```python
 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
```

```python
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(32, (3,3), activation=
```

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

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