

PROJECT DEVELOPMENT PHASE

SPRINT-III

Date	12 th November 2022
Team ID	PNT2022TMID54341
Project Name	Natural Disaster Intensity Analysis and Classification using Artificial Intelligence

DETECTION AND ANALYSIS OF DATA:

After Testing and Training the model, data which given in dataset are analysed and visualised effectively to detect the Disaster Type. Using webcam, it can capture image or video stream of Disaster, to detect and analyse the type of Disaster.

```
Inserting necessary libraries

In [1]: 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
#Flatten-used for flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D,MaxPooling2D #Convolutional layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator

Using TensorFlow backend.

In [2]: tensorflow.__version__
Out[2]: '2.5.0'

In [3]: tensorflow.keras.__version__
Out[3]: '2.5.0'

Image Data Augmentation

In [4]: #setting parameter for Image Data augmentation to the training data
train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
#Image Data augmentation to the testing data
test_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
```

MODEL BUILDING:

Building a Model with web application named “FLASK”, model building process consist several steps like,

- Import the model building Libraries
- Initializing the model

- Adding CNN Layers
- Adding Hidden Layer
- Adding Output Layer
- Configure the Learning Process
- Training and testing the model

all the above processes are done and saved in a model.

Inserting necessary libraries

```
In [1]: 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
#Flatten-used for flattening the input or change the dimension
from tensorflow.keras.layers import Conv2D,MaxPooling2D #Convolutional Layer
#MaxPooling2D-for downsampling the image
from keras.preprocessing.image import ImageDataGenerator

Using TensorFlow backend.
```

```
In [2]: tensorflow.__version__
```

```
Out[2]: '2.5.0'
```

```
In [3]: tensorflow.keras.__version__
```

```
Out[3]: '2.5.0'
```

Image Data Augmentation

```
In [4]: #setting parameter for Image Data augmentation to the training data
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
#Image Data augmentation to the testing data
test_datagen=ImageDataGenerator(rescale=1./255)
```

```
# import the necessary packages
from flask import Flask,render_template,request
# Flask-It is our framework which we are going to use to run/serve our application.
#request-for accessing file which was uploaded by the user on our application.
#import operator
import cv2 # opencv library
from tensorflow.keras.models import load_model#to load our trained model
import numpy as np
import os
from werkzeug.utils import secure_filename
#from playsound import playsound
#from gtts import gTTS
...
def playaudio(text):
    speech=gTTS(text)
    print(type(speech))
    speech.save("output1.mp3")
    playsound("output1.mp3")
    return
...
app = Flask(__name__,template_folder="templates") # initializing a flask app
# loading the model
model=load_model(r'C:\Users\user\Desktop\IBM\Flask\templates\disaster.h5')
print("Loaded model from disk")

app=Flask(__name__,template_folder="templates")
@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'])
```