Emerging Methods for Early Detection of Forest Fire

Submitted by

Team no: 4

G. D Chandra Shekhar

(19BCE10207)

Durvankur Dama

(19BCG10052)

Randhir Prakash

(19BCG10060)

Rajveer Kumar

(19BCG10092)

1.INTRODUCTION

1.1 overview

Forests, which are diverse centres of flora and wildlife and create 1/3 of the world's oxygen, are at risk of forest fires, both natural and man-made. The precaution of averting such a massive devastating flare can save many animals and the environment. Protecting forests before they are harmed is a method of repaying Mother Nature's everlasting gift.

Wildfires are one of the biggest catastrophes faced by our society today causing irrevocable damages. These forest fires can be man-made or caused bymother nature by different weather conditions, torrential winds. These fires cause damages not only to the environment they also destroy vast homes and property.

1.2 purpose

Forest fires have become a major threat around the world, causing many negative impacts on human habitats and forest ecosystems. Climatic changes and the greenhouse effect are some of the consequences of such destruction. Interestingly, a higher percentage of forest fires occur due to human activities.

The goal of the project is to develop a forest fire detection system that can identify forest fires in their early phases.

2.literature survey

2.1 existing problem

Every year, there are an estimated 340,000 premature deaths from respiratory and cardiovascular issues attributed to wildfire smoke.

The increasing frequency and severity of wildfires pose a growing threat to biodiversity globally. Individuals, companies and public authorities bear great economic costs due to fires. In order to reduce all these, we need to detect the forest fire at an early stage and prevent it.

Some of the existing solutions for solving this problem are:

Technology

The present technology includes particle and smoke detection systems, which are commonly used in facilities and families. These systems detect moisture in a space and determine whether the current atmosphere is safe or if an alarm should be triggered. The same way that a fire alarm works by spraying water throughout the room to put out the fire.

Fire fighter

To tackle fire problems, highly trained humans are used. Firefighters employ techniques and trucks to suppress forest fires throughout the conditions.

The priority of a firefighter is to protect people and reduce the number of people killed or injured by fire. Firefighting and property damage are the second and third priorities, respectively.

2.2 proposed solution

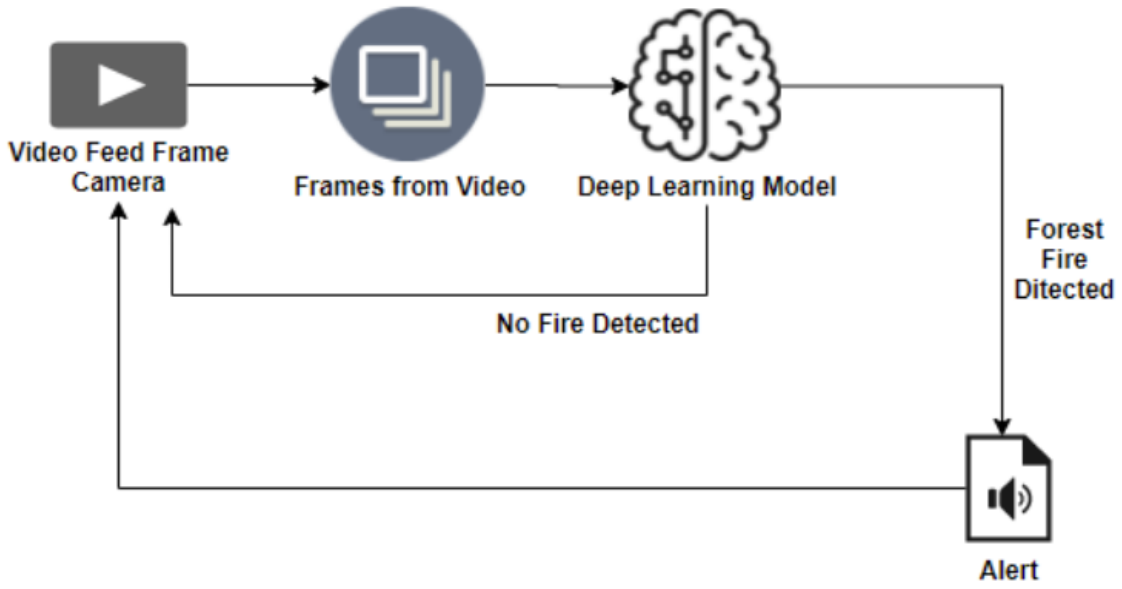
The following paper describes the system to detect fire before becoming a big flame of destruction:

