PROJECT DEVELOPMENT PHASE

SPRINT-II

Date	08 November 2022
Team ID	PNT2022TMID27138
Project Name	Natural Disaster Intensity Analysis and Classification using Artificial Intelligence

INSERTING NECESSARY LIBRARIES:

Numpy: It is an open source numerical python library.

Scikit-learn: It is a machine learning library for python.

OpenCV: OpenCV is a library of programming functions mainly aimed at real-time

computer vision.

Flask: Web framework used for building web application.

```
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
#Faltten-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'
```

LOADING DATA AND PERFORMING DATA AUGUMENTATION:

Loading the data into the Jupyter notebook by using RR dataset path.

```
Loading our data and performing Data Augumentation

In [5]: #performing data agumentation to train data
x_train = train_datagen.flow_from_directory(r'C:\Users\ELCOT\Downloads\projest\lbm\dataset\train_set',target_size=(64, 64),batch_color_mode='rgb',class_mode='categorical')

#performing data agumentation to test data
x_test = test_datagen.flow_from_directory(r'C:\Users\ELCOT\Downloads\projest\lbm\dataset\test_set',target_size=(64, 64),batch_siz_color_mode='rgb',class_mode='categorical')

# Found 742 images belonging to 4 classes.

In [6]: print(x_train.class_indices)#checking the number of classes
{'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}

In [7]: print(x_test.class_indices)#checking the number of classes
{'Cyclone': 0, 'Earthquake': 1, 'Flood': 2, 'Wildfire': 3}

In [8]: from collections import Counter as c
c(x_train .labels)

Out(8]: Counter({0: 220, 1: 156, 2: 198, 3: 168})
```

CREATING THE MODEL:

Creating the Model a Classifier Sequential. Classifier is a machine learning algorithm that determines the class of the input element based on the set of the feature. In this model using convolution2D function. Convolution2D parameter is an number of filters that convolution layer will be learn from. Then we will be using MaxPooling2D function. Then, using a Flatten() function that flatten the multidimensional input denser into the denser.

Creating the Model

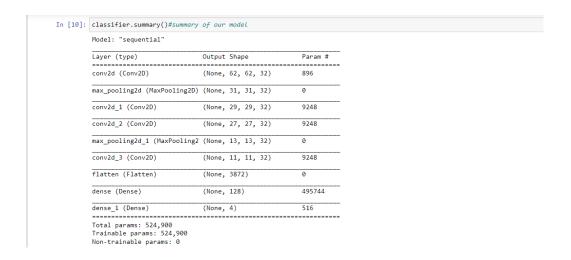
```
In [9]: # Initializing the CNW
classifier = Sequential()

# First convolution layer and poolingo
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
# 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 layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))

# Flattening the layers
classifier.add(Flatten())

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

Using classifier.summary() function summary of our model



COMPILING THE MODEL:

The model is compiled using the following code.

```
In [11]: # Compiling the CNN
# categorical_crossentropy for more than 2
classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

FITTING THE MODEL:

Fitting the Model with 70 epoch.

SAVING THE MODEL:

Saving the Model as disaster.h5. disaster.h5 file is used to find the image classification files. Model.json represents that Jason stands for JavaScript object rotation, Jason is a lite weight data format used for data inserting between multiple different language.

PREDICTING RESULTS:

Loading model from the tensorflow keras models and loading the image then converting image into array. Then predicting our model.