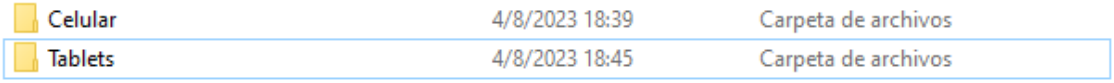
Integrantes: Cristian Idrobo, Jonathan Toapanta

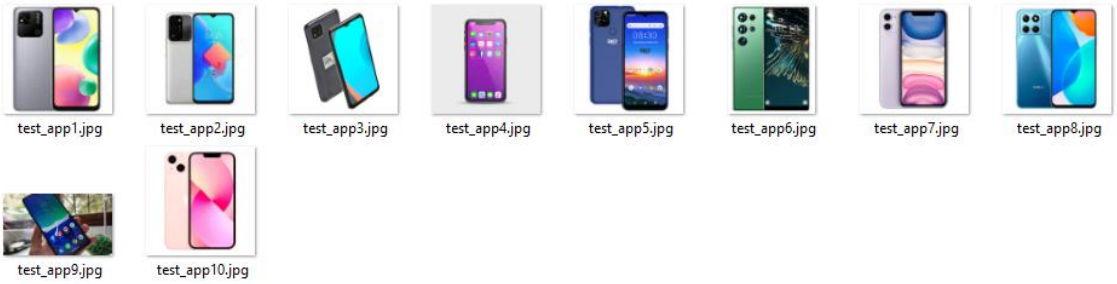
NRC: 10048

Tema: RED NEURONAL CONVOLUCIONAL

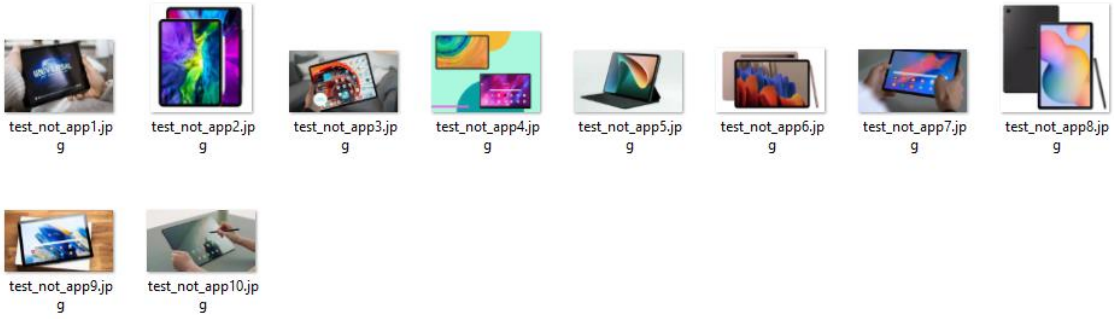
Creamos las carpetas de ejemplos train y data para el análisis y validación, dentro de ellas se encuentran las imágenes de celulares y tablets.



Imágenes de celulares



Imágenes de Tablets



Código de ejecución

```
import os               # Librería para interactuar con el sistema operativo (no se utiliza explícitamente en el código).
import numpy as np      # Librería para el procesamiento numérico en Python.
import tensorflow as tf # Librería de aprendizaje automático y redes neuronales desarrollada por Google.
import scipy            # Librería con herramientas para el procesamiento de señales, optimización, estadísticas, etc. (no se utiliza explícitamente en el código).
from tensorflow.keras import layers, models, optimizers  # Funciones y clases relacionadas con el diseño y entrenamiento de modelos de redes neuronales.
from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Clase para crear generadores de lotes de datos para el entrenamiento y validación de modelos.
from PIL import Image   # Clase para cargar y realizar operaciones básicas con imágenes, proporcionada por la librería Pillow (Python Imaging Library).

# Directorios con las imágenes de entrenamiento y prueba
train_dir = 'C:/Users/Labs-DCCO/Downloads/hamburguesas/dataset/train'
test_dir = 'C:/Users/Labs-DCCO/Downloads/hamburguesas/dataset/test'

# Parámetros de la red
input_shape = (150, 150, 3)
num_classes = 2

# Crear el modelo
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(num_classes, activation='softmax'))

# Compilar el modelo
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

# Preprocesamiento de imágenes
train_datagen = ImageDataGenerator(rescale=1.0/255)
```

```
test_datagen = ImageDataGenerator(rescale=1.0/255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=20,
    class_mode='categorical'
)

# Entrenar el modelo
history = model.fit_generator(
    train_generator,
    steps_per_epoch=100,
    epochs=10,
    validation_data=test_generator,
    validation_steps=50
)

# Función para cargar y redimensionar una imagen para hacer predicciones
def load_and_preprocess_image(image_path):
    img = tf.keras.preprocessing.image.load_img(image_path,
target_size=(150, 150))
    img_array = tf.keras.preprocessing.image.img_to_array(img)
    img_array = img_array / 255.0
    img_array = np.expand_dims(img_array, axis=0)
    return img_array

# Ejemplo de predicción
sample_image_path = 'C:/Users/Labs-
DCCO/Downloads/hamburguesas/dataset/examples/example_1.jpg'  # Colocamos
aquí la ruta de la imagen de prueba
sample_image_array = load_and_preprocess_image(sample_image_path)
prediction = model.predict(sample_image_array)

if prediction[0][0] > prediction[0][1]:
    print("Es un celular.")
else:
    print("Es una tablet.")
```

Ejecución

```
100/100 [==============================] - 3s 12ms/step - loss: 0.6946 - accuracy: 0.4500 - val_loss: 0.6795 - val_accuracy: 0.5000
1/1 [==============================] - 0s 101ms/step
Es una tablet.
```

```
100/100 [==============================] - 9s 19ms/step - loss: 0.6888 - accuracy: 0.5000 - val_loss: 1.5546 - val_accuracy: 0.5000
1/1 [==============================] - 1s 1s/step
Es un celular.
```