1. To build a system to detect the fire in woods through image processing.
2. To overcome the physically and molecular dynamic to detect fire for faster response.
3. A conventional neural network is being used to develop a model used to train through various images. This system will help to detect fire with before response system to prevent huge destruction.

3.Theoritical analysis

3.1 Block diagram

Architecture:



3.2 hardware/software designing

Hardware requirements:

Operating System	Windows, Mac, Linux
CPU (for training)	Multi Core Processors (i3 or above/equivalent)
GPU (for training)	NVIDIA AI Capable / Google's TPU
WebCam	Integrated or External with FullHD Support

Software Requirements:

Python	v3.9.0 or Above
Python Packages	flask, tensorflow, opencv-python, keras, numpy, pandas, virtualenv, pillow
Web Browser	Mozilla Firefox, Google Chrome or any modern web browser
IBM Cloud (for training)	Watson Studio- Model Training & Deployment as Machine Learning Instance

4.Experimental Investigations

Training and Testing usingDataset Provided:

In [3]:

```
#import keras library
import keras
#import ImageDataGenerator class from keras
from keras.preprocessing.image import ImageDataGenerator
```

In [4]:

```
#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)
```

In [5]:

```
#: Applying ImageDataGenerator functionality to trainset.
#give the path of training images folder
x_train = train_datagen.flow_from_directory(r'C:\Users\shekh\OneDrive\Desktop\Smart Externs
                                            target_size = (128,128),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 563 images belonging to 2 classes.
```

In [6]:

```
#: Applying ImageDataGenerator functionality to testset.
#give the path of testing images folder
x_test = test_datagen.flow_from_directory(r'C:\Users\shekh\OneDrive\Desktop\Smart Externshi
                                          target_size = (128,128),
                                          batch_size = 32,
                                          class_mode = 'binary')
```

```
Found 121 images belonging to 2 classes.
```

