***Natural Disasters Intensity Analysis And Classification Using AI***

## Team Leader: Yenni Manivaraprasad, Sri Sivani college of Engineering

## Team Member: Batna janardhana, Sri Sivani college of Engineering

## Team Member: Mandangi Kiran, Sri Sivani college of Engineering

## Team Member: Yandava Santosh ,Sri Sivani college of Engineering

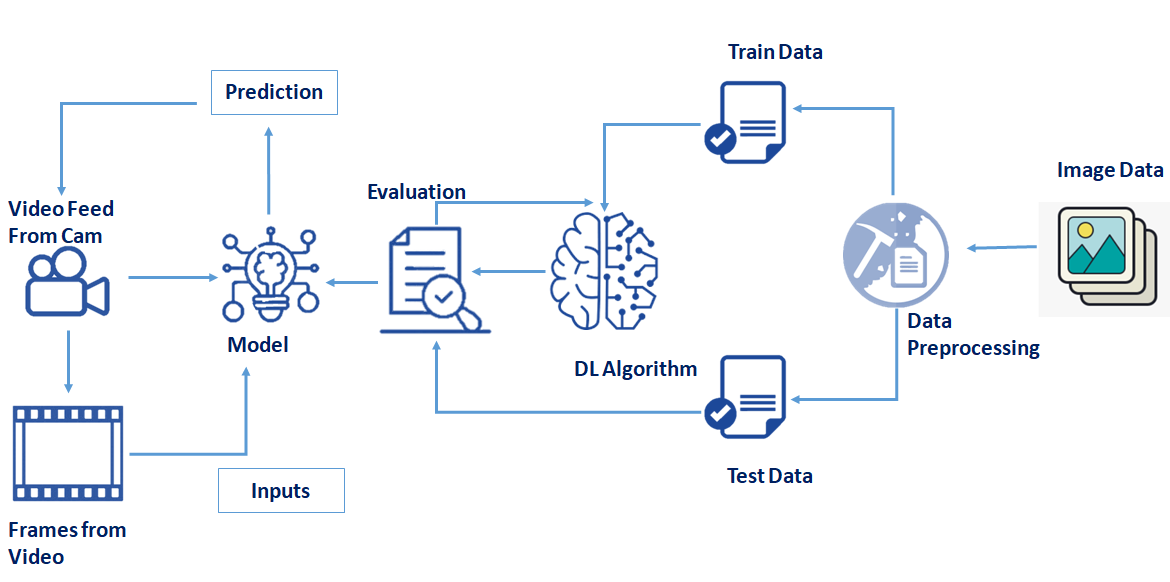
## Team Member: Ambati Venkatamallikarjunareddy, Sri Sivani college of

## Engineering

**Natural Disasters Intensity Analysis And Classification Using AI**

Natural disasters not only disturb the human ecological system but also destroy the properties and critical infrastructures of human societies and even lead to permanent change in the ecosystem. Disaster can be caused by naturally occurring events such as earthquakes, cyclones, floods, and wildfires. Many deep learning techniques have been applied by various researchers to detect and classify natural disasters to overcome losses in ecosystems, but detection of natural disasters still faces issues due to the complex and imbalanced structures of images. To tackle this problem, we developed a multilayered deep convolutional neural network model that classifies the natural disaster and tells the intensity of disaster  of natural The model uses an integrated webcam to capture the video frame and the video frame is compared with the Pre-trained model and the type of disaster is identified and showcased on the OpenCV window.

**Technical Architecture:**



**Methodology**

An external file that holds a picture, illustration, etc.
Object name is sensors-21-02648-g001.jpg

Proposed architecture of multilayered deep convolutional neural network.

