Emerging Methods for Early Detection of Forest Fires

MODEL BUILDING

TRAINING THE MODEL

Team ID	PNT2022TMID01865
Project Name	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/Da
taset/Training',  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/Data
set/Testing', 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=["
ac curacy"])
```

Train the model

```
model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_d
a ta=x_test,validation_steps=4)
Epoch 1/10
14/14 [==============================] - 97s 7s/step - loss:
1.3060 -
accuracy: 0.7775 - val_loss: 0.5513 - val_accuracy: 0.8512 Epoch 2/10
14/14 [==============================] - 26s 2s/step - loss:
0.3178 -
accuracy: 0.8807 - val_loss: 0.1299 - val_accuracy: 0.9421 Epoch 3/10
```

```
14/14 [==============================] - 26s 2s/step - loss:
0.2226 -
accuracy: 0.9106 - val_loss: 0.1311 - val_accuracy: 0.9421 Epoch 4/10
14/14 [==============================] - 31s 2s/step - loss:
0.1836 -
accuracy: 0.9174 - val_loss: 0.1129 - val_accuracy: 0.9339 Epoch 5/10
14/14 [==============================] - 30s 2s/step - loss:
0.1675 –
accuracy: 0.9243 - val_loss: 0.0925 - val_accuracy: 0.9669 Epoch 6/10
14/14 [==============================] - 26s 2s/step - loss:
0.1884 –
accuracy: 0.9289 - val_loss: 0.1287 - val_accuracy: 0.9339 Epoch 7/10
14/14 [==============================] - 28s 2s/step - loss:
0.1724 -
accuracy: 0.9335 - val_loss: 0.0926 - val_accuracy: 0.9752 Epoch 8/10
14/14 [==============================] - 26s 2s/step - loss:
0.1510 -
accuracy: 0.9404 - val_loss: 0.0757 - val_accuracy: 0.9752
Epoch 9/10 14/14 [==============================] - 26s
2s/step - loss: 0.1732
accuracy: 0.9174 - val_loss: 0.0537 - val_accuracy: 0.9835
Epoch 10/10 14/14 [==============================] - 26s
2s/step - loss: 0.1546
accuracy: 0.9312 - val_loss: 0.0573 - val_accuracy: 0.9835
```