5.Flowchart

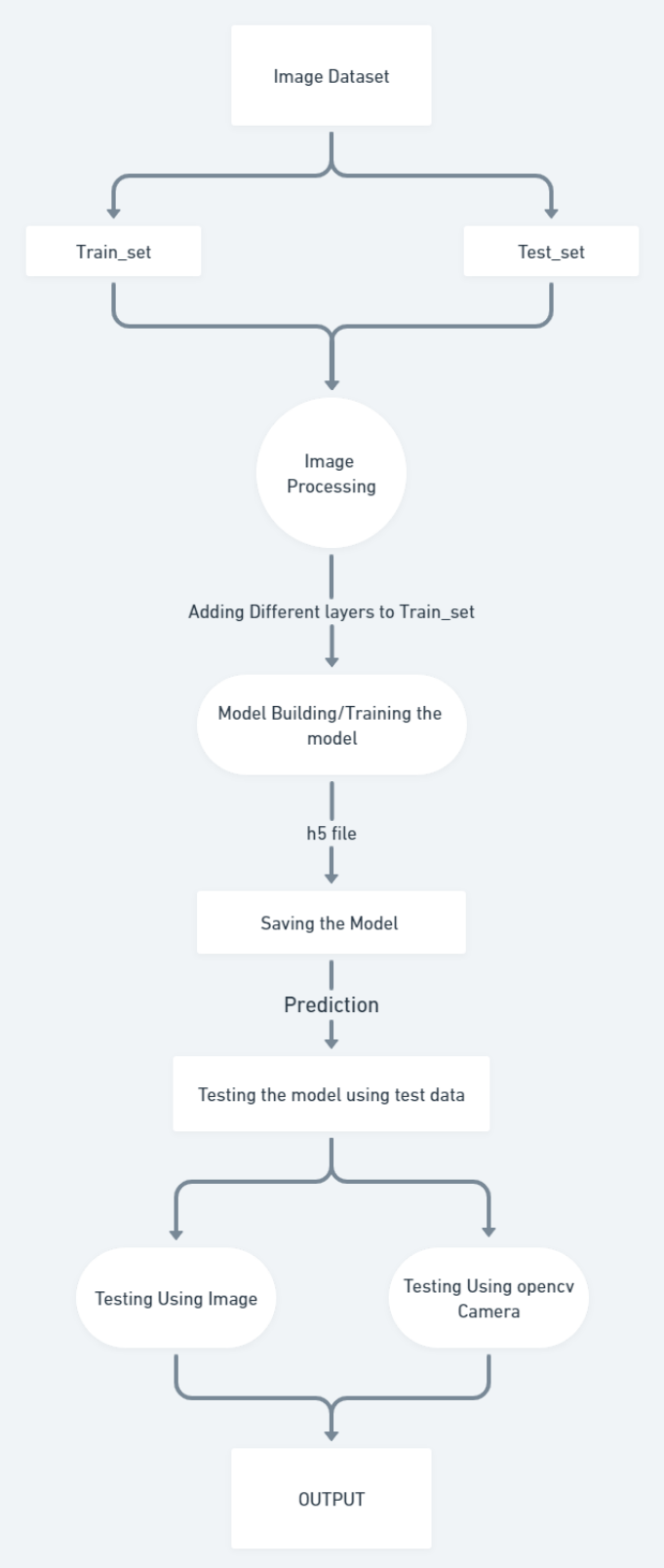


Image Dataset
Train_set
Test_set
Image Processing
Adding Different layers to Train_set
Model Building/Training the model
h5 file
Saving the Model
Prediction
Testing the model using test data
Testing Using Image
Testing Using opencv Camera
OUTPUT

6.Result

The proposed procedurewas implemented and tested with set of images. The sets of images 324 images of forest with fire and normal for training database and set of 324 images of forest fire and normal for testing database. Once the model recognises the appropriate result on the screen.

Some examples images of the output are provided below:

Prediction

In [19]:
```
from keras.models import load_model
from keras.preprocessing import image
import numpy as np
import cv2
```

In [20]:
```
model = load_model("forest1.h5")
```

In [21]:
```
img = image.load_img(r'C:\Users\shekh\OneDrive\Desktop\Smart Externship\Project\AI_Project\Dataset\test_set\with fire\Untitled_de
x = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
```

In [22]:
```
img
```

Out[22]:

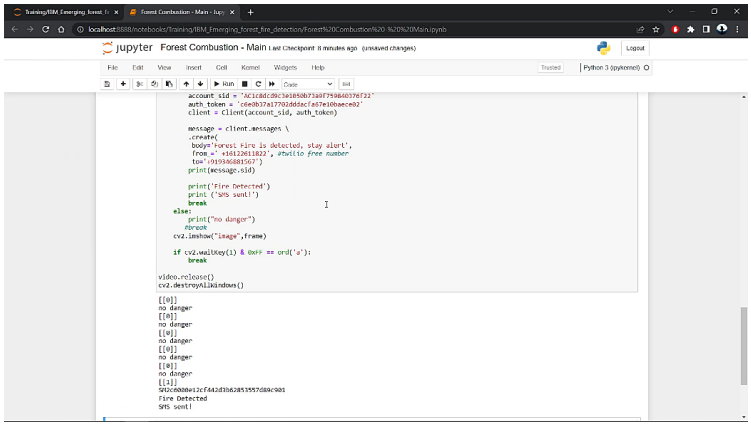
In [23]:
```
pred = model.predict_classes(x)
```

