# PROJECT DEVELOPMENT PHASE SPRINT -II

Date	17 - NOV - 2022
Team ID	PNT2022TMID49204
Project	Natural Disaster Intensity Analysis and Classification using
Name	ArtificialIntelligence

## <u>INSERTING NECESSARY LIBRARIES:</u>

**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-timecomputer 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

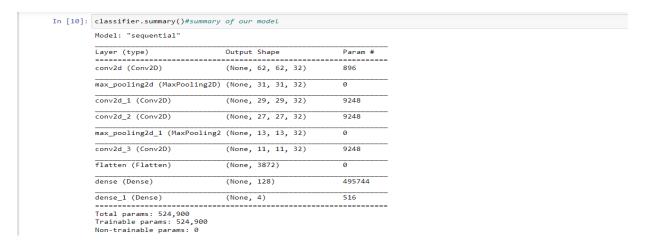
## <u>CREATING THE MODEL:</u>

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



## **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.

```
Cell Kernel Widgets
        Insert
                                                Trusted
                                                     Python 3 (ipykernel) O
B + % € B ↑ ↓ ▶ Run ■ C ▶ Markdown
      בטסרוו מס//מ
      y: 0.7374
      149/149 [=========] - 62s 415ms/step - loss: 0.0574 - accuracy: 0.9838 - val_loss: 1.7194 - val_accurac
      y: 0.7424
      v: 0.7374
      y: 0.7323
      Epoch 69/70
      y: 0.7424
      Epoch 70/70
      149/149 [=========== ] - 63s 424ms/step - loss: 0.0242 - accuracy: 0.9933 - val_loss: 3.4165 - val_accurac
      v. A 6818
```

## **SAVING THE MODEL:**

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

## Saving the Model

#### **PREDICTING RESULTS:**

Loading model from the tensorflow keras models and loading the imagethen 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 \backslash n",
          pred
          s() is deprecated and will be removed after 2021-01-01. Please use instead: ^* inp.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]])
Out[17]: 'Cyclone'
```