Informe 2ª práctica: redes neuronales

Raúl Cruz Ortega y Laura González Suárez

El objetivo de esta práctica es desarrollar una red neuronal para la clasificación de imágenes, en concreto hemos seleccionado la clasificación de enfermedades de las hojas de la planta del tomate.

Información del dataset: en la página de repositorios *kaggle* hemos seleccionado el dataset *"Tomato Leaves Dataset"*[1], que cuenta con 11 categorías de enfermedades de la hoja del tomate. Debido a que cada categoría contaba con demasiadas imágenes, decidimos recortar el dataset e incluir únicamente 1000 imágenes por categoría. Inspeccionando las imágenes del repositorio hemos observado que los creadores utilizaron la técnica "data augmentation" para obtener más muestras, debido a que muchas imágenes eran iguales pero con pequeños cambios en la iluminación, fondo y orientación.

Ejecuciones de las pruebas con la red neuronal:

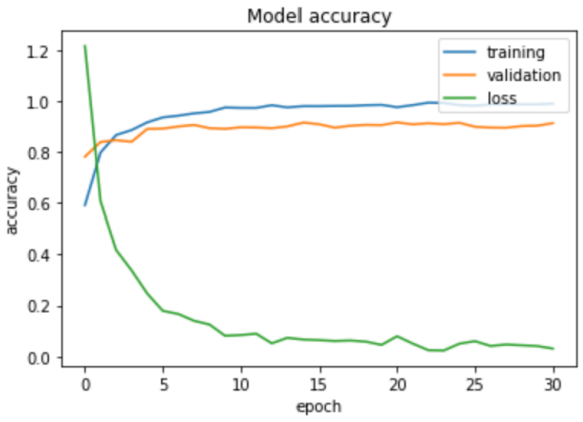
Con el objetivo de obtener la mejor precisión posible hemos modificado los siguientes hiperparámetros:

- **Número de capas convolutivas**: número de filtros que se aplicarán a la entrada. A medida que se va bajando la resolución de la foto, aumenta el número de filtros que utilizamos, ya que seremos capaces de centrarnos en los valores importantes de la foto y no en los detalles
- **Max Pooling**: operación que calcula el valor máximo para una matriz definida de píxeles, gracias a ella podemos descartar algunos datos (detalles) de la foto. En la práctica somos capaces de variar el tamaño de la matriz.
- **Dropout:** representa el porcentaje de neuronas que se deshabilitan aleatoriamente. Su valor oscila entre el 0,1 y 0,5 y es la mejor forma de solucionar el overfitting, ya que se promedian las predicciones de varias redes neuronales entrenadas separadas.

A continuación, se muestran los valores de precisión y la gráfica obtenida para las diferentes combinaciones de hiperparámetros que hemos probado.

Ejecución 1: Capas convolutivas=[32, 64, 128], mp=2 y dropout=0.1

```
# MODEL ----------------------------------------------------
model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.1))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```

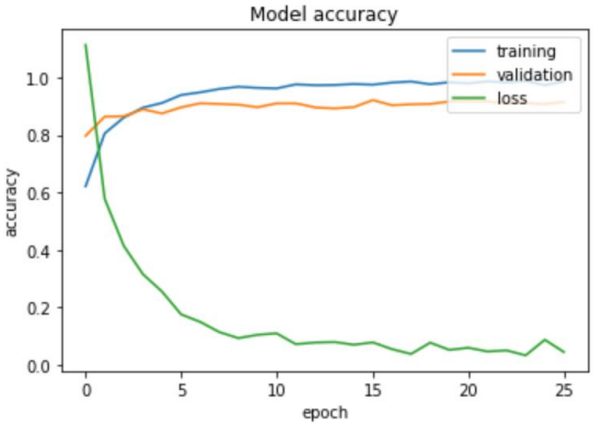



```
223/224 [============================>.] - ETA: 0s - loss: 0.0307 - accuracy: 0.9892Restoring model weights from the end of the best epoch: 21.
224/224 [==============================] - 21s 90ms/step - loss: 0.0307 - accuracy: 0.9892 - val_loss: 0.4853 - val_accuracy: 0.9124
Epoch 31: early stopping
```

Precisión: 0,9124

Ejecución 2: Capas convolutivas=[32, 64, 128], mp=2 y dropout=0.2

```
# MODEL -----------------------------------------------------
model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```

