```
 1 import tensorflow as tf
 2 from tensorflow.keras.optimizers import RMSprop
 3 from tensorflow.keras.preprocessing.image import
   ImageDataGenerator
 4 import matplotlib.pyplot as plt
 5 import os
 6
 7
 8 base_dir = 'G:\GIEyA\TFG\MeteorClassificationProject\
   labeledData'
 9 train_dir = os.path.join(base_dir, 'train')
10 validation_dir = os.path.join(base_dir, 'validation')
11
12 train_meteors_dir = os.path.join(train_dir, 'meteors'
   )
13 train_non_meteors_dir = os.path.join(train_dir, '
   non_meteors')
14 validation_meteors_dir = os.path.join(validation_dir
   , 'meteors')
15 validation_non_meteors_dir = os.path.join(
   validation_dir, 'non_meteors')
16
17 print('total training meteors images: ', len(os.
   listdir(train_meteors_dir)))
18 print('total training non-meteors images: ', len(os.
   listdir(train_non_meteors_dir)))
19 print('total validation meteors images: ', len(os.
   listdir(validation_meteors_dir)))
20 print('total validation non-meteors images: ', len(os
   .listdir(validation_non_meteors_dir)))
21
22
23 #Rescale all images by 1./255
24
25 train_datagen = ImageDataGenerator(rescale=1.0/255,
26                                    rotation_range=40
   , #Range from 0 to 180 degrees to randomly rotate
   images. In this case it's going to rotate between 0
   and 40 degrees
27                                    width_shift_range=
   0.2, #Move image in this fram (20%)
28                                    height_shift_range
   =0.2,
29                                    shear_range=0.2, #
```

```
29 Girar la imagen un 20%
30                                     zoom_range=0.5, #
   Zoom up-to 20%
31                                     horizontal_flip=
   True, #Efecto cámara: girar la imagen con respecto al
    eje vertical
32                                     fill_mode='nearest
   ') #Ckeck other options
33
34 test_datagen = ImageDataGenerator(rescale=1.0/255.)
35
36 train_generator = train_datagen.flow_from_directory(
   train_dir,
37
   batch_size=8,
38
   class_mode='binary',
39
   color_mode='grayscale',
40
   target_size=(300, 300))
41 validation_generator = test_datagen.
   flow_from_directory(validation_dir,
42
      batch_size=4,
43
      class_mode='binary',
44
      color_mode='grayscale',
45
      target_size=(300, 300))
46
47
48 model = tf.keras.models.Sequential([
49     tf.keras.layers.Conv2D(16, (3,3), activation='
   relu', input_shape=(300, 300, 1)),
50     tf.keras.layers.MaxPooling2D(2,2),
51     tf.keras.layers.Conv2D(32, (3,3), activation='
   relu'),
52     tf.keras.layers.MaxPooling2D(2,2),
53     tf.keras.layers.Conv2D(64, (3, 3), activation='
   relu'),
54     tf.keras.layers.MaxPooling2D(2, 2),
55     tf.keras.layers.Flatten(),
```

```
56      #tf.keras.layers.Dropout(0.2),
57      tf.keras.layers.Dense(1024, activation='relu'),
58      tf.keras.layers.Dense(1, activation='sigmoid')])
59
60 print(model.summary())
61
62 model.compile(optimizer=RMSprop(lr=0.001),
63               loss='binary_crossentropy',
64               metrics=['accuracy'])
65
66 #53.079 -> Training
67 #13.271 -> Validation
68 #53.079/batch_size =
69 #13.271/batch_size =
70
71 history = model.fit_generator(train_generator,
72                     validation_data=
   validation_generator,
73                     steps_per_epoch=6000,
74                     epochs=10, #Later train with more
    epochs if neccessary
75                     validation_steps=1500,
76                     verbose=1)
77
78 acc      = history.history['accuracy']
79 val_acc  = history.history['val_accuracy']
80 loss     = history.history['loss']
81 val_loss = history.history['val_loss']
82 epochs = range(len(acc)) #Get number of epochs
83
84 plt.plot(epochs, acc)
85 plt.plot(epochs, val_acc)
86 plt.title('Meteor detection training and validation
   accuracy')
87 plt.figure()
88
89 plt.plot(epochs, loss)
90 plt.plot(epochs, val_loss)
91 plt.title('Meteor detection training and validation
   loss')
92
93 plt.show()
```