```
 1 import tensorflow as tf
 2 from tensorflow.keras.callbacks import
   LearningRateScheduler
 3 from tensorflow.keras.layers import Dense,  Conv2D,
   MaxPooling2D, Dropout, Flatten,
   GlobalAveragePooling2D
 4 from tensorflow.keras.optimizers import Adam, SGD,
   RMSprop
 5 from tensorflow.keras.preprocessing.image import
   ImageDataGenerator
 6 import numpy as np
 7 import matplotlib.pyplot as plt
 8 import os
 9 import multiprocessing
10
11 def trainCNN( ):
12
13     tf.keras.backend.clear_session()
14
15     base_dir = 'G:\GIEyA\TFG\meteor_classification\
   labeledData\evenData'
16     train_dir = os.path.join(base_dir, 'train')
17     validation_dir = os.path.join(base_dir, 'valid')
18
19     train_meteors_dir = os.path.join(train_dir, '
   meteors')
20     train_non_meteors_dir = os.path.join(train_dir, '
   non_meteors')
21     validation_meteors_dir = os.path.join(
   validation_dir, 'meteors')
22     validation_non_meteors_dir = os.path.join(
   validation_dir, 'non_meteors')
23
24     print('total training meteors images: ', len(os.
   listdir(train_meteors_dir)))
25     print('total training non-meteors images: ', len(
   os.listdir(train_non_meteors_dir)))
26     print('total validation meteors images: ', len(os
   .listdir(validation_meteors_dir)))
27     print('total validation non-meteors images: ',
   len(os.listdir(validation_non_meteors_dir)))
28
29
30     #Rescale all images by 1./255
```

```
31
32      train_datagen = ImageDataGenerator(rescale=1.0/
    255,
33                                         rotation_range
    =10, # Range from 0 to 180 degrees to randomly rotate
     images
34
    width_shift_range=0.05,
35
    height_shift_range=0.05,
36                                         shear_range=5
    , # Shear the image by 5 degrees
37                                         zoom_range=0.1
    ,
38
    horizontal_flip=True,
39                                         vertical_flip=
    True,
40                                         fill_mode='
    nearest'
41                                         )
42
43      test_datagen = ImageDataGenerator(rescale=1.0/255
    .)
44
45      train_generator = train_datagen.
    flow_from_directory(train_dir,
46
        batch_size=16,
47
        class_mode='binary',
48
        color_mode='grayscale',
49
        target_size=(480, 480)) # 640x360 = 480x480. (640
    , 360)
50      validation_generator = test_datagen.
    flow_from_directory(validation_dir,
51
            batch_size=16,
52
            class_mode='binary',
53
            color_mode='grayscale',
```

```
54
          target_size=(480, 480))
55
56
57      model = tf.keras.models.Sequential([
58          Conv2D(32, (3, 3), activation='relu',
    input_shape=(480, 480, 1)), MaxPooling2D(2,2), #
    Dropout(0.05),
59          Conv2D(16, (3, 3), activation='relu',
    kernel_initializer='he_uniform'), MaxPooling2D(2, 2
    ), #Dropout(0.05),
60          Conv2D(16, (3, 3), activation='relu',
    kernel_initializer='he_uniform'), MaxPooling2D(2, 2
    ), #Dropout(0.05),
61          Conv2D(12, (2, 2), activation='relu',
    kernel_initializer='he_uniform'), MaxPooling2D(2, 2
    ), #Dropout(0.05),
62          Conv2D(8,  (2, 2), activation='relu',
    kernel_initializer='he_uniform'), MaxPooling2D(2, 2
    ), #Dropout(0.05),
63          Conv2D(4,  (2, 2), activation='relu',
    kernel_initializer='he_uniform'), MaxPooling2D(2, 2
    ), #Dropout(0.05),
64          #Conv2D(4, (2, 2), activation='relu'),
65          Flatten(),
66          Dense(144, activation='relu',
    kernel_initializer='he_uniform'),
67          #Dense(32, activation='relu',
    kernel_initializer='he_uniform'),
68          Dense(8, activation='relu',
    kernel_initializer='he_uniform'),
69          Dense(1, activation='sigmoid',
    kernel_initializer='he_uniform')
70      ])
71
72      print(model.summary())
73      optimizer = Adam(learning_rate=4e-3) #3e-3 # Try
    with more and less learning rate # 5e-3
74      model.compile(optimizer=optimizer,
75                    loss='binary_crossentropy',
76                    metrics=['accuracy'])
77
78      #39.480 -> Training 39480 = 2 x 2 x 2 × 3 × 5 × 7
     × 47
```

```
 79     #9.872 -> Validation = 2 x 2 x 2 x 2 × 617
 80
 81     history = model.fit(train_generator,
 82                         validation_data=
    validation_generator,
 83                         steps_per_epoch=2467, #2467
 84                         epochs=20, #Later train with
     more epochs if neccessary
 85                         validation_steps=617, #617
 86                         verbose=1)
 87
 88     acc      = history.history['accuracy']
 89     val_acc  = history.history['val_accuracy']
 90     loss     = history.history['loss']
 91     val_loss = history.history['val_loss']
 92     epochs = range(len(acc)) #Get number of epochs
 93
 94     plt.plot(epochs, acc)
 95     plt.plot(epochs, val_acc)
 96     plt.title('Meteor detection training and
    validation accuracy')
 97
 98     plt.figure()
 99     plt.plot(epochs, loss)
100     plt.plot(epochs, val_loss)
101     plt.title('Meteor detection training and
    validation loss')
102
103     plt.show()
104
105 if __name__ == '__main__':
106     p = multiprocessing.Process(target=trainCNN)
107     p.start()
108     p.join()
```