**Project Objectives**

By the end of this project you will:

* know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks
* Gain a broad understanding of image data.
* Work with Sequential type of modeling
* Work with Keras capabilities
* Work with image processing techniques
* Work with Opencv

### Project Flow

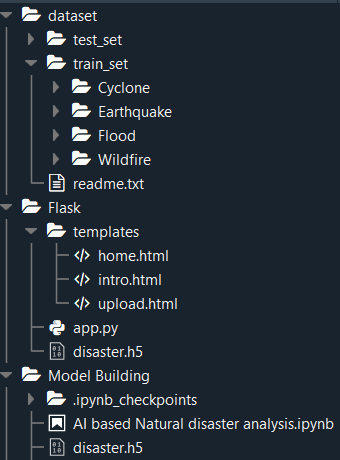
* The user interacts with the UI (User Interface) to open the integrated webcam.
* The video frames are captured and analyzed by the model which is integrated with flask application.
* Once model analyses the video frames, the prediction is showcased on the UI and OpenCV window

To accomplish this, we have to complete all the activities and tasks listed below

* Data Collection.
  + Collect the dataset or Create the dataset
* Data Preprocessing.
* Import the ImageDataGenerator library
* Configure ImageDataGenerator class
* ApplyImageDataGenerator functionality to Trainset and Testset
* Model Building
  + Import the model building Libraries

### Project Structure

Create a Project folder which contains files as shown below



* Dataset folder contains the training and testing images for training our model.
* We are building a Flask Application that needs  HTML pages stored in the templates folder and a python script app.py for serverside scripting
* we need the model which is saved and the saved model in this content is a disaster.h5
* templates folder contains home.html,intro.html,upload.html pages.

**Prerequisites**

**To complete this project, you must require the following software’s,  concepts, and packages**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform,  package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and Spyder

To install Anaconda navigator and to know how to use Jupyter Notebook & Spyder using Anaconda watch the video

Link: [Click here to](https://www.youtube.com/watch?v=5mDYijMfSzs&feature=emb_logo) watch the video

1. **To build Machine learning models you must require the following packages**

* **Numpy**:
* It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations
* **Scikit-learn:**
* It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors, and it also supports Python numerical and scientific libraries like NumPy and SciPy

* **OpenCV**
* [OpenCV](https://en.wikipedia.org/wiki/OpenCV) is a library of programming functions mainly aimed at real-time computer vision. Here, OpenCV is used to capture frames by accessing the webcam in real-time.
* Open anaconda prompt and type command

“pip install opencv-contrib-python”

* **Flask:**

Web framework used for building Web applications

* **Python packages:**
  + open anaconda prompt as administrator
  + Type “pip install numpy” and click enter.
  + Type “pip install pandas” and click enter.
  + Type “pip install scikit-learn” and click enter.
  + Type “pip install opencv-contrib-python” and click enter.
  + Type “pip install tensorflow==2.3.0” and click enter.
  + Type “pip install keras==2.4.0” and click enter.
  + Type “pip install Flask” and click enter.

**Prior Knowledge**

One should have knowledge on the following Concepts :

**Supervised and unsupervised learning:**

Watch the below video to know about the types of machine learnings

**Regression Classification and Clustering :**

**Artificial Neural Networks:**

**Convolution Neural Networks :**

**Flask :**

***Collection Of Dataset***

This milestone lets you create the dataset or download the dataset

In our research, the dataset used was collected from PyImage Search readers, who used Google Images to collect the total number (4428) of images in different classes. The dataset was separated into four classes: cyclone, earthquake, flood and wildfire, with 928, 1350, 1073 and 1077 images, respectively, as shown in [Figure 3](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8069408/figure/sensors-21-02648-f003/). The dataset was preprocessed to remove the noise by using an adaptive histogram equalizer. The whole dataset was divided into three groups: training, testing and validation. In total, 60% of the dataset was used for training, 23% for testing and 17% for validation. These percentages of the dataset were used to inform the machine on the percentage values of the dataset to be used for testing, training and validation purposes. The validation set was used to count the number of epochs for the whole training process.

An external file that holds a picture, illustration, etc.
Object name is sensors-21-02648-g003.jpg

**Image Preprocessing**

Image Pre-processing includes the following main tasks

* Import ImageDataGenerator Library.
* Configure ImageDataGenerator Class.
* Applying ImageDataGenerator functionality to the trainset and test set.

Note: The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data.

