```
// 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_2048x2
048.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/
```