Prueba técnica

Renato García Morán

Clasificación por transfer learning

Implementación con TensorFlow

- Modelo pre-entrenado utilizado: https://www.kaggle.com/models/google/mobilenet-v2
- Dataset utilizado: https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset
- Precisión obtenida: 93%
- Etiquetas: Covid, Viral Pneumonia, Normal

Instalacion de tf_keras

```
#@title Instalacion de tf_keras
# !pip install tf_keras
```

Importaciones necesarias

```
#@title Importaciones necesarias
import tensorflow as tf
import tf_keras as tfk
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
import tensorflow_hub as hub
import os
import glob
import cv2
import numpy as np
import random
```

Preparación del dataset

```
#@title Preparación del dataset
# https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset
#! unzip archive.zip
```

Definición del dataset

```
#@title Definición del dataset
paths = [
    "Covid19-dataset/train/",
    "Covid19-dataset/test/"
]
```

Definición de las tres etiquetas

```
#@title Definición de las tres etiquetas
possible_labels=os.listdir(paths[0])
possible_labels
```

```
['Normal', 'Viral Pneumonia', 'Covid']
```

Carga de las imágenes ajustando las dimensiones

```
#@title Carga de las imágenes ajustando las dimensiones
data = []

for i, path in enumerate(paths):
    for label_int, label_string in enumerate(possible_labels):
        filenames = glob.glob(path + label_string + "/*.jpg") + glob.glob(path + label_string + "/*.jpeg") + glob.glob(path + label_string +
```

```
        for filename in filenames:
            img = cv2.imread(filename)
            data.append([label_int, cv2.resize(img, (224, 224)), filename])
len(data)
random.Random(0).shuffle(data)
```

Separación

```
#@title Separación

x_train = []
y_train = []

x_val = []
y_val = []

x_test = []
y_test = []

# Separar los datos en entrenamiento, validación y prueba
for i, sample in enumerate(data):
    label = sample[0]
    img = sample[1]
    img_path = sample[2]  # Ruta de la imagen

    if i <= 0.8 * len(data):
        x_train.append(img)
        y_train.append(label)

    elif i > 0.8 * len(data) and i <= 0.9 * len(data):
        x_val.append(img)
        y_val.append(label)

    else:
        x_test.append(img)
        y_test.append(label)
        print(f"Imagen asignada a test: {img_path}")

x_train = np.array(x_train)
x_val = np.array(x_val)
x_test = np.array(x_test)

y_train = np.array(y_train)
y_val = np.array(y_val)
y_test = np.array(y_test)
```

```
Imagen asignada a test: Covid19-dataset/train/Viral Pneumonia/096.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/076.jpg
Imagen asignada a test: Covid19-dataset/train/Covid/058.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/COVID-00020.jpg
Imagen asignada a test: Covid19-dataset/train/Normal/071.jpeg
Imagen asignada a test: Covid19-dataset/test/Viral Pneumonia/0112.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/065.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/COVID-00010.jpg
Imagen asignada a test: Covid19-dataset/train/Normal/012.jpeg
Imagen asignada a test: Covid19-dataset/train/Normal/06.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/091.jpg
Imagen asignada a test: Covid19-dataset/train/Viral Pneumonia/024.jpeg
Imagen asignada a test: Covid19-dataset/test/Viral Pneumonia/0109.jpeg
Imagen asignada a test: Covid19-dataset/train/Viral Pneumonia/021.jpeg
Imagen asignada a test: Covid19-dataset/train/Normal/094.jpeg
Imagen asignada a test: Covid19-dataset/test/Covid/0106.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/074.jpg
Imagen asignada a test: Covid19-dataset/train/Viral Pneumonia/033.jpeg
Imagen asignada a test: Covid19-dataset/test/Normal/0118.jpeg
Imagen asignada a test: Covid19-dataset/train/Viral Pneumonia/047.jpeg
Imagen asignada a test: Covid19-dataset/test/Covid/radiopaedia-2019-novel-coronavirus-infected-pneumonia.jpg
Imagen asignada a test: Covid19-dataset/train/Covid/COVID-00029.jpg
Imagen asignada a test: Covid19-dataset/train/Covid/04.png
Imagen asignada a test: Covid19-dataset/train/Covid/086.jpg
Imagen asignada a test: Covid19-dataset/train/Covid/051.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/COVID-00015a.png
Imagen asignada a test: Covid19-dataset/test/Normal/0108.jpeg
Imagen asignada a test: Covid19-dataset/train/Viral Pneumonia/010.jpeg
Imagen asignada a test: Covid19-dataset/train/Normal/073.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/031.jpeg
Imagen asignada a test: Covid19-dataset/train/Covid/08.jpeg
```