To know more about the data generator class  click on this [link](https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/)

**Import The ImageDataGenerator Library**

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

Let us import the ImageDataGenerator class from Keras

https://lh5.googleusercontent.com/LPMKGc0vye-Ui4HJSb8mNnqez4Y3N-cJQRdJJBE4B81tXVj2nGUU451ahgT6ffADDi8eh6LT1vFvTzWfZtpbHXqHNawlY4Evk7Bn-RW6qudYN_tk4JCYtKeUPiX-4F-49F_9e7Kp

**Configure ImageDataGenerator Class**

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

Image shifts via the width\_shift\_range and height\_shift\_range arguments.

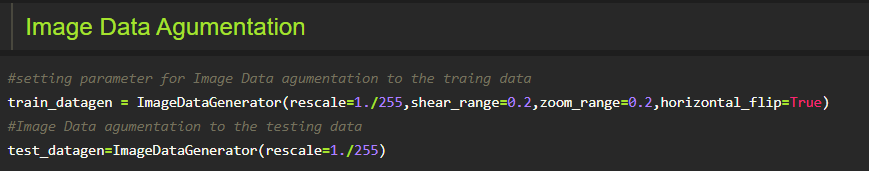
The image flips via the horizontal\_flip and vertical\_flip arguments.

Image rotations via the rotation\_range argument

Image brightness via the brightness\_range argument.

Image zoom via the zoom\_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.



Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code

For Training set using flow\_from\_directory function.

This function will return batches of images from the subdirectories Cyclone, Earthquake, Flood, Wildfire together with labels 0 to 3{Cyclone: 0, Earthquake: 1, Flood: 2, Wildfire: 3, }

Arguments:

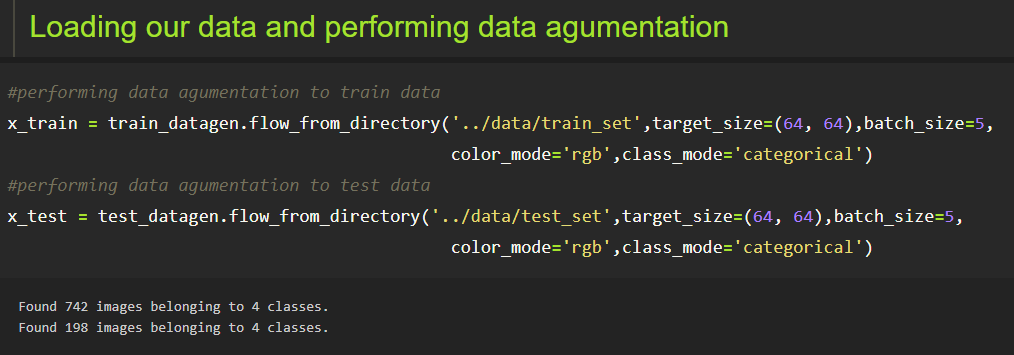
* directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
* batch\_size: Size of the batches of data. Default: 32.
* target\_size: Size to resize images after they are read from disk.
* class\_mode:

-  ‘int': means that the labels are encoded as integers (e.g. for sparse\_categorical\_crossentropy loss).

- 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical\_crossentropy loss).

- 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary\_crossentropy).

 - None (no labels).



We notice that 742 images are belonging to 4 classes for training and  198 images belonging to 4 classes for testing purposes.

### Model Building

### We are ready with the augmented and pre-processed image data, Lets begin our model building, this activity includes the following steps

### Import the model building Libraries

### Initializing the model

### Adding CNN Layers

### Adding Hidden Layer

### Adding Output Layer

### Configure the Learning Process

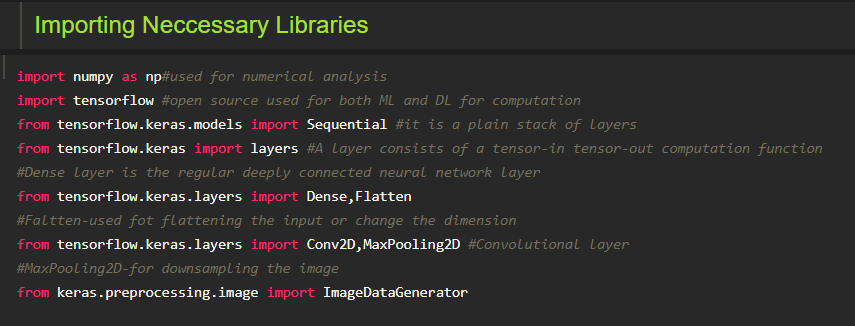
### Training and testing the model

### Saving the model

### To know more about model building please [click here](https://keras.io/about/)

### Train Test And Save Model

**The first step in building a model is to import the libraries**



**Activity 2: Initializing the model**

Keras has 2 ways to define a neural network:

