EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRE

SPRINT 2

Project Name	Emerging Methods for Early Detection of Forest Fires

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')
```

In [4]:

```
Found 436 images belonging to 2 classes.
```

```
#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 7/10
14/14 [==============================]         - 88s 6s/step - loss: 0.1756 - accuracy: 0.9312 - val_
loss: 0.0831 - val_accuracy: 0.9752
Epoch 8/10
14/14 [==============================] - 86s 6s/step - loss: 0.1564 - accuracy: 0.9404 - val_ loss:
0.1073 - val_accuracy: 0.9669
Epoch 9/10
14/14 [==============================]          - 77s 6s/step - loss: 0.1480 - accuracy: 0.9427 - val_
loss: 0.0754 - val_accuracy: 0.9835
Epoch 10/10
14/14 [==============================]         - 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