Definición del modelo preentrenado

```
#@title Definición del modelo preentrenado
import kagglehub

path_model = kagglehub.model_download("google/mobilenet-v2/tensorFlow2/100-224-feature-vector")
```

Definicion de la estructura de la red

```
#@title Definicion de la estructura de la red
def transferLearning_model():
    model = tfk.Sequential()
    model.add(hub.KerasLayer(path_model, trainable=False))  # Modelo preentrenado
    model.add(tfk.layers.Dense(30, activation='relu'))
    model.add(tfk.layers.Dense(3, activation='softmax'))  # Tenemos 3 clases
    model.compile(optimizer='SGD', loss='categorical_crossentropy', metrics=['accuracy'])
    model.build([None, 224, 224, 3])  # Definimos las dimensiones de la imagen en forma de tensor
    return model
```

Creamos la estructura de la red neuronal, con la diferencia de que añadimos el modelo entrenado como si fuera una capa más en lugar de añadir capas de convolución y definimos que no queremos dejar fijos sus parámetros.

Inicializamos el modelo

```
#@title Inicializamos el modelo
model = transferLearning_model()
```

Listamos las caracteristicas

```
#@title Listamos las caracteristicas
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_3 (KerasLayer)  (None, 1280)              2257984

 dense_6 (Dense)             (None, 30)                38430

 dense_7 (Dense)             (None, 3)                 93

=================================================================
Total params: 2296507 (8.76 MB)
Trainable params: 38523 (150.48 KB)
Non-trainable params: 2257984 (8.61 MB)
_________________________________________________________________
```

Tenemos 1280 features

```
y_trainOneHot=tf.one_hot(y_train,len(possible_labels))
y_valOneHot=tf.one_hot(y_val,len(possible_labels))
y_testOneHot=tf.one_hot(y_test,len(possible_labels))


history = model.fit(x_train,
          y_trainOneHot,
          epochs=40,
          batch_size=100,
          validation_data=(x_val,y_valOneHot),
          )
```

```
Epoch 17/40
3/3 [==============================] - 0s 131ms/step - loss: 0.4870 - accuracy: 0.8425 - val_loss: 0.7224 - val_accuracy: 0.7500
Epoch 18/40
3/3 [==============================] - 0s 125ms/step - loss: 0.4847 - accuracy: 0.8346 - val_loss: 0.6956 - val_accuracy: 0.7500
Epoch 19/40
3/3 [==============================] - 0s 123ms/step - loss: 0.4651 - accuracy: 0.8386 - val_loss: 0.6687 - val_accuracy: 0.7500
Epoch 20/40
3/3 [==============================] - 0s 122ms/step - loss: 0.4550 - accuracy: 0.8504 - val_loss: 0.7879 - val_accuracy: 0.6250
Epoch 21/40
3/3 [==============================] - 0s 136ms/step - loss: 0.4600 - accuracy: 0.8268 - val_loss: 0.6347 - val_accuracy: 0.7500
Epoch 22/40
3/3 [==============================] - 0s 122ms/step - loss: 0.4178 - accuracy: 0.8740 - val_loss: 0.5987 - val_accuracy: 0.7812
Epoch 23/40
3/3 [==============================] - 0s 118ms/step - loss: 0.4174 - accuracy: 0.8622 - val_loss: 0.6131 - val_accuracy: 0.7500
Epoch 24/40
3/3 [==============================] - 0s 112ms/step - loss: 0.4205 - accuracy: 0.8661 - val_loss: 0.7856 - val_accuracy: 0.6875
Epoch 25/40
3/3 [==============================] - 0s 109ms/step - loss: 0.4750 - accuracy: 0.8386 - val_loss: 0.5850 - val_accuracy: 0.8125
Epoch 26/40
3/3 [==============================] - 0s 109ms/step - loss: 0.3937 - accuracy: 0.8780 - val_loss: 0.6851 - val_accuracy: 0.6875
Epoch 27/40
3/3 [==============================] - 0s 109ms/step - loss: 0.3908 - accuracy: 0.8740 - val_loss: 0.5990 - val_accuracy: 0.7812
Epoch 28/40
3/3 [==============================] - 0s 110ms/step - loss: 0.3797 - accuracy: 0.8858 - val_loss: 0.7250 - val_accuracy: 0.6562
Epoch 29/40
3/3 [==============================] - 0s 109ms/step - loss: 0.3774 - accuracy: 0.8543 - val_loss: 0.5814 - val_accuracy: 0.8438
Epoch 30/40
3/3 [==============================] - 0s 108ms/step - loss: 0.3801 - accuracy: 0.8622 - val_loss: 0.7066 - val_accuracy: 0.6875
Epoch 31/40
3/3 [==============================] - 0s 129ms/step - loss: 0.4248 - accuracy: 0.8425 - val_loss: 0.5390 - val_accuracy: 0.8438
Epoch 32/40
3/3 [==============================] - 0s 109ms/step - loss: 0.3421 - accuracy: 0.9094 - val_loss: 0.5613 - val_accuracy: 0.8125
Epoch 33/40
3/3 [==============================] - 0s 110ms/step - loss: 0.3548 - accuracy: 0.8701 - val_loss: 0.5776 - val_accuracy: 0.7812
Epoch 34/40
3/3 [==============================] - 0s 112ms/step - loss: 0.3626 - accuracy: 0.8622 - val_loss: 0.5243 - val_accuracy: 0.8438
Epoch 35/40
3/3 [==============================] - 0s 110ms/step - loss: 0.3306 - accuracy: 0.9055 - val_loss: 0.6027 - val_accuracy: 0.7500
Epoch 36/40
3/3 [==============================] - 0s 110ms/step - loss: 0.3335 - accuracy: 0.8898 - val_loss: 0.5540 - val_accuracy: 0.8125
Epoch 37/40
3/3 [==============================] - 0s 109ms/step - loss: 0.3177 - accuracy: 0.8976 - val_loss: 0.5732 - val_accuracy: 0.7500
Epoch 38/40
3/3 [==============================] - 0s 113ms/step - loss: 0.3215 - accuracy: 0.9055 - val_loss: 0.6038 - val_accuracy: 0.7188
Epoch 39/40
3/3 [==============================] - 0s 112ms/step - loss: 0.3088 - accuracy: 0.9134 - val_loss: 0.5104 - val_accuracy: 0.8438
Epoch 40/40
3/3 [==============================] - 0s 122ms/step - loss: 0.3483 - accuracy: 0.8543 - val_loss: 0.5340 - val_accuracy: 0.8438
```

