actividad

October 20, 2024

[ ]:
```
!unzip dataset.zip -d /content/dataset
```

[ ]:
```
!pip install pycocotools
```

```
Requirement already satisfied: pycocotools in /usr/local/lib/python3.10/dist-
packages (2.0.8)
Requirement already satisfied: matplotlib>=2.1.0 in
/usr/local/lib/python3.10/dist-packages (from pycocotools) (3.7.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from pycocotools) (1.26.4)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools)
(1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib>=2.1.0->pycocotools) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools)
(4.54.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools)
(1.4.7)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools)
(24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib>=2.1.0->pycocotools) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools)
(3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib>=2.1.0->pycocotools)
(2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib>=2.1.0->pycocotools) (1.16.0)
```

[2]:
```
import os
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
```

```
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras import Model
import matplotlib.pyplot as plt

# Paso 1: Configurar las rutas del dataset
dataset_dir = '/content/dataset/electronics'
IMG_SIZE = (160, 160)
BATCH_SIZE = 32

# Paso 2: Cargar el dataset con TensorFlow
train_dataset = tf.keras.utils.image_dataset_from_directory(
    os.path.join(dataset_dir, 'train'),
    shuffle=True,
    batch_size=BATCH_SIZE,
    image_size=IMG_SIZE
)

val_dataset = tf.keras.utils.image_dataset_from_directory(
    os.path.join(dataset_dir, 'val'),
    shuffle=True,
    batch_size=BATCH_SIZE,
    image_size=IMG_SIZE
)

test_dataset = tf.keras.utils.image_dataset_from_directory(
    os.path.join(dataset_dir, 'test'),
    shuffle=True,
    batch_size=BATCH_SIZE,
    image_size=IMG_SIZE
)

# Paso 3: Crear y configurar el modelo usando ResNet50
IMG_SHAPE = IMG_SIZE + (3,)

base_model = ResNet50(input_shape=IMG_SHAPE,
                      include_top=False,
                      weights='imagenet')

base_model.trainable = False

global_average_layer = GlobalAveragePooling2D()
prediction_layer = Dense(len(train_dataset.class_names))

inputs = tf.keras.Input(shape=(160, 160, 3))
x = base_model(inputs, training=False)
x = global_average_layer(x)
x = Dropout(0.2)(x)
```

```
outputs = prediction_layer(x)
model = Model(inputs, outputs)

# Compilación del modelo
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.
 ↪Adam(learning_rate=base_learning_rate),
              loss=tf.keras.losses.
 ↪SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.summary()

# Paso 4: Entrenar el modelo
initial_epochs = 10

history = model.fit(
    train_dataset,
    validation_data=val_dataset,
    epochs=initial_epochs
)

# Paso 5: Graficar la precisión y la pérdida
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```

```
Found 2145 files belonging to 3 classes.
Found 370 files belonging to 3 classes.
```

```
Found 1098 files belonging to 3 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736             1s
0us/step
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 160, 160, 3)	0
resnet50 (Functional)	(None, 5, 5, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dropout (Dropout)	(None, 2048)	0
dense (Dense)	(None, 3)	6,147

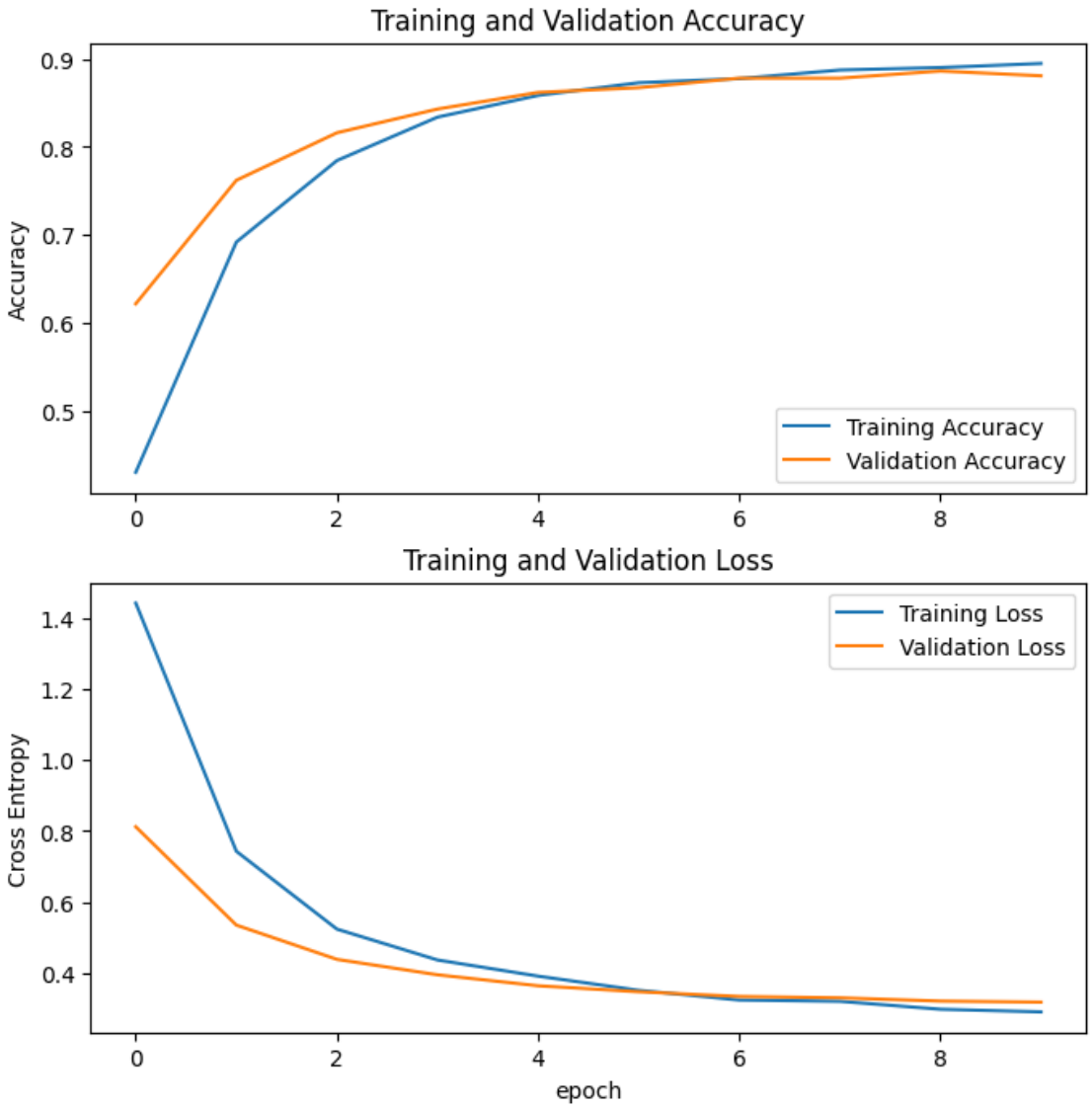
Total params: 23,593,859 (90.00 MB)