* Sequential
* Function API

The Sequential class is used to define a linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add() method.

https://lh5.googleusercontent.com/KMIYCwuKIKNBNaf7vD7Tc-YLPcJGbhXFbYepr7_1mQg3vaSX3hpsjAVIyy_cf3vcQ950xmpJ_YGE4TZH2HGqzU-OrOm6wAmJMh4_YdZ1ENbYHGOoeeJ-BmIivIXp0E4oj98Iuo1X

**Activity 3: Adding CNN Layers**

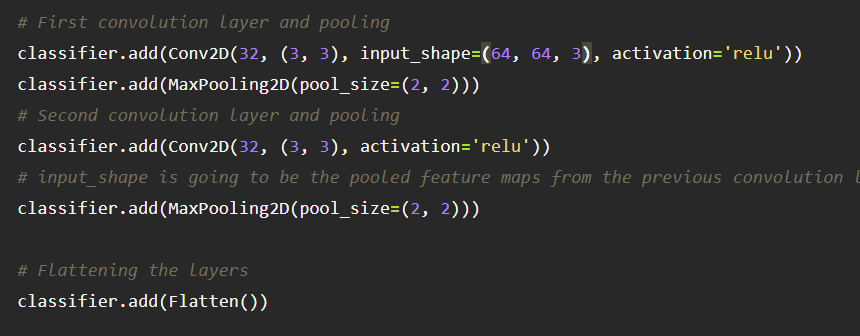
* For information regarding CNN Layers refer to the link

Link: <https://victorzhou.com/blog/intro-to-cnns-part-1/>

* As the input image contains three channels, we are specifying the input shape as (64,64,3).
* We are adding a convolution layer with activation function as “relu” and

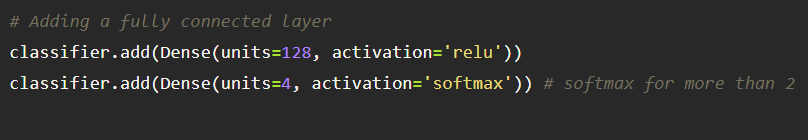
with a small filter size (3,3) and the number of filters (32) followed by a max-pooling layer.

* Max pool layer is used to downsample the input.( Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter)
* Flatten layer flattens the input. Does not affect the batch size.



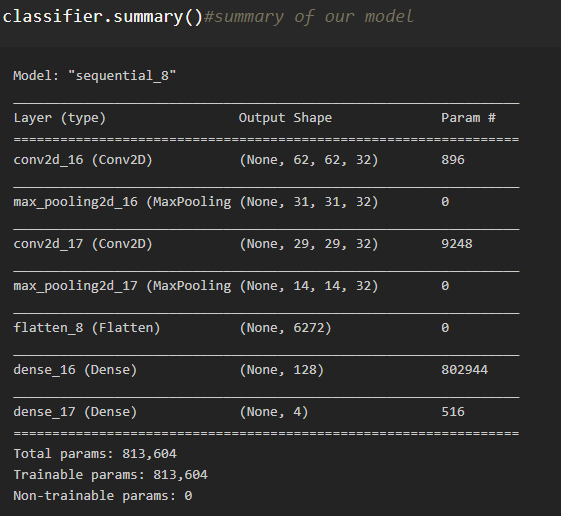
**Activity 5: Adding Dense Layers**

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.



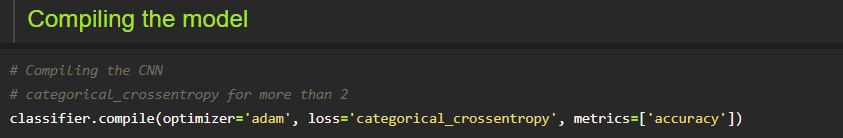
The number of neurons in the Dense layer is same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.



