```
 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=48,
47
       class_mode='binary',
48
       color_mode='grayscale',
49
       target_size=(640, 360))
50      validation_generator = test_datagen.
    flow_from_directory(validation_dir,
51
           batch_size=48,
52
           class_mode='binary',
53
           color_mode='grayscale',
54
           target_size=(640, 360))
```

```
55
56
57      model = tf.keras.models.Sequential([
58          Conv2D(16, (2,2), activation='relu',
    input_shape=(640, 360, 1)),
59          MaxPooling2D(2,2),
60          Dropout(0.2),
61          Conv2D(16, (2, 2), activation='relu'),
62          MaxPooling2D(2, 2),
63          #Dropout(0.2), # No dropout in transitions
64          Conv2D(12, (2, 2), activation='relu'),
65          MaxPooling2D(2, 2),
66          Dropout(0.2),
67          Conv2D(12, (2, 2), activation='relu'),
68          #MaxPooling2D(2, 2),
69          #Dropout(0.2), # No dropout in transitions
70          Conv2D(12, (2, 2), activation='relu'),
71          MaxPooling2D(2, 2),
72          Dropout(0.2),
73          Conv2D(8, (2, 2), activation='relu'),
74          MaxPooling2D(2, 2),
75          Dropout(0.2),
76          Conv2D(8, (2, 2), activation='relu'),
77          Dropout(0.2),
78          Conv2D(8, (2, 2), activation='relu'),
79          Dropout(0.2),
80          Conv2D(4, (2, 2), activation='relu'),
81          Dropout(0.2),
82          Conv2D(4, (2, 2), activation='relu'),
83          Dropout(0.2),
84          Conv2D(4, (2, 2), activation='relu'),
85          MaxPooling2D(2, 2),
86          #Dropout(0.2), # No dropout in transitions
87          Flatten(),
88          Dense(48, activation='relu'),
89          Dense(24, activation='relu'),
90          Dense(12, activation='relu'),
91          Dense(1, activation='sigmoid')])
92
93      print(model.summary())
94      optimizer = Adam(learning_rate=3e-3)
95      model.compile(optimizer=optimizer,
96                    loss='binary_crossentropy',
97                    metrics=['accuracy'])
```

```
 98
 99      #39.480 -> Training 39480 = 2 x 2 x 2 × 3 × 5 × 
    7 × 47
100      #9.872 -> Validation = 2 x 2 x 2 x 2 × 617
101
102      history = model.fit(train_generator,
103                          validation_data=
    validation_generator,
104                          steps_per_epoch=822,
105                          epochs=10, #Later train with
     more epochs if neccessary
106                          validation_steps=205,
107                          verbose=1)
108
109      acc      = history.history['accuracy']
110      val_acc  = history.history['val_accuracy']
111      loss     = history.history['loss']
112      val_loss = history.history['val_loss']
113      epochs = range(len(acc)) #Get number of epochs
114
115      plt.plot(epochs, acc)
116      plt.plot(epochs, val_acc)
117      plt.title('Meteor detection training and
    validation accuracy')
118
119      plt.figure()
120      plt.plot(epochs, loss)
121      plt.plot(epochs, val_loss)
122      plt.title('Meteor detection training and
    validation loss')
123
124      plt.show()
125
126  if __name__ == '__main__':
127      p = multiprocessing.Process(target=trainCNN)
128      p.start()
129      p.join()
```