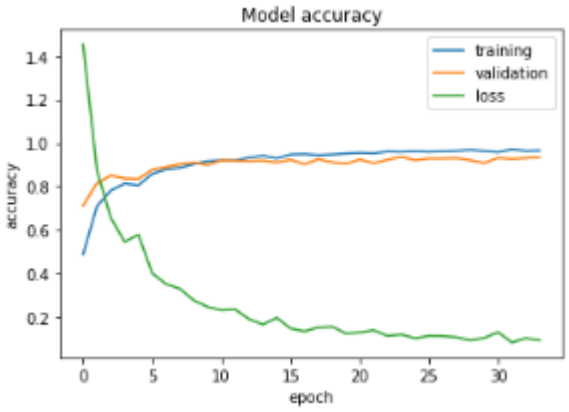



```
223/224 [============================>.] - ETA: 0s - loss: 0.0441 - accuracy: 0.9870Restoring model weights from the end of the best epoch: 16.
224/224 [==============================] - 21s 90ms/step - loss: 0.0441 - accuracy: 0.9870 - val_loss: 0.4475 - val_accuracy: 0.9157
Epoch 26: early stopping
```

Precisión: 0,9157

Ejecución 3: Capas convolutivas=[32, 64, 128], mp=2 y dropout=0.25 y 0.5

```
# MODEL -----------------------------------------------------

model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```

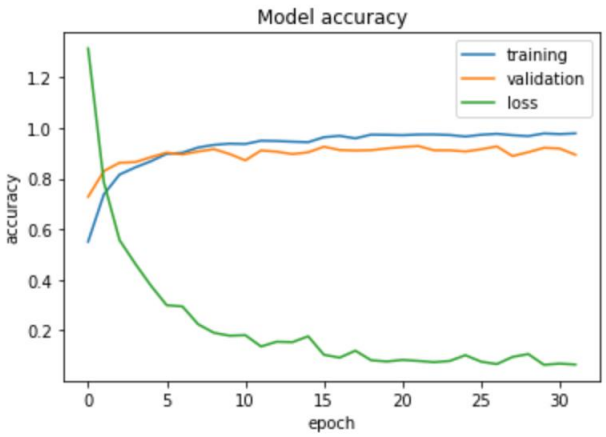



```
Epoch 34/200
223/224 [============================>.] - ETA: 0s - loss: 0.0935 - accuracy: 0.9658Restoring model weights from the end of the best epoch: 24.
224/224 [==============================] - 20s 89ms/step - loss: 0.0934 - accuracy: 0.9658 - val_loss: 0.3275 - val_accuracy: 0.9343
Epoch 34: early stopping
```

Precisión: 0,9343

Ejecución 4: Capas convolutivas=[32, 64, 128], mp=2 y dropout=0.3

```
# MODEL -------------------------------------------------
model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```

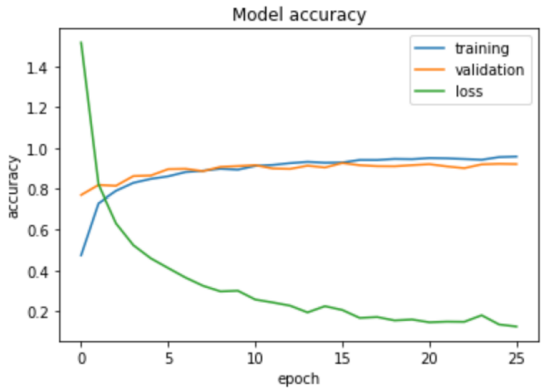



```
223/224 [============================>.] - ETA: 0s - loss: 0.0656 - accuracy: 0.9781Restoring model weights from the end of the best epoch: 22.
224/224 [==============================] - 20s 89ms/step - loss: 0.0656 - accuracy: 0.9782 - val_loss: 0.5500 - val_accuracy: 0.8938
Epoch 32: early stopping
```

Precisión: 0,8938

Ejecución 5: Capas convolutivas=[32, 64, 128], mp=2 y dropout=0.4

```
# MODEL -------------------------------------------------
model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```

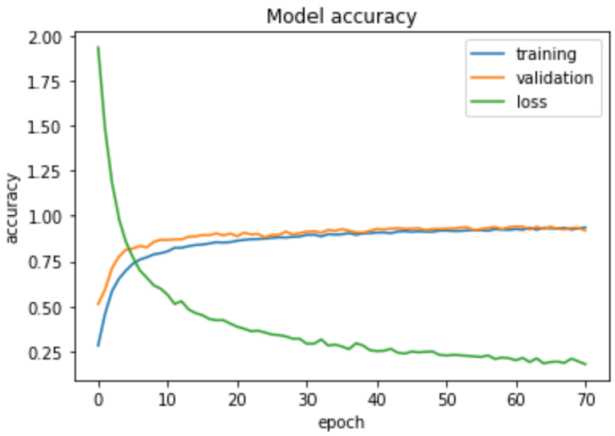



