# EMERGING METHODS FOR EARLY DETECTION OF

**FOREST FIRE**

**SPRINT 2**

|  |  |
| --- | --- |
| Date | 17-11-2022 |
| Team id | PNT2022TMID34407 |
| Project Name | Emerging Methods for Early Detection of  ForestFires |

In [1]: **import** keras

**from** keras.preprocessing.image **import** ImageDataGenerator

**import** keras

**from** keras.preprocessing.image **import** ImageDataGenerator

In [2]: *#Define the parameters/arguments for ImageDataGenerator class* train\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255,shear\_range**=**0.2,rotation\_range**=**180, zoom\_range

test\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255)

*#Define the parameters/arguments for ImageDataGenerator class* train\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255,shear\_range**=**0.2,rotation\_range**=**180, zoom\_range

test\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255)

In [3]: *#Applying ImageDataGenerator functionality to trainset*

x\_train**=**train\_datagen**.**flow\_from\_directory(r'C:\Users\devi\Downloads\archive\Dataset

\Dataset\ target\_size**=**(128,128), batch\_size**=**32, class\_mode**=**'binary')

*#Applying ImageDataGenerator functionality to trainset*

x\_train**=**train\_datagen**.**flow\_from\_directory(r'C:\Users\devi\Downloads\archive\Dataset\Dataset\

target\_size**=**(128,128), batch\_size**=**32, class\_mode**=**'binary')

Found 436 images belonging to 2 classes.

In [4]: *#Applying ImageDataGenerator functionality to testset*

x\_test**=**test\_datagen**.**flow\_from\_directory(r'C:\Users\devi\Downloads\archive\Dataset\Dataset\te

target\_size**=**(128,128), batch\_size**=**32, class\_mode**=**'binary')

*#Applying ImageDataGenerator functionality to testset*

x\_test**=**test\_datagen**.**flow\_from\_directory(r'C:\Users\devi\Downloads\archive\Dataset

\Dataset\te target\_size**=**(128,128), batch\_size**=**32, class\_mode**=**'binary')

Found 121 images belonging to 2 classes.

In [5]:*#import model building libraries*

*#To define Linear initialisation import Sequential*

**from** keras.models **import** Sequential

*#To add layers import Dense*

**from** keras.layers **import** Dense

*#To create Convolution kernel import Convolution2D*

**from** keras.layers **import** Convolution2D

*#import Maxpooling layer*

**from** keras.layers **import** MaxPooling2D

*#import flatten layer*

**from** keras.layers **import** Fl

*#import model building libraries*

*#To define Linear initialisation import Sequential*

**from** keras.models **import** Sequential

*#To add layers import Dense*

**from** keras.layers **import** Dense

*#To create Convolution kernel import Convolution2D*

**from** keras.layers **import** Convolution2D

*#import Maxpooling layer*

**from** keras.layers **import** MaxPooling2D

*#import flatten layer*

**from** keras.layers **import** Flatten **import** warnings warnings**.**filterwarnings('ignore')

In [7]: *#initializing the model*

model**=**Sequential()

*#initializing the model*

model**=**Sequential()

In [8]: *#add convolutional layer* model**.**add(Convolution2D(32,(3,3),input\_shape**=**(128,128,3),activation**=**'relu')) *#add maxpooling layer*

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

*#add flatten layer*

model**.**add(Flatten())

*#add convolutional layer* model**.**add(Convolution2D(32,(3,3),input\_shape**=**(128,128,3),activation**=**'relu')) *#add maxpooling layer*

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

*#add flatten layer*

model**.**add(Flatten())

In [9]: *#add hidden layer*

model**.**add(Dense(150,activation**=**'relu')) *#add output layer* model**.**add(Dense(1,activation**=**'sigmoid'))

*#add hidden layer* model**.**add(Dense(150,activation**=**'relu')) *#add output layer* model**.**add(Dense(1,activation**=**'sigmoid'))

In [10]: *#configure the learning process*

model**.**compile(loss**=**'binary\_crossentropy',optimizer**=**"adam",metrics**=**["accuracy"])

*#configure the learning process*

model**.**compile(loss**=**'binary\_crossentropy',optimizer**=**"adam",metrics**=**["accuracy"])

In [11]: *#Training the model*

model**.**fit\_generator(x\_train,steps\_per\_epoch**=**14,epochs**=**10, validation\_data**=**x\_test,validation\_st