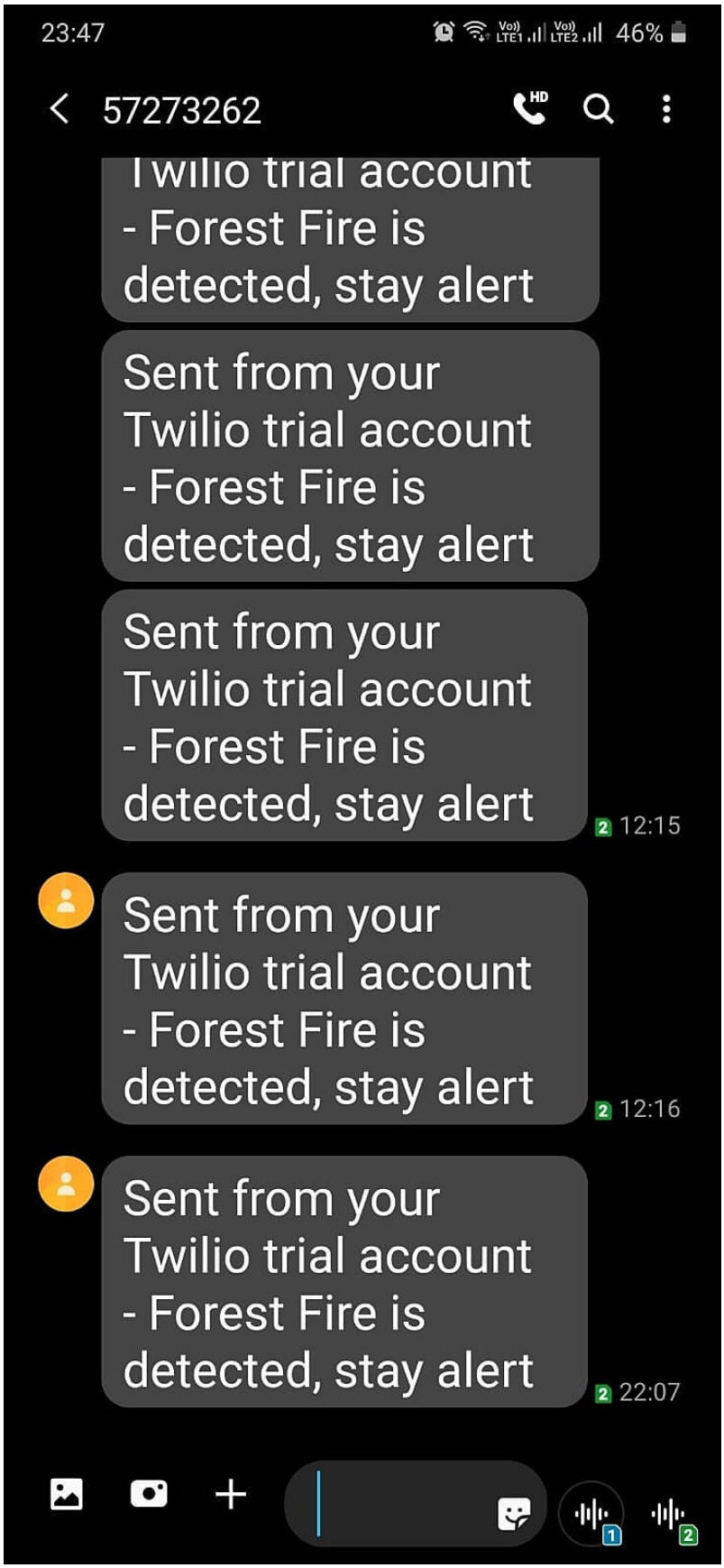
In [24]:
```
pred
```

Out[24]:
```
array([[1]])
```

In [29]:
```
print(x_train.class_indices)
```

```
{'forest': 0, 'with fire': 1}
```