```
223/224 [============================>.] - ETA: 0s - loss: 0.1253 - accuracy: 0.9568Restoring model weights from the end of the best epoch: 16.
224/224 [==============================] - 20s 90ms/step - loss: 0.1252 - accuracy: 0.9569 - val_loss: 0.3148 - val_accuracy: 0.9199
Epoch 26: early stopping
```

Precisión: 0,9199

Ejecución 6: Capas convolutivas=[32, 64, 128], mp=4 y dropout=0.4

```
# MODEL -----------------------------------------------------
model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Dropout(0.4))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Dropout(0.4))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```

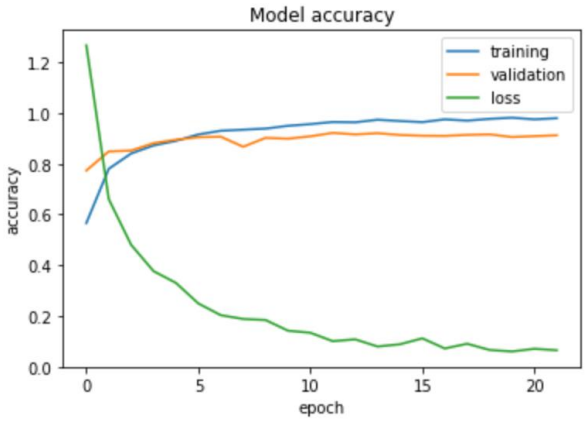



```
221/224 [============================>.] - ETA: 0s - loss: 0.1789 - accuracy: 0.9385Restoring model weights from the end of the best epoch: 61.
224/224 [==============================] - 20s 87ms/step - loss: 0.1796 - accuracy: 0.9385 - val_loss: 0.2164 - val_accuracy: 0.9212
Epoch 71: early stopping
```

Precisión: 0,9212

Ejecución 7: Capas convolutivas=[64, 128, 256], mp=2 y dropout=0.25

```
# MODEL -----------------------------------------------------
model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```




```
223/224 [============================>.] - ETA: 0s - loss: 0.0644 - accuracy: 0.9794Restoring model weights from the end of the best epoch: 12.
224/224 [==============================] - 22s 97ms/step - loss: 0.0644 - accuracy: 0.9794 - val_loss: 0.4522 - val_accuracy: 0.9121
Epoch 22: early stopping
```

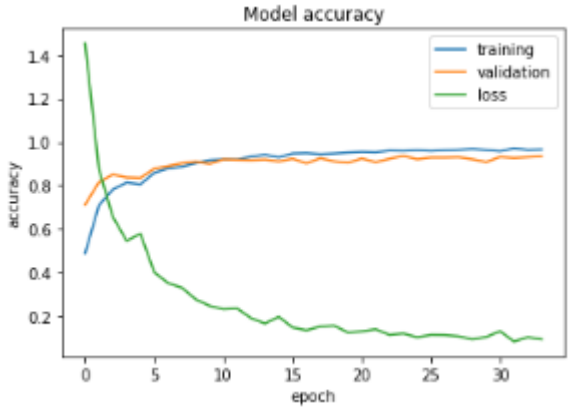
Precisión: 0,9121

Mejor modelo probado:

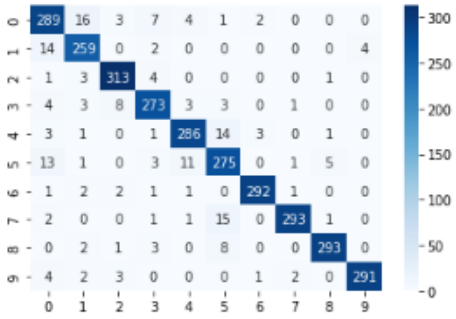
Hemos seleccionado la ejecución 3, debido a que es la de mayor precisión.

Parámetros: Capas convolutivas=[32, 64, 128], mp=2 y dropout=0.25 y 0.5

