**Final Code**

# prerequisities

# conda create –n tensorflow python=3.5

# activate tensorflow

# pip install –ignore—installed –upgrade tensorflow

!pip install tensorflow

!pip install opencv-python

!pip install opencv-contrib-python

import tensorflow as tf

import numpy as np

from tensorflow import keras

import os

import cv2

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.preprocessing import image

import matplotlib.pyplot as plt

#its important to split the training and testing - Data set is stored in the path specified.

train = ImageDataGenarator(rescale=1/255)

test = ImageDataGenerator(rescale=1/255)

train\_dataset = train.flow\_from\_directory("D:/archive/fire\_dataset/Training and validation/",

target\_size=(150,150),

batch\_size =32,

class\_mode = 'binary')

train\_dataset = test.flow\_from\_directory("D:/archive/fire\_dataset/Testing/",

target\_size=(150,150),

batch\_size =32,

class\_mode ='binary')

test\_dataset.class\_indices

# we shall build the model here!

# simple CNN shall do the task, you can try other tech as well.

# Try with other activation functions also.

model = keras.Sequential()

model.add(keras.layers.Conv2D(32,(3,3),activation='relu',input\_shape=(150,150,3)))

model.add(keras.layers.MaxPool2D(2,2))

model.add(keras.layers.Conv2D(64,(3,3),activation='relu'))

model.add(keras.layers.MaxPool2D(2,2))

model.add(keras.layers.Conv2D(128,(3,3)),activation='relu'))

model.add(keras.layers.MaxPool2D(2,2))

model.add(keras.layers.Conv2D(128,(3,3)),activation='relu'))

model.add(keras.layers.MaxPool2d(2,2))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(512,activation='relu'))

model.add(keras.layers.Dense(1,activation='sigmod'))

# It is time to compile the model, let us compile.model.compile(optimizer='adam',loss='binary\_crossentropy',metrics=['accuracy'])

# let's get the model fit.

r = model.fit(train\_dataset, epochs = 7, validation\_data = test\_dataset)

# Epochs you can vary!

# Can we work on the testing dataset,the prediction happen here.

predictions = model.predict(test\_dataset)

predictions = np.round(predictions)

predictions

print(len(prediction))

# it's time to plot it! lets plot loss.

import matplib.pyplot as plt

plt.plot(r.history['loss'], label='loss')

plt.plot(r.history['val\_loss'], label='val\_loss')

plt.legend()

# This helps in taking individual images from the Dataset, load and check results.

def predictImage(filename):

img1 = image.load\_img(filename,target\_size=(150,150))

plt.imshow(img1)

Y = image.img\_to\_array(img1)

X = np.expand\_dims(Y,axis=0)

val = model.predict(X)

print(val)

if val ==1:

plt.xlabel("No fire",fontsize=30)

elif val == 0:

plt.xlabel("fire",fontsize=30)

predictImage("")