PROJECT DEVELOPMENT PHASE SPRINT -II

Date	7-NOV-2022
Team ID	PNT2022TMID47175
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.

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
#Faltten-used fot 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\ibm\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\ibm\dataset\test_set',target_size=(64, 64),batch_si
                                                  color_mode='rgb',class_mode='categorical')
```

```
Found 742 images belonging to 4 classes.
Found 198 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 CNN
        classifier = Sequential()

        # First convolution layer and poolingo
        classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
        classifier.add(MaxPooling2D(pool_size=(2, 2)))
        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=4, activation='softmax')) # softmax for more than 2
```

Using classifier.summary() function summary of our model

```
In [10]: classifier.summary()#summary of our model
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 27, 27, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 32)        9248
_________________________________________________________________
flatten (Flatten)            (None, 3872)              0
_________________________________________________________________
dense (Dense)                (None, 128)               495744
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 516
=================================================================
Total params: 524,900
Trainable params: 524,900
Non-trainable params: 0
```

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.

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Markdown

```
Epoch 65/70
149/149 [==============================] - 63s 421ms/step - loss: 0.0427 - accuracy: 0.9852 - val_loss: 1.8291 - val_accurac
y: 0.7374
Epoch 66/70
149/149 [==============================] - 62s 415ms/step - loss: 0.0574 - accuracy: 0.9838 - val_loss: 1.7194 - val_accurac
y: 0.7424
Epoch 67/70
149/149 [==============================] - 62s 415ms/step - loss: 0.0434 - accuracy: 0.9879 - val_loss: 1.8268 - val_accurac
y: 0.7374
Epoch 68/70
149/149 [==============================] - 62s 415ms/step - loss: 0.0796 - accuracy: 0.9717 - val_loss: 1.7385 - val_accurac
y: 0.7323
Epoch 69/70
149/149 [==============================] - 64s 427ms/step - loss: 0.0287 - accuracy: 0.9865 - val_loss: 2.0659 - val_accurac
y: 0.7424
Epoch 70/70
149/149 [==============================] - 63s 424ms/step - loss: 0.0242 - accuracy: 0.9933 - val_loss: 3.4165 - val_accurac
y: 0.6818
```

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.

Saving the Model

In [13]:
```
# Save the model
classifier.save('disaster.h5')
```

In [14]:
```
model_json = classifier.to_json()
with open("model-bw.json", "w") as json_file:
    json_file.write(model_json)
```

PREDICTING RESULTS:

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

In [15]:
```
from tensorflow.keras.models import load_model
from keras.preprocessing import image
model = load_model("disaster.h5") #loading the model for testing
```

In [ ]:
```
```

In [16]:
```
img = image.load_img(r'C:\Users\ELCOT\Downloads\projest\ibm\dataset\test_set\Cyclone\870.jpg',grayscale=False, target_size= (64,6
x = image.img_to_array(img)#image to array\n",
x = np.expand_dims(x,axis = 0)#changing the shape\n",
pred = model.predict_classes(x)#predicting the classes\n",
pred
```

```
C:\Users\ELCOT\anaconda3\lib\site-packages\tensorflow\python\keras\engine\sequential.py:455: UserWarning: `model.predict_classes()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

Out[16]:
```
array([0], dtype=int64)
```

In [17]:
```
index=['Cyclone','Earthquake','Flood','Wildfire']
result=str(index[pred[0]])
result
```

Out[17]:
```
'Cyclone'
```