**Activity 6: Configure The Learning Process**

* The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires loss function during the model compilation process.
* Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
* Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process



**Activity 7: Train The model**

Now, let us train our model with our image dataset. The model is trained for 20 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 20 epochs and probably there is further scope to improve the model.

**fit\_generator** functions used to train a deep learning neural network

**Arguments:**

* steps\_per\_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of     steps\_per\_epoch as the total number of samples in your dataset divided by the batch size.
* Epochs: an integer and number of epochs we want to train our model for.
* validation\_data can be either:

                      - an inputs and targets list

                      - a generator

                      - an inputs, targets, and sample\_weights list which can be used to evaluate

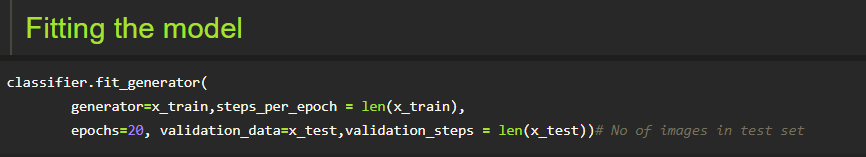
                        the loss and metrics for any model after any epoch has ended.

* validation\_steps: only if the validation\_data is a generator then only this argument

can be used. It specifies the total number of steps taken from the generator before it is

stopped at every epoch and its value is calculated as the total number of validation data points

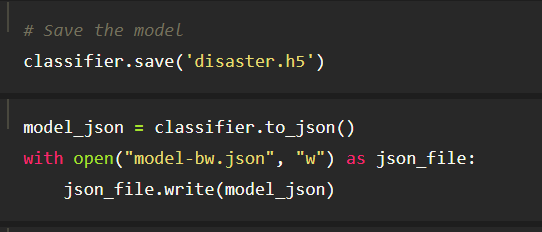
in your dataset divided by the validation batch size.



**Activity 8: Save the Model**

The model is saved with .h5 extension as follows

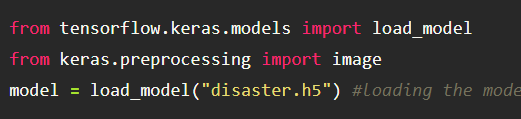
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.



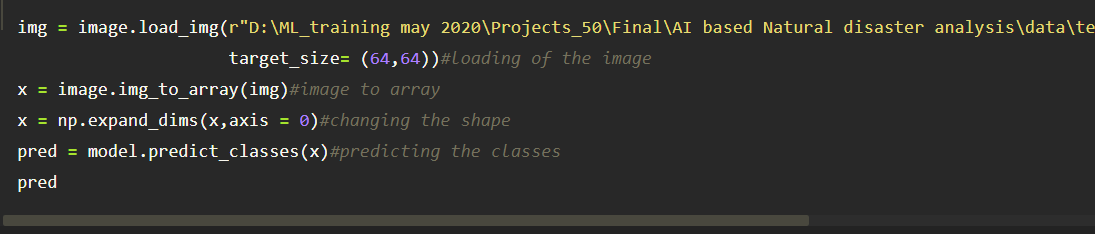
**Activity 9: Test The model**

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.

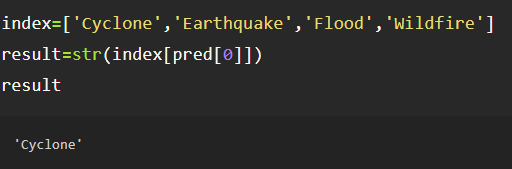
Load the saved model using load\_model



Taking an image as input and checking the results



By using the model we are predicting the output for the given input image



The predicted class index name will be printed here.

