

data\_path = f"{base\_path}/datos"

# Lista de carpetas correspondientes

folders = [

"Condensadores\_Electroliticos",

"Condensadores\_Ceramicos",

"Condensadores\_Poliester",

"Diodos\_Rectificadores",

"Diodos\_Zener",

"Diodos\_Led",

"Resistencias\_1\_2\_W",

"Resistencias\_1\_4\_W",

"Resistencias\_1\_W",

"Resistencias\_5\_W"

]

# Limitar la cantidad de imágenes a copiar

max\_images = 130

# Copiar imágenes y limitar para que todas tengan la misma cantidad de imágenes

for folder in folders:

carpeta\_fuente = f"{base\_path}/{folder}"

carpeta\_destino = f"{data\_path}/{folder}"

# Crear la carpeta de destino si no existe

if not os.path.exists(carpeta\_destino):

os.makedirs(carpeta\_destino)

imagenes = os.listdir(carpeta\_fuente)

for i, nombreimg in enumerate(imagenes):

if i < max\_images:

# Copia de la carpeta fuente a la destino

shutil.copy(os.path.join(carpeta\_fuente, nombreimg), os.path.join(carpeta\_destino, nombreimg))

print("Copiado de imágenes completado.")

# Se muestran cuantas imagenes hay en cada carpeta

from google.colab import drive

drive.mount('/content/drive')

# Ruta base a la carpeta en Google Drive

base\_path = '/content/drive/MyDrive/Components/datos'

# Lista de carpetas correspondientes

folders = [

"Condensadores\_Electroliticos",

"Condensadores\_Poliester",

"Condensadores\_Ceramicos",

"Diodos\_Rectificadores",

"Diodos\_Zener",

"Diodos\_Led",

"Resistencias\_1\_2\_W",

"Resistencias\_1\_4\_W",

"Resistencias\_1\_W",

"Resistencias\_5\_W"

]

# Mostrar cuántas imágenes hay en cada carpeta

for folder in folders:

folder\_path = f"{base\_path}/{folder}"

print(f"{folder}:")

!ls {folder\_path} | wc -l

#Aumento de datos con ImageDataGenerator

from google.colab import drive

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np

import matplotlib.pyplot as plt

drive.mount('/content/drive', force\_remount=True)

#Crear el dataset generador

datagen = ImageDataGenerator(

rescale=1. / 255,

rotation\_range = 30,

width\_shift\_range = 0.25,

height\_shift\_range = 0.25,

shear\_range = 15,

zoom\_range = [0.5, 1.5],

validation\_split=0.2 #20% para pruebas

)

#Generadores para sets de entrenamiento y pruebas

data\_gen\_entrenamiento = datagen.flow\_from\_directory('/content/drive/MyDrive/Components/datos', target\_size=(224,224),

batch\_size=32, shuffle=True, subset='training')

data\_gen\_pruebas = datagen.flow\_from\_directory('/content/drive/MyDrive/Components/datos', target\_size=(224,224),

batch\_size=32, shuffle=True, subset='validation')

#Imprimir 16 imagenes del generador de entrenamiento

for imagen, etiqueta in data\_gen\_entrenamiento:

for i in range(16):

plt.subplot(4,4,i+1)

plt.xticks([])

plt.yticks([])

plt.imshow(imagen[i])

break

plt.show()

# Se usa tensorflow que es el modelo de una red neuronal

import tensorflow as tf

import tensorflow\_hub as hub

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

url = "https://www.kaggle.com/models/google/mobilenet-v2/TensorFlow2/tf2-preview-feature-vector/4"

mobilenetv2 = hub.KerasLayer(url, input\_shape=(224,224,3))

# Se detiene el modelo de red neuronal descargado

mobilenetv2.trainable = False

modelo = tf.keras.Sequential([

mobilenetv2,

tf.keras.layers.Dense(10, activation='softmax')

])

modelo.summary()

# Se compila el modelo de red neuronal

modelo.compile(

optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy']

)

# Se entrena el modelo de red neuronal

EPOCAS = 42 # Se defiene para determinar la duración y mejora del entrenamiento del modelo

historial = modelo.fit(

data\_gen\_entrenamiento, epochs=EPOCAS, batch\_size=32,

validation\_data=data\_gen\_pruebas

)

# Se grafica la precisión del modelo, con el objetivo de ir mejorando el modelo de red neuronal

acc = historial.history['accuracy']

val\_acc = historial.history['val\_accuracy']

loss = historial.history['loss']

val\_loss = historial.history['val\_loss']

rango\_epocas = range(42)

plt.figure(figsize=(8,8))

plt.subplot(1,2,1)

plt.plot(rango\_epocas, acc, label='Precisión Entrenamiento',color='purple')

plt.plot(rango\_epocas, val\_acc, label='Precisión Pruebas',color='green')

plt.legend(loc='lower right')

plt.title('Precisión de entrenamiento y pruebas')

plt.subplot(1,2,2)

plt.plot(rango\_epocas, loss, label='Pérdida de entrenamiento',color='purple')

plt.plot(rango\_epocas, val\_loss, label='Pérdida de pruebas',color='green')

plt.legend(loc='upper right')

plt.title('Pérdida de entrenamiento y pruebas')

plt.show()

# Para esta parte se categorizará una imagen de internet a traves de su 'url'

from PIL import Image

import requests

from io import BytesIO

import cv2

def categorizar(url):

respuesta = requests.get(url)

img = Image.open(BytesIO(respuesta.content))

img = np.array(img).astype(float)/255

img = cv2.resize(img, (224,224))

prediccion = modelo.predict(img.reshape(-1, 224, 224, 3))

return np.argmax(prediccion[0], axis=-1)

# Esta parte categoriza la imagen gracias a su 'url' y se define un número para identificar el tipo de componente

# 0 = Capacitor\_Eletrolitico,

# 1 = Capacitor\_Poliester,

# 2 = Capacitor\_Ceramico,

# 3 = Diodo\_Rectificador,

# 4 = Diodo\_Zener,

# 5 = Diodo\_Led,

# 6 = Resistencia\_1\_2,

# 7 = Resistencia\_1\_4,

# 8 = Resistencia\_1 ,

# 9 = Resistencia\_5

url = 'https://cdn.shopify.com/s/files/1/0131/0792/0996/products/Resistencias\_de\_ceramica\_5W\_2048x2048.jpg?v=1583944678' #debe ser 2

url2= 'https://comeind.com.mx/wp-content/uploads/2022/02/electroliticos.jpg'

prediction = categorizar (url)

prediction2 = categorizar (url2)

print(prediction)

print(prediction2)

# Se exporta el modelo en formato h5

modelo.save('parcial9\_2195533-cnn-ad.h5')

!ls

!pip install tensorflowjs

!mkdir output\_folder

# Para poder usarlo en un explorador

!tensorflowjs\_converter --input\_format keras parcial9\_2195533-cnn-ad.h5 output\_folder

!ls

# Crear una carpeta para exportarla a TensroFlow Serving

!mkdir -p output\_folder/modelo\_parcial9\_2195533/1

# Guardar el modelo de red neuronal en formato SavedModel

modelo.save('output\_folder/modelo\_parcial9\_2195533/1')

# Se convierte en archivo .un zip para descargarlo y usarlo en otro lado

!zip -r modelo\_parcial9\_2195533.zip /content/output\_folder/