Trainable params: 6,147 (24.01 KB)

Non-trainable params: 23,587,712 (89.98 MB)

```
Epoch 1/10
68/68             223s 3s/step -
accuracy: 0.3487 - loss: 1.7621 - val_accuracy: 0.6216 - val_loss: 0.8122
Epoch 2/10
68/68             256s 3s/step -
accuracy: 0.6569 - loss: 0.8157 - val_accuracy: 0.7622 - val_loss: 0.5353
Epoch 3/10
68/68             267s 3s/step -
accuracy: 0.7642 - loss: 0.5808 - val_accuracy: 0.8162 - val_loss: 0.4389
```

```
Epoch 4/10
68/68             211s 3s/step -
accuracy: 0.8242 - loss: 0.4670 - val_accuracy: 0.8432 - val_loss: 0.3954
Epoch 5/10
68/68             271s 3s/step -
accuracy: 0.8526 - loss: 0.4217 - val_accuracy: 0.8622 - val_loss: 0.3645
Epoch 6/10
68/68             261s 3s/step -
accuracy: 0.8612 - loss: 0.3871 - val_accuracy: 0.8676 - val_loss: 0.3476
Epoch 7/10
68/68             261s 3s/step -
accuracy: 0.8665 - loss: 0.3540 - val_accuracy: 0.8784 - val_loss: 0.3343
Epoch 8/10
68/68             253s 3s/step -
accuracy: 0.8895 - loss: 0.3317 - val_accuracy: 0.8784 - val_loss: 0.3306
Epoch 9/10
68/68             274s 3s/step -
accuracy: 0.8903 - loss: 0.3084 - val_accuracy: 0.8865 - val_loss: 0.3213
Epoch 10/10
68/68             253s 3s/step -
accuracy: 0.8933 - loss: 0.3058 - val_accuracy: 0.8811 - val_loss: 0.3185
```



[3]:
```
# Paso 1: Evaluar el modelo en el conjunto de validación
loss, accuracy = model.evaluate(val_dataset)
print('Validation accuracy:', accuracy)

# Paso 2: Inferencia sobre un batch de imágenes del conjunto de validación
image_batch, label_batch = val_dataset.as_numpy_iterator().next()
predictions = model.predict_on_batch(image_batch)

# Aplicar softmax para obtener probabilidades por clase
predictions = tf.nn.softmax(predictions)

# Obtener las clases predichas
```

```
predicted_classes = tf.argmax(predictions, axis=1)

# Obtener los nombres de las clases
class_names = train_dataset.class_names

# Paso 3: Mostrar las imágenes junto con las predicciones y las etiquetas reales
plt.figure(figsize=(10, 10))
for i in range(25):
    ax = plt.subplot(5, 5, i + 1)
    plt.imshow(image_batch[i].astype("uint8"))
    plt.title(f"Pred: {class_names[predicted_classes[i]]}\nTrue:␣
 ↪{class_names[label_batch[i]]}")
    plt.axis("off")
plt.show()
```

```
12/12            30s 3s/step -
accuracy: 0.9067 - loss: 0.2418
Validation accuracy: 0.8810811042785645
```


Pred: monitor
True: mouse
Pred: monitor
True: monitor
Pred: mouse
True: mouse
Pred: monitor
True: monitor
Pred: mouse
True: mouse
Pred: monitor
True: monitor
Pred: mouse
True: monitor
Pred: monitor
True: monitor
Pred: monitor
True: monitor
Pred: monitor
True: keyboard
Pred: monitor
True: monitor
Pred: keyboard
True: keyboard
Pred: mouse
True: mouse
Pred: mouse
True: mouse
Pred: keyboard
True: keyboard
Pred: mouse
True: mouse
Pred: monitor
True: monitor
Pred: monitor
True: monitor
True: mouse
Pred: monitor
True: monitor
Pred: mouse
True: mouse
Pred: monitor
True: monitor
Pred: monitor
True: mouse
Pred: mouse
True: monitor
Pred: monitor
True: monitor
Pred: keyboard
True: keyboard