```
 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 import multiprocessing
 7
 8 import multiprocessing
 9
10 def trainCNN( ):
11
12     base_dir = 'G:\GIEyA\TFG\
   MeteorClassificationProject\labeledData'
13     train_dir = os.path.join(base_dir, 'train_640x360
   ')
14     validation_dir = os.path.join(base_dir, '
   validation_640x360')
15
16     train_meteors_dir = os.path.join(train_dir, '
   meteors')
17     train_non_meteors_dir = os.path.join(train_dir, '
   non_meteors')
18     validation_meteors_dir = os.path.join(
   validation_dir, 'meteors')
19     validation_non_meteors_dir = os.path.join(
   validation_dir, 'non_meteors')
20
21     print('total training meteors images: ', len(os.
   listdir(train_meteors_dir)))
22     print('total training non-meteors images: ', len(
   os.listdir(train_non_meteors_dir)))
23     print('total validation meteors images: ', len(os
   .listdir(validation_meteors_dir)))
24     print('total validation non-meteors images: ',
   len(os.listdir(validation_non_meteors_dir)))
25
26
27     #Rescale all images by 1./255
28
29     train_datagen = ImageDataGenerator(rescale=1.0/
   255,
30                                        rotation_range
   =40, #Range from 0 to 180 degrees to randomly rotate
```

```
30 images. In this case it's going to rotate between 0
   and 40 degrees
31
   width_shift_range=0.2, #Move image in this fram (20%)
32
   height_shift_range=0.2,
33                                          shear_range=0.
   2, #Girar la imagen un 20%
34                                          zoom_range=0.5
   , #Zoom up-to 20%
35
   horizontal_flip=True, #Efecto cámara: girar la imagen
    con respecto al eje vertical
36                                          fill_mode='
   nearest') #Ckeck other options
37
38     test_datagen = ImageDataGenerator(rescale=1.0/255
   .)
39
40     train_generator = train_datagen.
   flow_from_directory(train_dir,
41
      batch_size=8,
42
      class_mode='binary',
43
      color_mode='grayscale',
44
      target_size=(300, 300))
45      validation_generator = test_datagen.
   flow_from_directory(validation_dir,
46
         batch_size=4,
47
         class_mode='binary',
48
         color_mode='grayscale',
49
         target_size=(300, 300))
50
51
52     model = tf.keras.models.Sequential([#Try Dropout
   after each Conv2D + MaxPôoling2D stage
53         tf.keras.layers.Conv2D(16, (3,3), activation=
```

```
53 'relu', input_shape=(300, 300, 1)),
54         tf.keras.layers.MaxPooling2D(2,2),
55         tf.keras.layers.Dropout(0.2),
56         tf.keras.layers.Conv2D(64, (3,3), activation=
   'relu'),
57         tf.keras.layers.MaxPooling2D(2,2),
58         tf.keras.layers.Dropout(0.2),
59         tf.keras.layers.Flatten(),
60         tf.keras.layers.Dense(128, activation='relu'
   ),
61         tf.keras.layers.Dense(16, activation='relu'),
62         tf.keras.layers.Dense(4, activation='relu'),
63         tf.keras.layers.Dense(1, activation='sigmoid'
   )])
64
65     print(model.summary())
66
67     model.compile(optimizer=RMSprop(lr=0.001),
68                   loss='binary_crossentropy',
69                   metrics=['accuracy'])
70
71     #53.079 -> Training
72     #13.271 -> Validation
73     #53.079/batch_size =
74     #13.271/batch_size =
75
76     history = model.fit_generator(train_generator,
77                           validation_data=
   validation_generator,
78                           steps_per_epoch=6000,
79                           epochs=20, #Later train with
   more epochs if neccessary
80                           validation_steps=3000,
81                           verbose=2)
82
83     acc      = history.history['accuracy']
84     val_acc  = history.history['val_accuracy']
85     loss     = history.history['loss']
86     val_loss = history.history['val_loss']
87     epochs = range(len(acc)) #Get number of epochs
88
89     plt.plot(epochs, acc)
90     plt.plot(epochs, val_acc)
91     plt.title('Meteor detection training and
```

```
 91 validation accuracy')
 92     plt.figure()
 93 
 94     plt.plot(epochs, loss)
 95     plt.plot(epochs, val_loss)
 96     plt.title('Meteor detection training and
    validation loss')
 97 
 98     plt.show()
 99 
100 if __name__ == '__main__':
101     p = multiprocessing.Process(target=trainCNN)
102     p.start()
103     p.join()
```