```
#!/usr/bin/env python

# coding: utf-8


# In[ ]:



import drive

drive. mount('/content/drive')

get_ipython().system('unzip drive/My\\Drive/dataset.zip')



# In[ ]:



# import necessarylib.

from tensorflow.keras.preprocessing.image import ImageDataGenerator



# In[ ]:



#image Data Agumentation


#setting parameter for Image Data agumentation to the traing data
```

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


#Image Data agumentation to the testing data

test_datagen=ImageDataGenerator(rescale=1./255)



# In[ ]:



#Loading our data and performing data agumentation

#performing data agumentation to train data


x_train = train_datagen.flow_from_directory('/content/dataset/train_set',target_size=(64, 64),
batch_size=5, color_mode='rgb',class_mode='categorical')


#performing data agumentation to test data


x_test = test_datagen.flow_from_directory('/content/dataset/test_set',target_size=(64, 64),
batch_size=5, color_mode='rgb',class_mode='categorical')



# In[4]:



#Importing Neccessary Libraries
```

```
import numpy as np #used for numerical analysis

import tensorflow #open source used for both ML and DL for computation

from tensorflow.keras.models import Sequential #it is a plain stack of Layers

from tensorflow.keras import layers #A Layer consists of a tensor-in tensor-out computation function

#Dense layer is the regular deeply connected neural network Layer

from tensorflow.keras.layers import Dense, Flatten

#Faltten-used fot flattening the input or change the dimension

from tensorflow.keras.layers import Conv2D, MaxPooling2D #Convolutional Layer

#MaxPooling20-for downsampling the image

from keras.preprocessing.image import ImageDataGenerator


# In[6]:


classifier=Sequential()
```

```
# First convolution layer and pooling

classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Second convolution layer and pooling

classifier.add(Conv2D(32, (3, 3), activation='relu'))

# input_shape is going to be the pooled feature maps from the previous convolution I

classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the Layers

classifier.add(Flatten())


# In[7]:


# Adding a fully connected Layer
classifier.add(Dense (units=128, activation='relu'))
classifier.add(Dense (units=4, activation='softmax'))
```

```
# softmax for more than 2
classifier. summary()


# In[8]:


#Compiling the model

# Compiling the CNN

# categorical_crossentropy for more than 2

classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


# In[10]:


# Save the model

classifier.save('disaster.h5')

model_json = classifier.to_json()
```

```
with open("model-bw.json", "w") as json_file:

 json_file. write(model_json)



# In[21]:



from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("disaster.h5") #Loading the model



# In[14]:



from tensorflow.keras.preprocessing import image


import numpy as np
img = image.load_img('/content/dataset/test_set/Flood/1009.jpg', target_size=(64,64))
img
#Loading of the image



# In[13]:
```

```
x= image.img_to_array(img)

x

#image to array



# In[15]:



x = np.expand_dims(x,axis = 0)

x

#changing the shape



# In[16]:



from tensorflow.keras.preprocessing import image


import numpy as np

img = image.load_img('/content/dataset/test_set/Flood/1009.jpg', target_size=(64,64))

x= image.img_to_array(img)

x = np.expand_dims(x,axis = 0)

pred = np.argmax(model.predict(x))
```

```
Output=['earthquake','cyclone','flood','wildfire']

Output[pred]

#predicting the class




# In[ ]:




get_ipython().system('pip install flask')

from flask import Flask, render_template, request


# Flask-It is our framework which we are going to use to r #request-for accessing file which was uploaded by the user #import operator


import cv2

 # opencv library


from tensorflow.keras.models import load_model

#to load our


import numpy as np


#import os

from werkzeug.utils import secure_filename


app = Flask (__name__)
```

```
template_folder="templates"
# initializ # Loading the model
model=load_model('disaster.h5')
print("Loaded model from disk")

@app.route('/', methods=['GET'])
def index():
    return render_template('home.html')
@app.route('/home', methods=['GET'])
def home():
    return render_template('home.html') @app.route('/intro', methods=['GET'])
def about():
    return render_template('intro.html')
@app.route('/upload', methods=['GET', 'POST'])
def predict():
    cap= cv2.VideoCapture (0)
while True:
       cap= cv2.VideoCapture (0)
       _, frame =  cap . read()
 #capturing the video frame values #Simulating mirror image
       frame = cv2.flip(frame, 1)

#Loop over frames from the video file stream
       while True:
  # read the next frame from the file
```

```
          (grabbed, frame) = cap.read()

# if the frame was not grabbed, then we have reach # of the stream

          if not grabbed:

            break

# if the frame dimensions are empty, grab them

          if W is None or H is None:

            (H, W) = frame.shape[:2]

# clone the output frame, then convert it from BGI # ordering and resize the frame to a fixed
224x224

output = frame.copy()

frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

frame = cv2.resize(frame, (64, 64))

#frame = frame.astype(”float32”)

x=np.expand_dims (frame, axis=0)

result = np.argmax (model.predict(x), axis=-1)
```

```
index=['Cyclone', 'Earthquake', 'Flood', 'Wildfire']

result=str(index [result[0]])

#print (result)

#result=result.tolist()

cv2.putText(output, "activity: {}".format(result), (10, 120), cv2.FONT_HERSHEY_PLAIN, 1,
(0,255,255), 1)
#playaudio ("Emergency it is a disaster")

cv2.imshow("Output", output)
key = cv2.waitKey(1) & 0xFF

# if the 'q' key was pressed, break from the loop
if key ==  ord("q"):
        break

# release the file pointers
print("[INFO] cleaning up...")
vs.release()
return render_template("upload.html")

if  __name__ == " __main__":
    app.run(host='0.0.0.0', port=8000, debug=False)
```

```
# In[ ]:
```