In [1]:
```
# pip install --upgrade keras
```

Clonamos el repositorio para obtener el dataset

In [2]:
```
# !git clone https://github.com/joanby/deeplearning-az.git
```

In [3]:
```
# from google.colab import drive
# drive.mount('/content/drive')

train_ds_path  ='C:/Users/Usuario/Documents/Master/Aprendizaje Profundo/Udemy/
test_ds_path   ='C:/Users/Usuario/Documents/Master/Aprendizaje Profundo/Udemy/
cat_or_dog_path='C:/Users/Usuario/Documents/Master/Aprendizaje Profundo/Udemy/
```

Parte 1 - Construir el modelo de CNN

Importar las librerías y paquetes

In [4]:
```
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense

"""
Necesitamos la capa de olvido para eviter el sobre-entrenamiento
"""
from keras.layers import Dropout
```

Inicializar la CNN

In [5]:
```
classifier = Sequential()
```

Paso 1 - Convolución

In [6]:
```
"""
frame_size = (  ,  )

esta dupla nos permitirá parametrizar la resolución
de entrada de las imágenes
"""
frame_size = (96, 96)

classifier.add(Conv2D(filters = 32,kernel_size = (3, 3),
                      input_shape = (*frame_size, 3), activation = "relu"))
```

Paso 2 - Max Pooling

In [7]:
```
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

Una segunda capa de convolución y max pooling

In [8]:
```
classifier.add(Conv2D(filters = 32,kernel_size = (3, 3), activation = "relu"))
```

In [9]:
```
classifier.add(MaxPooling2D(pool_size = (2,2)))
```

Paso 3 - Flattening

In [10]:
```
classifier.add(Flatten())
```

Paso 4 - Full Connection

In [11]:
```
classifier.add(Dense(units = 128, activation = "relu"))
"""
classifier.add(Dropout(  ))

Añadimos esta capa de olvido para evitar el
sobre-entrenamiento que hemos detectado en la
versión anterior
"""
classifier.add(Dropout(0.5))

classifier.add(Dense(units = 1, activation = "sigmoid"))
```

Compilar la CNN

Como va a ser entrenada?

In [12]:
```
classifier.compile(optimizer = "adam",
                   loss = "binary_crossentropy",
                   metrics = ["accuracy"])
```

Parte 2 - Ajustar la CNN a las imágenes para entrenar

In [13]:

```
from keras.preprocessing.image import ImageDataGenerator

"""
batch_size=32

32 es el valor por defecto que usaría
model.fit_generator, aunque aquí lo debemos
especificar en los generadores en lugar de
en la llamada al bucle de entrenamiento
"""
batch_size=32

train_datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

training_dataset = train_datagen.flow_from_directory(train_ds_path,
                                                     target_size=frame_size,
                                                     batch_size=batch_size,
                                                     class_mode='binary')

testing_dataset = test_datagen.flow_from_directory(test_ds_path,
                                                   target_size=frame_size,
                                                   batch_size=batch_size,
                                                   class_mode='binary')
"""
steps_per_epoch=8000/batchsize
validation_steps=2000/batch_size

a partir de keras 2.2.0 la función
fit_generator queda en proceso de obsolencencia
(programada ;-) pero sigue funcionando si
ajustamos el número de steps, dividiéndolo
por el tamaño de lote
"""
"""
workers=4

si pedimos más de un proceso para el generador de
imágenes, el rendimiento de las pruebas mejora un poco
"""

classifier.fit_generator(generator=training_dataset,
                         steps_per_epoch=8000/batch_size,
                         epochs=25,
                         validation_data=testing_dataset,
                         validation_steps=2000/batch_size,
                         workers=4)
```

```
Found 8000 images belonging to 2 classes.
Found 2000 images belonging to 2 classes.
C:\Users\Usuario\anaconda3\lib\site-packages\tensorflow\python\keras\engine\tr
aining.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be r
```

```
emoved in a future version. Please use `Model.fit`, which supports generators.
Epoch 1/25
250/250 [==============================] - 54s 207ms/step - loss: 0.6911 - acc
uracy: 0.5565 - val_loss: 0.6517 - val_accuracy: 0.6160
Epoch 2/25
250/250 [==============================] - 51s 202ms/step - loss: 0.6400 - acc
uracy: 0.6364 - val_loss: 0.6526 - val_accuracy: 0.6400
Epoch 3/25
250/250 [==============================] - 52s 206ms/step - loss: 0.5984 - acc
uracy: 0.6782 - val_loss: 0.5599 - val_accuracy: 0.7240
Epoch 4/25
250/250 [==============================] - 49s 196ms/step - loss: 0.5591 - acc
uracy: 0.7122 - val_loss: 0.5411 - val_accuracy: 0.7300
Epoch 5/25
250/250 [==============================] - 50s 198ms/step - loss: 0.5437 - acc
uracy: 0.7237 - val_loss: 0.5422 - val_accuracy: 0.7335
Epoch 6/25
250/250 [==============================] - 51s 204ms/step - loss: 0.5214 - acc
uracy: 0.7396 - val_loss: 0.5057 - val_accuracy: 0.7610
Epoch 7/25
250/250 [==============================] - 51s 204ms/step - loss: 0.5075 - acc
uracy: 0.7449 - val_loss: 0.4817 - val_accuracy: 0.7685
Epoch 8/25
250/250 [==============================] - 52s 205ms/step - loss: 0.4884 - acc
uracy: 0.7663 - val_loss: 0.4728 - val_accuracy: 0.7805
Epoch 9/25
250/250 [==============================] - 52s 205ms/step - loss: 0.4625 - acc
uracy: 0.7860 - val_loss: 0.4794 - val_accuracy: 0.7770
Epoch 10/25
250/250 [==============================] - 51s 204ms/step - loss: 0.4537 - acc
uracy: 0.7879 - val_loss: 0.4795 - val_accuracy: 0.7625
Epoch 11/25
250/250 [==============================] - 51s 201ms/step - loss: 0.4430 - acc
uracy: 0.7945 - val_loss: 0.4789 - val_accuracy: 0.7725
Epoch 12/25
250/250 [==============================] - 51s 202ms/step - loss: 0.4325 - acc
uracy: 0.7916 - val_loss: 0.4886 - val_accuracy: 0.7775
Epoch 13/25
250/250 [==============================] - 52s 205ms/step - loss: 0.4260 - acc
uracy: 0.8046 - val_loss: 0.4675 - val_accuracy: 0.7840
Epoch 14/25
250/250 [==============================] - 50s 200ms/step - loss: 0.4124 - acc
uracy: 0.8119 - val_loss: 0.4758 - val_accuracy: 0.7690
Epoch 15/25
250/250 [==============================] - 51s 201ms/step - loss: 0.4226 - acc
uracy: 0.8084 - val_loss: 0.4378 - val_accuracy: 0.7980
Epoch 16/25
250/250 [==============================] - 52s 207ms/step - loss: 0.3971 - acc
uracy: 0.8220 - val_loss: 0.4670 - val_accuracy: 0.7905
Epoch 17/25
250/250 [==============================] - 51s 203ms/step - loss: 0.3982 - acc
uracy: 0.8246 - val_loss: 0.4691 - val_accuracy: 0.7880
Epoch 18/25
250/250 [==============================] - 52s 205ms/step - loss: 0.3842 - acc
uracy: 0.8223 - val_loss: 0.4831 - val_accuracy: 0.7975
Epoch 19/25
250/250 [==============================] - 52s 205ms/step - loss: 0.3854 - acc
uracy: 0.8216 - val_loss: 0.4579 - val_accuracy: 0.8045
Epoch 20/25
250/250 [==============================] - 52s 208ms/step - loss: 0.3628 - acc
uracy: 0.8461 - val_loss: 0.4684 - val_accuracy: 0.7900
Epoch 21/25
250/250 [==============================] - 51s 203ms/step - loss: 0.3634 - acc
uracy: 0.8340 - val_loss: 0.4424 - val_accuracy: 0.8120
Epoch 22/25
```

```
250/250 [==============================] - 52s 206ms/step - loss: 0.3500 - acc
uracy: 0.8450 - val_loss: 0.4514 - val_accuracy: 0.8170ccuracy: 0.84
Epoch 23/25
250/250 [==============================] - 51s 202ms/step - loss: 0.3387 - acc
uracy: 0.8525 - val_loss: 0.4704 - val_accuracy: 0.8040
Epoch 24/25
250/250 [==============================] - 52s 205ms/step - loss: 0.3247 - acc
uracy: 0.8583 - val_loss: 0.4691 - val_accuracy: 0.8090
Epoch 25/25
250/250 [==============================] - 52s 205ms/step - loss: 0.3315 - acc
uracy: 0.8529 - val_loss: 0.5011 - val_accuracy: 0.7880
```

Out[13]: <tensorflow.python.keras.callbacks.History at 0x1c5fe045fd0>

Comentario a los resultados y propuesta de mejora

Resultado

Arquitectura original

Epochs: 25, loss: 0.2710 - **accuracy: 0.8874** - val_loss: 0.5012 - **val_accuracy: 0.80**

Arquitectura 2 [añadida capa Dropout(0.2)]

Epochs: 25, loss: 0.3350 - **accuracy: 0.8510** - val_loss: 0.4406 - **val_accuracy: 0.8105**

Arquitectura 3 [cambio a 96x96 píxeles y Dropout(0.3)]

Epochs: 25, loss: 0.2813 - **accuracy: 0.8836** - val_loss: 0.6156 - **val_accuracy: 0.7655**

Arquitectura 4 [96x96 píxeles y Dropout(0.5)]

Epochs: 25, loss: 0.3315 - **accuracy: 0.8529** - val_loss: 0.5011 - **val_accuracy: 0.7880**

Comentario

Con el aumento del parámetro de olvido hasta 0.5, la divergencia entre las precisiones de entrenamiento y validación ha mejorado. Si la evolución descendente hasta la epoch 25 de la función de pérdida sugiere que aumentemos el número de epoch de entrenamiento.

Propuesta de mejora

Una nueva ejecución con 100 epochs debería ser suficiente para confirmar o descartar esta propuesta.

Parte 3 - Cómo hacer nuevas predicciones

In [14]:

```
import numpy as np
from keras.preprocessing import image
test_image = image.load_img(cat_or_dog_path, target_size = frame_size)
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = classifier.predict(test_image)
training_dataset.class_indices
if result[0][0] == 1:
    prediction = 'dog'
else:
    prediction = 'cat'

print(prediction)
```

```
dog
```