

```
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 from tensorflow.keras.callbacks import Callback
7 import numpy as np
8 import matplotlib.pyplot as plt
9 import os
10 from os.path import join
11 import multiprocessing
12
13 def trainCNN( ):
14
15     tf.keras.backend.clear_session()
16
17     base_dir = 'G:\GIEyA\TFG\meteor_classification\
  labeledData\evenData'
18     results_dir_weights = 'G:\GIEyA\TFG\
  meteor_classification\\results\weights'
19
20     train_dir = join(base_dir, 'train')
21     validation_dir = join(base_dir, 'valid')
22
23     train_meteors_dir = join(train_dir, 'meteors')
24     train_non_meteors_dir = join(train_dir, '
  non_meteors')
25     validation_meteors_dir = join(validation_dir, '
  meteors')
26     validation_non_meteors_dir = join(validation_dir
  , 'non_meteors')
27
28     print('total training meteors images: ', len(os.
  listdir(train_meteors_dir)))
29     print('total training non-meteors images: ', len(
  os.listdir(train_non_meteors_dir)))
30     print('total validation meteors images: ', len(os
  .listdir(validation_meteors_dir)))
31     print('total validation non-meteors images: ',
```

```

31 len(os.listdir(validation_non_meteors_dir)))
32
33
34     #Rescale all images by 1./255
35
36     train_datagen = ImageDataGenerator(rescale=1.0/
37 255,
38                                     rotation_range
39 =10, # Range from 0 to 180 degrees to randomly rotate
40 images
41 width_shift_range=0.05,
42 height_shift_range=0.05,
43                                     shear_range=5
44 , # Shear the image by 5 degrees
45                                     zoom_range=0.1
46 '
47 horizontal_flip=True,
48                                     vertical_flip=
49 True,
50                                     fill_mode='
51 nearest'
52                                     )
53
54     test_datagen = ImageDataGenerator(rescale=1.0/255
55 .)
56
57     train_generator = train_datagen.
58 flow_from_directory(train_dir,
59
60     batch_size=16, #16
61
62     class_mode='binary',
63
64     color_mode='grayscale',
65
66     target_size=(480, 480)) # 640x360 = 480x480. (640
67 , 360)
68
69     validation_generator = test_datagen.
70 flow_from_directory(validation_dir,
71
72     batch_size=16, #16

```

```

56         class_mode='binary',
57         color_mode='grayscale',
58         target_size=(480, 480))
59
60
61     model = tf.keras.models.Sequential([
62         Conv2D(32, (3, 3), activation='relu',
63         input_shape=(480, 480, 1)), MaxPooling2D(2,2), #
64         Dropout(0.05),
65         Conv2D(16, (3, 3), activation='relu',
66         kernel_initializer='he_uniform'), MaxPooling2D(2, 2
67         ), #Dropout(0.05),
68         Conv2D(16, (3, 3), activation='relu',
69         kernel_initializer='he_uniform'), MaxPooling2D(2, 2
70         ), #Dropout(0.05),
71         Conv2D(12, (2, 2), activation='relu',
72         kernel_initializer='he_uniform'), MaxPooling2D(2, 2
73         ), #Dropout(0.05),
74         Conv2D(8, (2, 2), activation='relu',
75         kernel_initializer='he_uniform'), MaxPooling2D(2, 2
76         ), #Dropout(0.05),
77         Conv2D(4, (2, 2), activation='relu',
78         kernel_initializer='he_uniform'), MaxPooling2D(2, 2
79         ), #Dropout(0.05),
80         #Conv2D(4, (2, 2), activation='relu'),
81         Flatten(),
82         Dense(144, activation='relu',
83         kernel_initializer='he_uniform'),
84         #Dense(32, activation='relu',
85         kernel_initializer='he_uniform'),
86         Dense(8, activation='relu',
87         kernel_initializer='he_uniform'),
88         Dense(1, activation='sigmoid',
89         kernel_initializer='he_uniform')
90     ])
91
92     print(model.summary())
93     optimizer = Adam(learning_rate=4e-3) #3e-3 # Try
94     with more and less learning rate # 5e-3
95     model.compile(optimizer=optimizer,
96                   loss='binary_crossentropy',

```

```

80             metrics=['accuracy'])
81
82     class SaveModelCallback(Callback):
83         def __init__(self, threshold):
84             super(SaveModelCallback, self).__init__(
85                 ()
86                 self.threshold = threshold
87
88         def on_epoch_end(self, epoch, logs=None):
89             if(logs.get('accuracy') > self.threshold
90 ):
91                 model.save_weights(join(
92                     results_dir_weights, 'model_acc_' + str(logs.get('
93                     accuracy'))[0:5] + '.h5'), save_format='h5')
94
95     callback90 = SaveModelCallback(0.90)
96
97     #39.480 -> Training 39480 = 2 x 2 x 2 x 3 x 5 x
98     7 x 47
99     #9.872 -> Validation = 2 x 2 x 2 x 2 x 617
100    history = model.fit(train_generator,
101                        validation_data=
102                        validation_generator,
103                        steps_per_epoch=2467, #2467
104                        epochs=15, #Later train with
105                        more epochs if neccessary
106                        validation_steps=617, #617
107                        verbose=1,
108                        callbacks=[callback90])
109
110    acc      = history.history['accuracy']
111    val_acc  = history.history['val_accuracy']
112    loss     = history.history['loss']
113    val_loss = history.history['val_loss']
114    epochs = range(len(acc)) #Get number of epochs
115
116    plt.plot(epochs, acc)
117    plt.plot(epochs, val_acc)
118    plt.title('Meteor detection training and
119    validation accuracy')
120
121    plt.figure()
122    plt.plot(epochs, loss)
123    plt.plot(epochs, val_loss)

```

```
116     plt.title('Meteor detection training and  
validation loss')  
117  
118     plt.show()  
119  
120 if __name__ == '__main__':  
121     p = multiprocessing.Process(target=trainCNN)  
122     p.start()  
123     p.join()  
124  
125
```