

```
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 ImageResolution = (640, 360)
14 ImageResolutionGrayScale = (640, 360, 1)
15
16 def trainCNN( ):
17
18     tf.keras.backend.clear_session()
19
20     base_dir = 'G:\GIEyA\TFG\meteor_classification\
  labeledData\evenData'
21     results_dir_weights = 'G:\GIEyA\TFG\
  meteor_classification\\results\weights\model_18'
22
23     train_dir = join(base_dir, 'train')
24     validation_dir = join(base_dir, 'valid')
25
26     train_meteors_dir = join(train_dir, 'meteors')
27     train_non_meteors_dir = join(train_dir, '
  non_meteors')
28     validation_meteors_dir = join(validation_dir, '
  meteors')
29     validation_non_meteors_dir = join(validation_dir
  , 'non_meteors')
30
31     print('total training meteors images: ', len(os.
  listdir(train_meteors_dir)))
32     print('total training non-meteors images: ', len(
  os.listdir(train_non_meteors_dir)))
```

```

33     print('total validation meteors images: ', len(os
        .listdir(validation_meteors_dir)))
34     print('total validation non-meteors images: ',
        len(os.listdir(validation_non_meteors_dir)))
35
36
37     #Rescale all images by 1./255
38
39     train_datagen = ImageDataGenerator(rescale=1.0/
        255#,
40                                     #
        rotation_range=10, # Range from 0 to 180 degrees to
        randomly rotate images
41                                     #
        width_shift_range=0.05,
42                                     #
        height_shift_range=0.05,
43                                     #shear_range=5
        , # Shear the image by 5 degrees
44                                     #zoom_range=0.
        1,
45                                     #
        horizontal_flip=True,
46                                     #vertical_flip
        =True,
47                                     #fill_mode='
        nearest'
48                                     )
49
50     test_datagen = ImageDataGenerator(rescale=1.0/255
        .)
51
52     train_generator = train_datagen.
        flow_from_directory(train_dir,
53
        batch_size=16, #16
54
        class_mode='binary',
55
        color_mode='grayscale',
56
        target_size=ImageResolution) # 640x360 = 480x480
        . (640, 360)
57     validation_generator = test_datagen.

```

```
57 flow_from_directory(validation_dir,
58
        batch_size=16, #16
59
        class_mode='binary',
60
        color_mode='grayscale',
61
        target_size=ImageResolution)
62
63
64     model = tf.keras.models.Sequential([
65         Conv2D(16, (11, 11), activation='relu',
input_shape=ImageResolutionGrayScale, strides=2),
66         MaxPooling2D(pool_size=(3, 3)),
67         Dropout(0.25),
68
69         Conv2D(12, (7, 7), activation='relu',
kernel_initializer='he_uniform'),
70         #Conv2D(12, (3, 3), activation='relu',
kernel_initializer='he_uniform'),
71         MaxPooling2D(pool_size=(2, 2)),
72         Dropout(0.25),
73
74         Conv2D(12, (5, 5), activation='relu',
kernel_initializer='he_uniform'),
75         #Conv2D(16, (3, 3), activation='relu',
kernel_initializer='he_uniform'),
76         MaxPooling2D(pool_size=(2, 2)),
77         Dropout(0.25),
78
79         Conv2D(12, (3, 3), activation='relu',
kernel_initializer='he_uniform'),
80         #Conv2D(8, (3, 3), activation='relu',
kernel_initializer='he_uniform'),
81         MaxPooling2D(pool_size=(2, 2)),
82
83         Flatten(),
84         Dense(480, activation='relu',
kernel_initializer='he_uniform'),
85         Dropout(0.30),
86         Dense(16, activation='relu',
kernel_initializer='he_uniform'),
87         Dropout(0.20),
```

```

88         Dense(1, activation='sigmoid',
kernel_initializer='he_uniform')
89     ])
90
91     print(model.summary())
92     optimizer = Adam(learning_rate=5e-4) #3e-3 # Try
with more and less learning rate # 5e-3
93     model.compile(optimizer=optimizer,
94                   loss='binary_crossentropy',
95                   metrics=['accuracy'])
96     #model.load_weights(join(results_dir_weights, '
model_acc_0.926.h5'))
97
98     class SaveModelCallback(Callback):
99         def __init__(self, threshold):
100             super(SaveModelCallback, self).__init__
101             ()
102             self.threshold = threshold
103
104         def on_epoch_end(self, epoch, logs=None):
105             if(logs.get('accuracy') >= self.
threshold):
106                 model.save_weights(join(
results_dir_weights, '10_layers_model_2_acc_' + str
(logs.get('accuracy'))[0:6] + '_val_acc' + str(logs.
get('val_accuracy'))[0:6] + '.h5'), save_format='h5'
)
107
108     callback92 = SaveModelCallback(0.920)
109
110     #39.480 -> Training 39480 = 2 x 2 x 2 x 3 x 5 x
7 x 47
111     #9.872 -> Validation = 2 x 2 x 2 x 2 x 617
112     history = model.fit(train_generator,
validation_data=
validation_generator,
113                         steps_per_epoch=2467, #2467
#4934
114                         epochs=150, #Later train
with more epochs if neccessary
115                         validation_steps=617, #617 #
1234
116                         verbose=1,
117                         callbacks=[callback92])

```

```
118
119     acc      = history.history['accuracy']
120     val_acc   = history.history['val_accuracy']
121     loss      = history.history['loss']
122     val_loss  = history.history['val_loss']
123     epochs = range(len(acc)) #Get number of epochs
124
125     plt.plot(epochs, acc)
126     plt.plot(epochs, val_acc)
127     plt.title('Meteor detection training and
validation accuracy')
128
129     plt.figure()
130     plt.plot(epochs, loss)
131     plt.plot(epochs, val_loss)
132     plt.title('Meteor detection training and
validation loss')
133
134     plt.show()
135
136 if __name__ == '__main__':
137     p = multiprocessing.Process(target=trainCNN)
138     p.start()
139     p.join()
140
141
```