```
# MODEL ----------------------------------------------------

model = keras.Sequential()
model.add(Rescaling(scale=(1./127.5),
                    offset=-1,
                    input_shape=(150, 150, 3)))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

model.compile(loss=tf.keras.losses.categorical_crossentropy,
              optimizer=tf.keras.optimizers.Adam(1e-3),
              metrics=['accuracy'])
```



	precision	recall	f1-score	support
0	0.8731	0.8975	0.8851	322
1	0.8962	0.9283	0.9120	279
2	0.9485	0.9720	0.9601	322
3	0.9254	0.9254	0.9254	295
4	0.9346	0.9256	0.9301	309
5	0.8703	0.8900	0.8800	309
6	0.9799	0.9733	0.9766	300
7	0.9832	0.9361	0.9591	313
8	0.9734	0.9544	0.9638	307
9	0.9864	0.9604	0.9732	303
accuracy			0.9363	3059
macro avg	0.9371	0.9363	0.9365	3059
weighted avg	0.9372	0.9363	0.9365	3059



```
Epoch 34/200
223/224 [============================>.] - ETA: 0s - loss: 0.0935 - accuracy: 0.9658Restoring model weights from the end of the best epoch: 24.
224/224 [==============================] - 20s 89ms/step - loss: 0.0934 - accuracy: 0.9658 - val_loss: 0.3275 - val_accuracy: 0.9343
Epoch 34: early stopping
```

Transfer learning:

Transfer Learning es un método de aprendizaje automático en el que un modelo desarrollado para una tarea se reutiliza como punto de partida para el desarrollo de una segunda tarea. Concretamente, al utilizar esta técnica utilizamos las capas convolutivas de una red previamente entrenada de *"Imagenet"* y utilizamos sus pesos para nuestro modelo. Con el objetivo de utilizar esta técnica hemos utilizado el modelo VGG16 de la librería Keras.

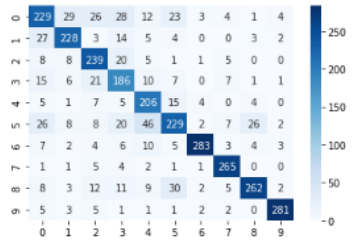
Ejecución:

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np
base_model = keras.applications.VGG16(
    weights='imagenet',  # Load weights pre-trained on ImageNet.
    input_shape=(150, 150, 3),
    include_top=False)  # Do not include the ImageNet classifier at the top.

base_model.trainable = False

inputs = keras.Input(shape=(150, 150, 3))

x = base_model(inputs, training=False)

x = keras.layers.GlobalAveragePooling2D()(x)

# Convert features of shape `base_model.output_shape[1:]` to vectors
x = keras.layers.Dense(256, activation='relu')(x)
outputs = keras.layers.Dense(10, activation='sigmoid')(x)

model = keras.Model(inputs, outputs)
model.compile(optimizer=keras.optimizers.Adam(),
              loss=keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=[keras.metrics.BinaryAccuracy()])

epochs = 1

es = EarlyStopping(monitor='val_binary_accuracy', mode='max', verbose=1, patience=10, restore_best_weights=True)

h = model.fit(
        train_ds,
        epochs=epochs,
        validation_data=val_ds,
        callbacks = [es]
)
```

```
224/224 [==============================] - 30s 123ms/step - loss: 0.2106 - binary_accuracy: 0.9331 - val_loss: 0.1265 - val_binary_accuracy: 0.9539
```

Precisión: 0,9539

	precision	recall	f1-score	support
0	0.6918	0.6379	0.6638	359
1	0.7889	0.7972	0.7930	286
2	0.7242	0.8328	0.7747	287
3	0.6305	0.7323	0.6776	254
4	0.6732	0.8340	0.7450	247
5	0.7247	0.6123	0.6638	374
6	0.9497	0.8654	0.9056	327
7	0.8893	0.9464	0.9170	280
8	0.8704	0.7616	0.8124	344
9	0.9525	0.9336	0.9430	301
accuracy			0.7872	3059
macro avg	0.7895	0.7953	0.7896	3059
weighted avg	0.7927	0.7872	0.7872	3059



Gracias a la técnica del transfer learning, hemos aumentado la precisión de nuestra red neuronal casi un 2%.

1. Tomato Leaves Dataset: https://www.kaggle.com/datasets/ashishmotwani/tomato Kaggle [en línea]