23:47
46%
57273262
Twilio trial account - Forest Fire is detected, stay alert
Sent from your Twilio trial account - Forest Fire is detected, stay alert
Sent from your Twilio trial account - Forest Fire is detected, stay alert
12:15
Sent from your Twilio trial account - Forest Fire is detected, stay alert
12:16
Sent from your Twilio trial account - Forest Fire is detected, stay alert
22:07

7. Advantage and disadvantages

- ### Advantages:

1. The proposed model can be used in combination with a night camera and a thermal camera in a forest to identify tiny fire signs.

2. More datasets and images can be used to train for a more accurate outcome when detecting flame destruction ability.

3. model can be implemented in mobile applications for camping experience enthusiasts.

- ### Disadvantage:

1. The model works for limited information.
2. The accuracy is low because to the limited quantity/quality of photos in the dataset, but this may easily be increased by changing the dataset.
3. The small amount of fire amount detection can also cause to trigger the alarm.

8.Applications:

1. will contribute to surveillance technology that improves the accuracy and predictability of fire detection.

2. able to detect the fire forest more precisely, as well as some forest plants and wildlife.

3.Detect the amount of dangers that should be treated and those that should not. extra assistance in contacting fire fighters for assistance system.

9.Conclusion

Forest fires are a major cause of rain forest and savanna degradation. This model will aid in minimising destruction by anticipating it to the system, allowing individuals to react more quickly

and prevent it.

The proposed methodology would deconstruct the threat to the environment by converting the image collected into signals that will trigger an alarm.

This system transmits video images to a model, which recognises them and determines whether or not to send a threat alert. The model extracts data from video feeds and defines image processing into RGB data for signal response modelling.

10.future scope

The availability of fire-fighting technology brings us one step closer to new AI for detection and security in the forest and at home. With the addition of a motion sensor, the technology can simply expand to compact decision-making with the addition of new software and hardware.

The system is utilized as a drone and surveillance system UAV to expand the surveillance area and detect heat signatures in order to identify human from fire plasma signatures.

11.BIBLOGRAPY

1. Environment Setup:https://www.youtube.com/watch?v=5mDYijMfSzs

2. Forest fire Dataset: https://drive.google.com/drive/folders/1vq8TRFWE7WH7_-dsqKAmvjJAsaxx-kPQ?usp=sharing

3. Keras Image Processing Doc: https://keras.io/api/preprocessing/image/

4. Keras ImageDataset From Directory Doc: https://keras.io/api/preprocessing/image/#imagedatasetfromdirectory-function

5. CNN using Tensorflow: https://www.youtube.com/watch?v=umGJ30-15_A

6. OpenCV Basicsof Processing Image:https://www.youtube.com/watch?v=mjKd1Tzl70I

7. Flask Basics:https://www.youtube.com/watch?v=lj4I_CvBnt0

8. IBM Academic Partner Account Creation: https://www.youtube.com/watch?v=x6i43M7BAqE

9. CNN Deployment and Download through IBM Cloud: https://www.youtube.com/watch?v=BzouqMGJ41k

12. Appendix

Source Codefor Model Trainingand Saving:

In [3]:

```
#import keras library
import keras
#import ImageDataGenerator class from keras
from keras.preprocessing.image import ImageDataGenerator
```

In [4]:

```
#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)
```

In [5]:

```
#: Applying ImageDataGenerator functionality to trainset.
#give the path of training images folder
x_train = train_datagen.flow_from_directory(r'C:\Users\shekh\OneDrive\Desktop\Smart Externs
                                            target_size = (128,128),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 563 images belonging to 2 classes.
```