*#Training the model* model**.**fit\_generator(x\_train,steps\_per\_epoch**=**14,epochs**=**10, validation\_data**=**x\_test,validation\_st

Epoch 1/10

14/14 [==============================] - 84s 6s/step - loss: 4.2334 - accuracy: 0.5619 -

val\_

loss: 1.3686 - val\_accuracy:

0.5950Epoch 2/10

14/14 [==============================] - 74s 5s/step - loss: 0.5689 - accuracy: 0.7362 -

val\_

loss: 0.2423 - val\_accuracy:

0.8926Epoch 3/10

14/14 [==============================] - 123s 9s/step - loss: 0.2231 - accuracy: 0.9197 -

val

\_loss: 0.1323 - val\_accuracy:

0.9669Epoch 4/10

14/14 [==============================] - 75s 5s/step - loss: 0.2170 - accuracy: 0.9128 -

val\_

loss: 0.1082 - val\_accuracy:

0.9669Epoch 5/10

14/14 [==============================] - 129s 10s/step - loss: 0.1918 - accuracy: 0.9151 -

va

l\_loss: 0.1145 - val\_accuracy:

0.9669Epoch 6/10

14/14 [==============================] - 111s 8s/step - loss: 0.1938 - accuracy: 0.9037 -

val

\_loss: 0.1030 - val\_accuracy: 0.9669

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Epoch  14/14 | 7/10  [==============================] | - 88s | 6s/step | - loss: | 0.1756 | - accuracy: | 0.9312 | - val\_ |
| loss: Epoch | 0.0831 - val\_accuracy: 0.9752 8/10 |  |  |  |  |  |  |  |
| 14/14  loss:  Epoch 14/14 | [==============================]  0.1073 - val\_accuracy: 0.9669 9/10 [==============================] | - 86s  - 77s | 6s/step  6s/step | * loss: * loss: | 0.1564  0.1480 | * accuracy: * accuracy: | 0.9404  0.9427 | * val\_ * val\_ |
| loss: Epoch  14/14 | 0.0754 - val\_accuracy: 0.9835 10/10  [==============================] | - 81s | 6s/step | - loss: | 0.1641 | - accuracy: | 0.9289 | - val\_ |
| loss: | 0.0601 - val\_accuracy: 0.9835 |  |  |  |  |  |  |  |

Out[11]: <keras.callbacks.History at 0x2546507bf10>

model**.**save("forest1.h5")

In [12]: model**.**save("forest1.h5")

In[13]: *#import load\_model from keras.model* **from** keras.models **import** load\_model *#import image class from keras*

**from** tensorflow.keras.preprocessing **import** image

*#import numpy* **import** numpy **as** np *#import cv2* **import** cv2

**from** tensorflow.keras.preprocessing **import** image

In [15]: *#load the saved model*

model **=** load\_model("forest1.h5")

*#load the saved model*

model **=** load\_model("forest1.h5")

In [16]: img**=**image**.**load\_img(r'C:\Users\devi\Downloads\archive\Dataset\Dataset\test\_set\with fire\skynx**=**image**.**img\_to\_array(img)

res **=** cv2**.**resize(x, dsize**=**(128, 128), interpolation**=**cv2**.**INTER\_CUBIC)

*#expand the image shape*

x**=**np**.**expand\_dims(res,axis**=**0)

img**=**image**.**load\_img(r'C:\Users\devi\Downloads\archive\Dataset\Dataset\test\_set\with fire\skynx**=**image**.**img\_to\_array(img)

res **=** cv2**.**resize(x, dsize**=**(128, 128), interpolation**=**cv2**.**INTER\_CUBIC)

*#expand the image shape*

x**=**np**.**expand\_dims(res,axis**=**0)

In [17]: pred**=**model**.**predict(x)

pred**=**model**.**predict(x)

1/1 [==============================] - 5s 5s/step

In [18]: pred

pred

Out[18]: array([[1.]], dtype=float32)

In[21]: **x\_train.class\_iundices**

**x\_train.class\_iundices**

Out[21]: {'forest': 0, 'with fire': 1}

In [24]: **if** (pred[0]**>**0.5):

print("forest with fire")

## else:

print("forest without fire")

**if** (pred[0]**>**0.5):

print("forest with fire")

**else**:

print("forest without fire")

forest with fire