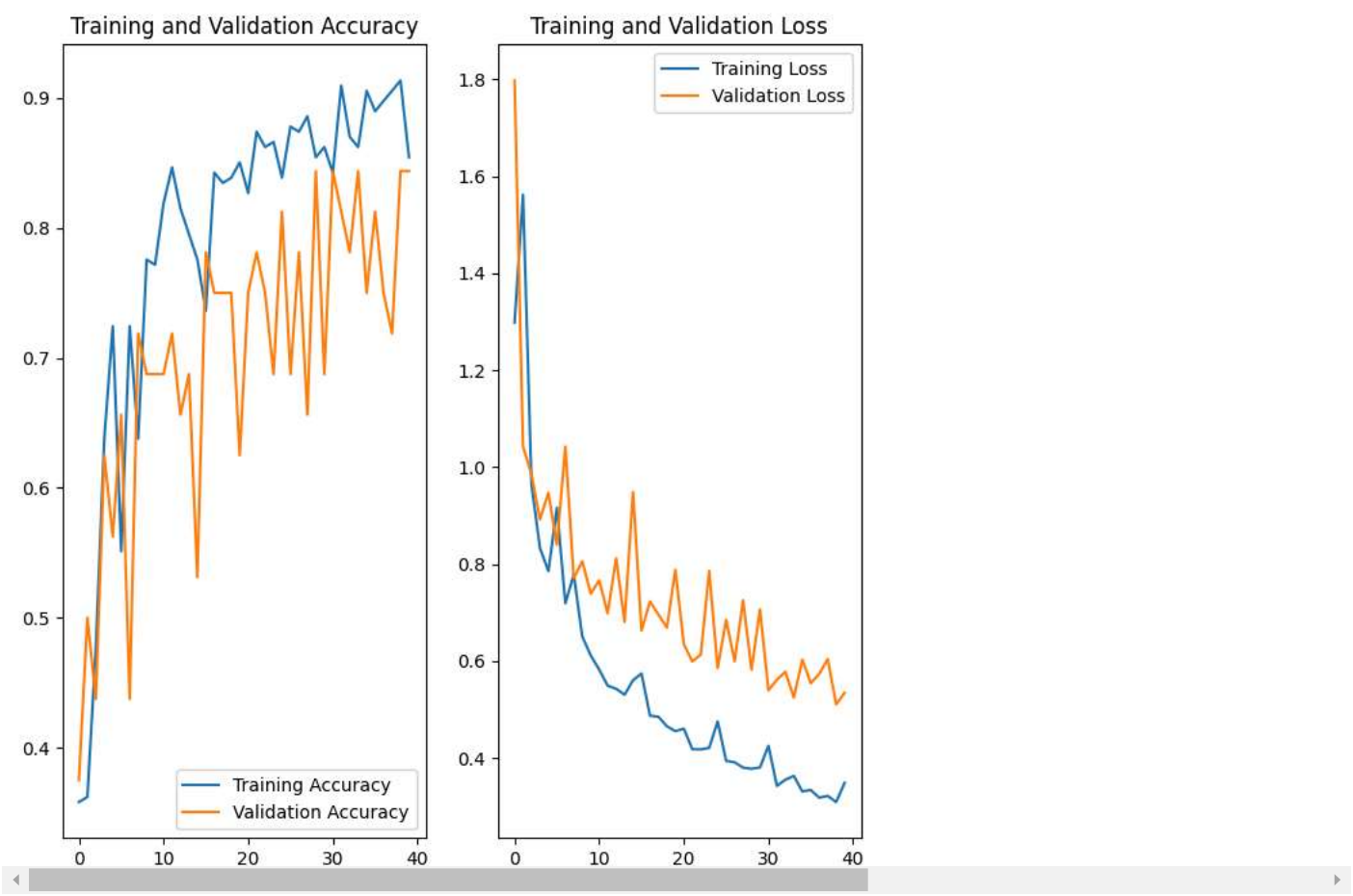
```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(len(acc))

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

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



Test individual imagen pneumonia

Mostrar código