In [6]:

```
#: Applying ImageDataGenerator functionality to testset.
#give the path of testing images folder
x_test = test_datagen.flow_from_directory(r'C:\Users\shekh\OneDrive\Desktop\Smart Externshi
                                          target_size = (128,128),
                                          batch_size = 32,
                                          class_mode = 'binary')
```

```
Found 121 images belonging to 2 classes.
```

In [8]:

```
#intializing the model
model =Sequential()
```

In [9]:

```
#add convolutional layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

In [10]:

```
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [11]:

```
#add flatten layer
model.add(Flatten())
```

In [12]:

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

In [13]:

```
#add hidden layermodel.add(Dense(output_dim=64,init='uniform',activation='relu'))
```

In [14]:

```
#add output layer
model.add(Dense(kernel_initializer='uniform',activation='sigmoid',units=1))
```

IBM Model Training & Download Code:

MODEL BUILDING

In [12]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
```

In [13]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [14]:
```
model = Sequential()
```

In [15]:
```
#add convolutional layer
model.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
```

In [16]:
```
#add maxpooling layer
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [17]:
```
#add flatten layer
model.add(Flatten())
```

In [18]:
```
#add hidden layer
model.add(Dense(kernel_initializer='uniform',activation='relu',units=150))
```

In [19]:
```
#add hidden layer
model.add(Dense(units=64,kernel_initializer='uniform',activation='relu'))
```

In [20]:
```
#add output layer
model.add(Dense(kernel_initializer='uniform',activation='sigmoid',units=1))
```

Training:

```
In [24]: model.fit_generator(x_train,steps_per_epoch=14,epochs=10,validation_data=x_test,validation_steps=4)

         Epoch 1/10
         14/14 [==============================] - 22s 2s/step - loss: 0.5460 - accuracy: 0.7110 - val_loss: 0.2490 - val_accuracy: 0.900
         8
         Epoch 2/10
         14/14 [==============================] - 20s 1s/step - loss: 0.2957 - accuracy: 0.8853 - val_loss: 0.1399 - val_accuracy: 0.933
         9
         Epoch 3/10
         14/14 [==============================] - 20s 1s/step - loss: 0.2964 - accuracy: 0.8739 - val_loss: 0.1181 - val_accuracy: 0.950
         4
         Epoch 4/10
         14/14 [==============================] - 21s 1s/step - loss: 0.2385 - accuracy: 0.9014 - val_loss: 0.1022 - val_accuracy: 0.975
         2
         Epoch 5/10
         14/14 [==============================] - 20s 1s/step - loss: 0.1972 - accuracy: 0.9151 - val_loss: 0.0878 - val_accuracy: 0.975
         2
         Epoch 6/10
         14/14 [==============================] - 20s 1s/step - loss: 0.1916 - accuracy: 0.9266 - val_loss: 0.1730 - val_accuracy: 0.909
         1
         Epoch 7/10
         14/14 [==============================] - 20s 1s/step - loss: 0.2449 - accuracy: 0.8968 - val_loss: 0.0702 - val_accuracy: 0.991
         7
         Epoch 8/10
         14/14 [==============================] - 20s 1s/step - loss: 0.1923 - accuracy: 0.9266 - val_loss: 0.1240 - val_accuracy: 0.950
         4
         Epoch 9/10
         14/14 [==============================] - 20s 1s/step - loss: 0.1806 - accuracy: 0.9312 - val_loss: 0.0938 - val_accuracy: 0.966
         9
         Epoch 10/10
         14/14 [==============================] - 20s 1s/step - loss: 0.1633 - accuracy: 0.9427 - val_loss: 0.0772 - val_accuracy: 0.983
         5

Out[24]: <keras.callbacks.History at 0x7fe2206a48b0>

In [25]: model.save('forestfire.h5')

In [26]: !tar -zcvf Emerging-Forest-Fires.tgz forestfire.h5

         forestfire.h5
```

Standard Reference for forest fire: