# Primera ejecución del modelo CNN simple

## Cambios vs. Version anterior:

* 1. Se han añadido una capa Dense y otra capa de Dropout tras esta. Ambas han sido incluidas a continuación de las Dense-Dropout preexistentes.
  2. En este caso, la nueva capa Dense tiene 64 neuronas en lugar de 128.
  3. Se conserva el ratio de Dropout en un 30%.

## Estructura:

* 1. from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Conv2D, MaxPool2D, Dropout, Flatten, \

Dense

import tensorflow as tf

physical\_devices = tf.config.experimental.list\_physical\_devices('GPU')

tf.config.experimental.set\_memory\_growth(physical\_devices[0], True)

training\_datagen = ImageDataGenerator(rescale=1. / 255,

rotation\_range=0.2,

shear\_range=0.05,

zoom\_range=[0.95, 1.2],

horizontal\_flip=True, )

test\_datagen = ImageDataGenerator(rescale=1. / 255)

training\_path = '/home/ruben/workspace/tfg/deep-learning-facial-recognition/data/age/training'

test\_path = '/home/ruben/workspace/tfg/deep-learning-facial-recognition/data/age/test'

training\_set = training\_datagen.flow\_from\_directory(training\_path,

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical',

shuffle=True,

seed=42)

test\_set = test\_datagen.flow\_from\_directory(test\_path,

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical',

shuffle=True,

seed=42)

age\_classifier = Sequential()

age\_classifier.add(

Conv2D(filters=32, kernel\_size=(3, 3), input\_shape=(64, 64, 3),

activation='relu'))

age\_classifier.add(MaxPool2D(pool\_size=(2, 2)))

age\_classifier.add(

Conv2D(filters=32, kernel\_size=(3, 3), activation='relu'))

age\_classifier.add(MaxPool2D(pool\_size=(2, 2)))

age\_classifier.add(Flatten())

age\_classifier.add(Dense(units=128, activation='relu'))

age\_classifier.add(Dropout(0.3))

age\_classifier.add(Dense(units=64, activation='relu'))

age\_classifier.add(Dropout(0.3))

age\_classifier.add(Dense(units=6, activation='softmax'))

age\_classifier.compile(optimizer='adam', loss='categorical\_crossentropy',

metrics=['accuracy'])

age\_classifier.fit(training\_set,

steps\_per\_epoch=(18966 // 32),

epochs=25,

validation\_data=test\_set,

validation\_steps=(4742//32))

## Resultado:

1. Epoch 1/25
2. 2020-05-20 13:58:01.853789: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:44] Successfully opened dynamic library libcublas.so.10
3. 2020-05-20 13:58:01.984760: I tensorflow/stream\_executor/platform/default/dso\_loader.cc:44] Successfully opened dynamic library libcudnn.so.7
4. 592/592 [==============================] - 37s 63ms/step - loss: 1.2849 - accuracy: 0.4713 - val\_loss: 1.0921 - val\_accuracy: 0.5230
5. Epoch 2/25
6. 592/592 [==============================] - 38s 64ms/step - loss: 1.0624 - accuracy: 0.5470 - val\_loss: 0.9609 - val\_accuracy: 0.5773
7. Epoch 3/25
8. 592/592 [==============================] - 39s 65ms/step - loss: 0.9903 - accuracy: 0.5719 - val\_loss: 0.9178 - val\_accuracy: 0.5961
9. Epoch 4/25
10. 592/592 [==============================] - 38s 64ms/step - loss: 0.9321 - accuracy: 0.5950 - val\_loss: 0.8843 - val\_accuracy: 0.6138
11. Epoch 5/25
12. 592/592 [==============================] - 38s 64ms/step - loss: 0.9044 - accuracy: 0.6071 - val\_loss: 0.8928 - val\_accuracy: 0.6083
13. Epoch 6/25
14. 592/592 [==============================] - 40s 67ms/step - loss: 0.8871 - accuracy: 0.6122 - val\_loss: 0.8796 - val\_accuracy: 0.6090
15. Epoch 7/25
16. 592/592 [==============================] - 42s 70ms/step - loss: 0.8632 - accuracy: 0.6209 - val\_loss: 0.8772 - val\_accuracy: 0.6187
17. Epoch 8/25
18. 592/592 [==============================] - 40s 68ms/step - loss: 0.8470 - accuracy: 0.6262 - val\_loss: 0.8360 - val\_accuracy: 0.6368
19. Epoch 9/25
20. 592/592 [==============================] - 39s 65ms/step - loss: 0.8357 - accuracy: 0.6359 - val\_loss: 0.8436 - val\_accuracy: 0.6294
21. Epoch 10/25
22. 592/592 [==============================] - 39s 66ms/step - loss: 0.8229 - accuracy: 0.6400 - val\_loss: 0.8515 - val\_accuracy: 0.6265
23. Epoch 11/25
24. 592/592 [==============================] - 39s 65ms/step - loss: 0.8067 - accuracy: 0.6461 - val\_loss: 0.8320 - val\_accuracy: 0.6381
25. Epoch 12/25
26. 592/592 [==============================] - 40s 67ms/step - loss: 0.7904 - accuracy: 0.6509 - val\_loss: 0.8182 - val\_accuracy: 0.6383
27. Epoch 13/25
28. 592/592 [==============================] - 43s 72ms/step - loss: 0.7902 - accuracy: 0.6570 - val\_loss: 0.8159 - val\_accuracy: 0.6453
29. Epoch 14/25
30. 592/592 [==============================] - 43s 73ms/step - loss: 0.7844 - accuracy: 0.6526 - val\_loss: 0.8009 - val\_accuracy: 0.6571
31. Epoch 15/25
32. 592/592 [==============================] - 40s 68ms/step - loss: 0.7764 - accuracy: 0.6637 - val\_loss: 0.8113 - val\_accuracy: 0.6474
33. Epoch 16/25
34. 592/592 [==============================] - 40s 67ms/step - loss: 0.7619 - accuracy: 0.6668 - val\_loss: 0.7889 - val\_accuracy: 0.6575
35. Epoch 17/25
36. 592/592 [==============================] - 40s 67ms/step - loss: 0.7588 - accuracy: 0.6687 - val\_loss: 0.7762 - val\_accuracy: 0.6622
37. Epoch 18/25
38. 592/592 [==============================] - 40s 68ms/step - loss: 0.7413 - accuracy: 0.6738 - val\_loss: 0.7995 - val\_accuracy: 0.6586
39. Epoch 19/25
40. 592/592 [==============================] - 39s 66ms/step - loss: 0.7432 - accuracy: 0.6774 - val\_loss: 0.8057 - val\_accuracy: 0.6421
41. Epoch 20/25
42. 592/592 [==============================] - 40s 67ms/step - loss: 0.7361 - accuracy: 0.6731 - val\_loss: 0.7885 - val\_accuracy: 0.6565
43. Epoch 21/25
44. 592/592 [==============================] - 41s 69ms/step - loss: 0.7288 - accuracy: 0.6840 - val\_loss: 0.7825 - val\_accuracy: 0.6554
45. Epoch 22/25
46. 592/592 [==============================] - 40s 68ms/step - loss: 0.7175 - accuracy: 0.6814 - val\_loss: 0.8006 - val\_accuracy: 0.6577
47. Epoch 23/25
48. 592/592 [==============================] - 41s 69ms/step - loss: 0.7195 - accuracy: 0.6873 - val\_loss: 0.7905 - val\_accuracy: 0.6552
49. Epoch 24/25
50. 592/592 [==============================] - 39s 66ms/step - loss: 0.7104 - accuracy: 0.6893 - val\_loss: 0.8019 - val\_accuracy: 0.6505
51. Epoch 25/25
52. 592/592 [==============================] - 39s 66ms/step - loss: 0.6982 - accuracy: 0.6952 - val\_loss: 0.8128 - val\_accuracy: 0.6518
53. Process finished with exit code 0

## Conclusiones y Resultado vs. Version Anterior:

Apenas hay diferencias con respecto al modelo anterior. Tanto la precisión de Train y de Test se mantienen cercanas, por lo que, de haber overfitting, se ha reducido considerablemente con respecto a las primeras arquitecturas de la CNN.

Se ha reducido ligeramente la precisión del modelo, tanto en entrenamiento como en test, en torno a un 0.60-0.70%. Sin embargo, parece que en las últimas épocas de entrenamiento la tasa de acierto continuaba creciendo lentamente, por lo que sospecho que podría aumentar el número de épocas para continuar explotando esta arquitectura, o bien, aumentar el tamaño del batch para que el ajuste de los pesos (en backpropagation) sea algo más preciso y el acelerar el proceso de aprendizaje.

Otras opciones: aumentar el número de ejemplos por época, aumentar el tamaño de imagen por si surgieran unos mapas de características (durante las convoluciones) más determinantes, añadir una tasa de aprendizaje de forma explícita (o, en su defecto, un learning rate schedule con ayuda de Keras).

Valores actuales:

- Acc. Training: 69.52%

- Acc. Test: 65.18%