```
 1 import tensorflow as tf
 2 from tensorflow.keras.callbacks import
   LearningRateScheduler
 3 from tensorflow.keras.layers import Dense,  Conv2D,
   MaxPooling2D, Dropout, Flatten
 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
```

```
54        target_size=(480, 480))
55
56
57    model = tf.keras.models.Sequential([
58        Conv2D(32, (2, 2), activation='relu',
   input_shape=(480, 480, 1)), MaxPooling2D(2,2), #
   Dropout(0.2),
59        Conv2D(16, (2, 2), activation='relu'),
   MaxPooling2D(2, 2), #Dropout(0.2),
60        Conv2D(16, (2, 2), activation='relu'),
   MaxPooling2D(2, 2), #Dropout(0.2),
61        Conv2D(12, (2, 2), activation='relu'),
   MaxPooling2D(2, 2), #Dropout(0.2),
62        Conv2D(12, (2, 2), activation='relu'),
   MaxPooling2D(2, 2), #Dropout(0.2),
63        Conv2D(12, (2, 2), activation='relu'),
   MaxPooling2D(2, 2), #Dropout(0.2),
64        Conv2D(12, (2, 2), activation='relu'),
65        Flatten(),
66        Dense(300, activation='relu'),
67        Dense(10, activation='relu'),
68        Dense(1, activation='sigmoid')
69    ])
70
71    print(model.summary())
72    optimizer = Adam(learning_rate=5e-3) #3e-3
73    model.compile(optimizer=optimizer,
74                  loss='binary_crossentropy',
75                  metrics=['accuracy'])
76
77    #39.480 -> Training 39480 = 2 x 2 x 2 × 3 × 5 × 7
    × 47
78    #9.872 -> Validation = 2 x 2 x 2 x 2 × 617
79
80    history = model.fit(train_generator,
81                        validation_data=
   validation_generator,
82                        steps_per_epoch=2466,
83                        epochs=10, #Later train with
   more epochs if neccessary
84                        validation_steps=616,
85                        verbose=1)
86
87    acc      = history.history['accuracy']
```

```
 88      val_acc  = history.history['val_accuracy']
 89      loss     = history.history['loss']
 90      val_loss = history.history['val_loss']
 91      epochs = range(len(acc)) #Get number of epochs
 92
 93      plt.plot(epochs, acc)
 94      plt.plot(epochs, val_acc)
 95      plt.title('Meteor detection training and
    validation accuracy')
 96
 97      plt.figure()
 98      plt.plot(epochs, loss)
 99      plt.plot(epochs, val_loss)
100      plt.title('Meteor detection training and
    validation loss')
101
102      plt.show()
103
104 if __name__ == '__main__':
105      p = multiprocessing.Process(target=trainCNN)
106      p.start()
107      p.join()
```