

# **EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES**

## **Video Analysis**

## **Sending Alert Message**

<b>Date</b>	06 November 2022
<b>Team ID</b>	PNT2022TMID10687
<b>Project Name</b>	Emerging Methods for Early Detection of Forest Fires

### ***Importing The ImageDataGenerator Library***

```
import keras  
from keras.preprocessing.image import ImageDataGenerator
```

### ***Define the parameters/arguments for ImageDataGenerator class***

```
train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2,  
rotation_range=180, zoom_range=0.2, horizontal_flip=True)  
test_datagen=ImageDataGenerator(rescale=1./255)
```

### ***Applying ImageDataGenerator functionality to trainset***

```
x_train=train_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/train_set',  
target_size=(128,128), batch_size=32, class_mode='binary')
```

Found 436 images belonging to 2 classes.

### ***Applying ImageDataGenerator functionality to testset***

```
x_test=test_datagen.flow_from_directory(r'/content/drive/MyDrive/Dataset/test_set'  
, target_size=(128,128),batch_size=32, class_mode='binary') Found 121 images  
belonging to 2 classes.
```

### ***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')
```

### ***Initializing the model***

```
model=Sequential()
```

### ***Add CNN 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 Dense Layer***

```
#add hidden layer  
model.add(Dense(150,activation='relu'))
```

*#add                      output                      layer*

model.add(Dense(1,activation='sigmoid'))

### ***Configure the learning process***

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

### ***Train the model***

model.fit\_generator(x\_train,steps\_per\_epoch=14,epochs=10,validation\_data=x\_test,validation\_steps=4)

Epoch 1/10

14/14 [=====] - 205s 15s/step - loss: 2.7344 - accuracy: 0.7454 - val\_loss: 0.2016 - val\_accuracy: 0.9256

Epoch 2/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.8945 - val\_loss: 0.2290 - val\_accuracy: 0.9339

Epoch 3/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.8922 - val\_loss: 0.0524 - val\_accuracy: 0.9835

Epoch 4/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9174 - val\_loss: 0.1570 - val\_accuracy: 0.9421

Epoch 5/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9083 - val\_loss: 0.0767 - val\_accuracy: 0.9752

Epoch 6/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9335 - val\_loss: 0.0749 - val\_accuracy: 0.9752

Epoch 7/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9312 - val\_loss: 0.1264 - val\_accuracy: 0.9421

Epoch 8/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9266 - val\_loss: 0.0652 - val\_accuracy: 0.9835

Epoch 9/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9358 - val\_loss: 0.0567 - val\_accuracy: 0.9835

Epoch 10/10

14/14 [=====] - 20s 1s/step - loss: accuracy: 0.9404 - val\_loss: 0.0448 - val\_accuracy: 0.9917

0.3267 -

0.2991 -

0.2418 -

0.1984 -  
0.1643 -  
0.1538 -  
0.1732 -  
0.1514 -  
0.1445 -  
<keras.callbacks.History at 0x7f51fdf33610>

### ***Save The Model***

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

### ***Predictions***

```
#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
```

```
#load the saved model model =  
load_model("forest1.h5")
```

```
img=image.load_img(r'/content/drive/MyDrive/Dataset/test_set/forest/  
0.48007200_1530881924_final_forest.jpg') x=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) pred=  
model.predict(x)
```

```
1/1 [=====] - 0s 94ms/step pred array([[0.]],  
dtype=float32)
```

## ***OpenCV For Video Processing***

```
from tensorflow.keras.utils import load_img,img_to_array
```

```
while(1):
```

```
    sucess, frame= video.read()
```

```
    cv2.imwrite("image.jpg",frame)
```

```
    img=load_img("image.jpg",target_size=(64,64))
```

```
    x=img_to_array(img)
```

```
    x=np.expand_dims(x,axis=1)
```

```
    #predict_x=model.predict(x)
```

```
    #classes_x=np.argmax(predict_x,axis=1)
```

```
    pred=model.predict_classes(x)
```

```
    p=classes_x[0]
```

```
    print(classes_x)
```

```
    cv2.putText(frame,"predicted
```

```
class="+str(name[p]),(100,100),cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,0), 1)
```

```
    pred = model.predict_classes(x)
```

```
    if pred[0]==1:
```

```
        account_sid='AC7fbd9e1b65a166f13459d8eca7b664cf'
```

```
        auth_token='8e7e8e6672a8fb0a908ab3137560022d'
```

```
        client=Client(account_sid,auth_token)
```

```
        message=client.messages \
```

```
        .create(
```

```
            body='Forest Fire is detected, stay alert', from_='+18434385489',to='+91 95666  
05556')
```

```
            print(message.sid)
```

```
            print('Fire Detected')
```

```
            print('SMS sent!')
```

```
    else:
```

```

    print('No Danger')

    cv2.imshow("image",frame)

if cv2.waitKey(1) & 0xFF == ord('a'):

    break

video.release()

cv2.destroyAllWindows()

```

### ***Creating An Account In Twilio Service***

```

account_sid='AC7fbd9e1b65a166f13459d8eca7b664cf'
auth_token='8e7e8e6672a8fb0a908ab3137560022d'
client=Client(account_sid,auth_token)
message=client.messages \
.create(
    body='Forest Fire is detected, stay alert',
    from_='+18434385489',
    to='+91 95666 05556'
)
print(message.sid)

```

### ***Sending Alert Message***

```

while(1):
    sucess, frame= video.read() cv2.imwrite("image.jpg",frame)
    img=image.load_img("image.jpg",target_size=(64,64)) x=image.img_to_array(img)
    x=np.expand_dims(x,axis=0) pred=model.predict_classes(x) p=pred[0] print(pred)
    cv2.putText(frame,"predicted class="+str(name[p]),(100,100),

cv2.FONT_HERSHEY_SIMPLEX,1, (0,0,0), 1) pred = model.predict_classes(x) if
pred[0]==1:

account_sid='ACfb4e6d0e7b0d25def63044919f1b96e3'
auth_token='f9ae4fc4a617a527da8672e97eefb2d8'
client=Client(account_sid,auth_token) message=client.messages \

.create(
    body='Forest Fire is detected, stay alert', from_='+1 302 248 4366', to='+91
99400 12164'

)

```

```
print(message.sid) print('Fire Detected') print('SMS sent!')
```

```
else:
```

```
print('No Danger') cv2.imshow("image",frame) if
```

```
cv2.waitKey(1) & 0xFF == ord('a'): break
```

```
video.release() cv2.destroyAllWindows()
```