```
1/1 [==============================] - 0s 61ms/step
La imagen tiene una probabilidad de 22.56% de ser COVID-19
La imagen tiene una probabilidad de 55.55% de ser pneumonia
La imagen tiene una probabilidad de 21.88% de ser normal
Esta imagen probablemente pertenece a pneumonia con una confianza del 55.55%.
```

Evaluamos el modelo con las imagenes de prueba para ver la efectividad

```
#@title Evaluamos el modelo con las imagenes de prueba para ver la efectividad
model.evaluate(x=x_test,y=y_testOneHot)
```

```
1/1 [==============================] - 0s 70ms/step - loss: 0.3708 - accuracy: 0.9355
[0.370831698179245, 0.9354838728904724]
```

Guardamos el modelo

```
#@title Guardamos el modelo
model.save('model.keras')


predicciones = model.predict(x_test)
```

```
1/1 [==============================] - 0s 22ms/step
```

Matriz de confusion

```
#@title Matriz de confusion
import numpy as np
from tensorflow.keras.utils import to_categorical
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

```
y_testOneHot = to_categorical(y_test, num_classes=len(possible_labels))

predicciones_clase = np.argmax(predicciones, axis=1)

unique, counts = np.unique(y_test, return_counts=True)
print(f"Distribución de etiquetas en el conjunto de prueba: {dict(zip(unique, counts))}")

for i, (pred, real) in enumerate(zip(predicciones_clase, y_test)):
    print(f"Imagen {i}: Predicción -> {possible_labels[pred]}, Real -> {possible_labels[real]}")

cm = confusion_matrix(y_test, predicciones_clase)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=possible_labels, yticklabels=possible_labels)
plt.ylabel('Etiqueta Real')
plt.xlabel('Etiqueta Predicha')
plt.title('Matriz de Confusión')
plt.show()

print(classification_report(y_test, predicciones_clase, target_names=possible_labels))

resultado = model.evaluate(x=x_test, y=y_testOneHot)
print(f"Resultado de evaluación del modelo: {resultado}")
```

```
Imagen 0: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 1: Predicción -> Covid, Real -> Covid
Imagen 2: Predicción -> Covid, Real -> Covid
Imagen 3: Predicción -> Viral Pneumonia, Real -> Covid
Imagen 4: Predicción -> Normal, Real -> Normal
Imagen 5: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 6: Predicción -> Covid, Real -> Covid
Imagen 7: Predicción -> Covid, Real -> Covid
Imagen 8: Predicción -> Normal, Real -> Normal
Imagen 9: Predicción -> Normal, Real -> Normal
Imagen 10: Predicción -> Covid, Real -> Covid
Imagen 11: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 12: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 13: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 14: Predicción -> Covid, Real -> Normal
Imagen 15: Predicción -> Covid, Real -> Covid
Imagen 16: Predicción -> Covid, Real -> Covid
Imagen 17: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 18: Predicción -> Normal, Real -> Normal
Imagen 19: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
Imagen 20: Predicción -> Covid, Real -> Covid
Imagen 21: Predicción -> Covid, Real -> Covid
Imagen 22: Predicción -> Covid, Real -> Covid
Imagen 23: Predicción -> Covid, Real -> Covid
Imagen 24: Predicción -> Covid, Real -> Covid
Imagen 25: Predicción -> Covid, Real -> Covid
Imagen 26: Predicción -> Normal, Real -> Normal
Imagen 27: Predicción -> Viral Pneumonia, Real -> Viral Pneumonia
```