### Application Building

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building server-side script

**Build HTML Pages**

* We use HTML to create the front end part of the web page.
* Here, we  have created 3 HTML pages- home.html, intro.html, and upload.html
* home.html displays the home page.
* Intro.html displays an introduction about the project
* upload.html gives the emergency alert

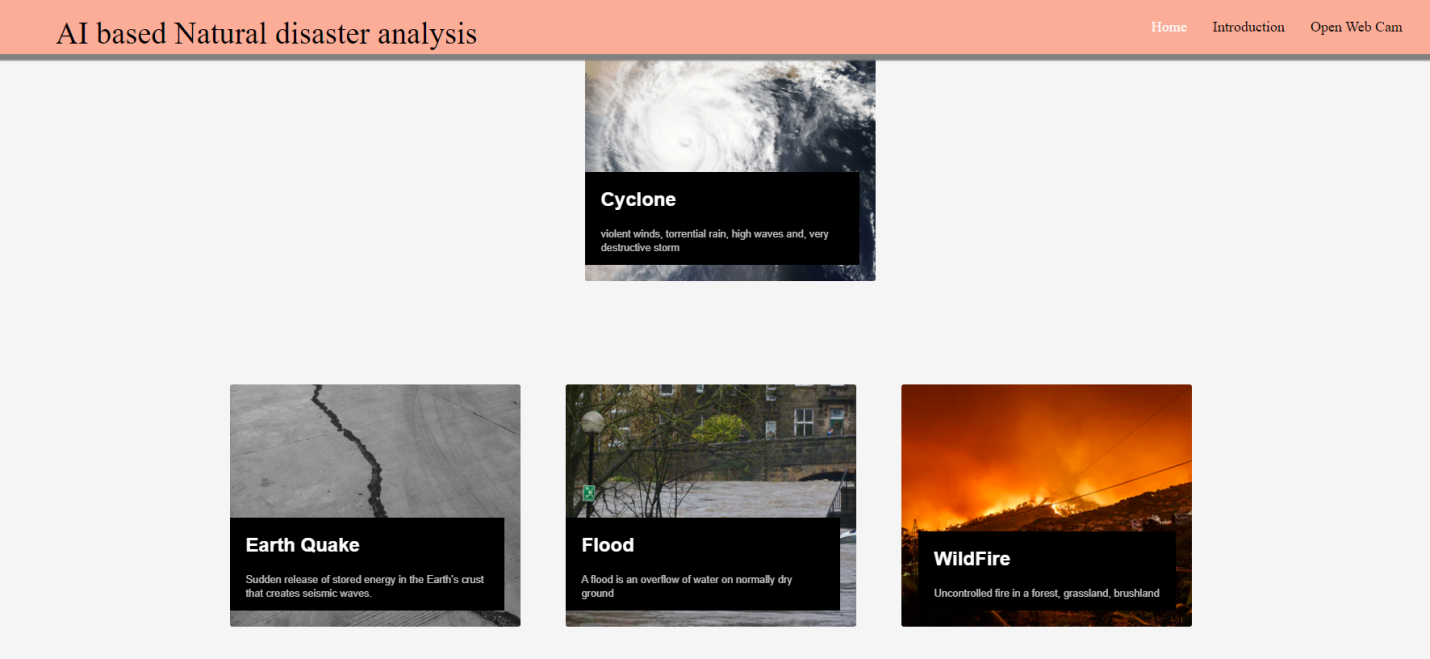
For more information regarding HTML

<https://www.w3schools.com/html/>

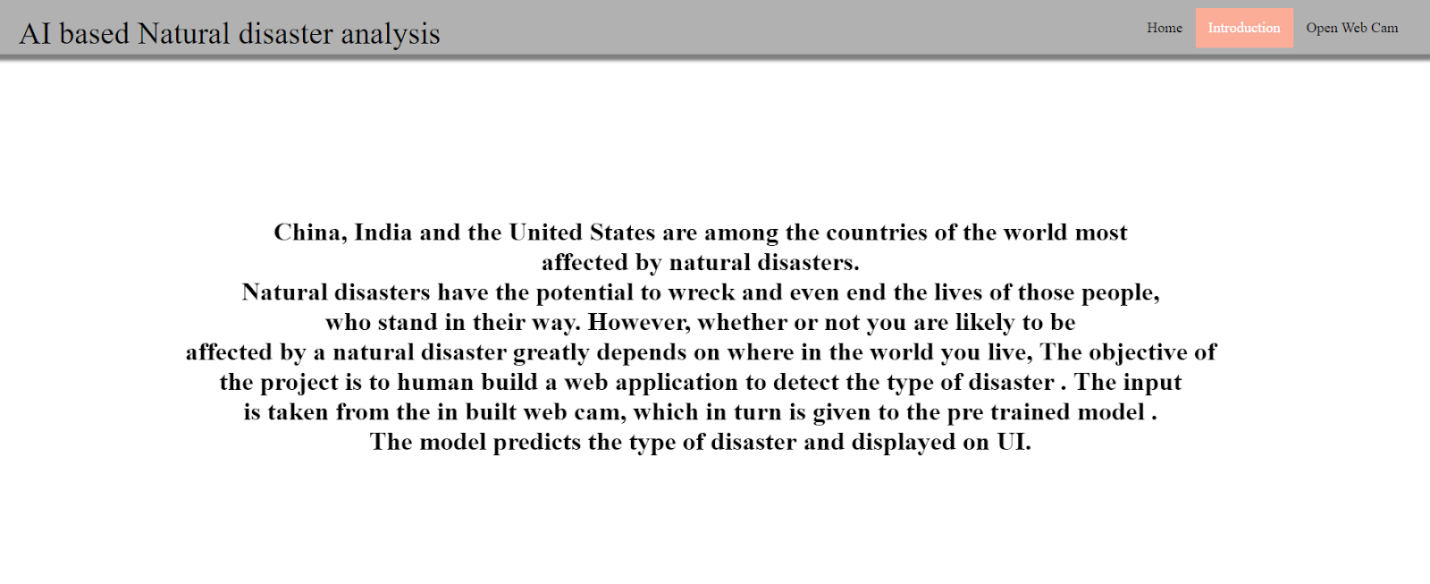
* We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.
* Link :[CSS](https://www.w3schools.com/css/) , [JS](https://www.w3schools.com/js/DEFAULT.asp)

Note : Refer to project structure section to download html files

Home.html looks like this



Intro.html

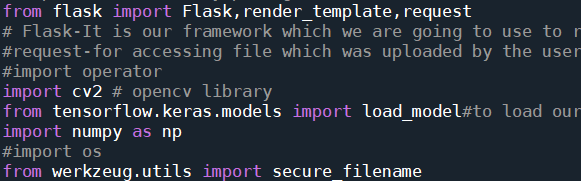
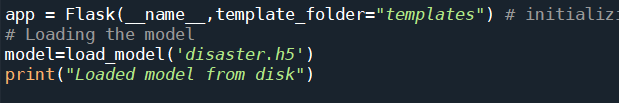
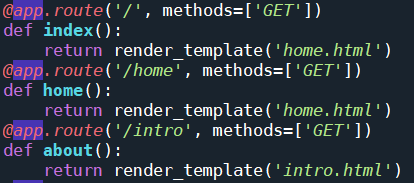


Upload.html



**Build Python Code**

* Let us build the flask file ‘app.py’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.
* The app starts running when the “\_\_name\_\_” constructor is called in main.
* render\_template is used to return HTML file.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.

Task 1: Importing Libraries  
The first step is usually importing the libraries that will be needed in the program.  
  
Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (\_\_name\_\_) as argument Pickle library to load the model file.  
  
Task 2: Creating our flask application and loading our model by using load\_model method  
  
  
  
Task 3: Routing to the html Page  
Here, the declared constructor is used to route to the HTML page created earlier.  
   
In the above example, the ‘/’ URL is bound with the home.html function. Hence, when the home page of the webserver is opened in the browser, the HTML page is rendered. Whenever you enter the values from the HTML page the values can be retrieved using the POST Method.  
Here, “home.html” is rendered when the home button is clicked on the UI  
  
  
  
When “Open Web Cam “ is clicked on the UI, predict function is executed  
  
https://lh3.googleusercontent.com/c2aAmzCfcYZC4F16q9sVGnjRwOqf7UsSg0DtdQ28szKv-ylE0zJPdIFzVpEK_EHEL4SuKztmdIGEmPFP2aRx9SyqCvktVVXvlvccyKtQIbX3IPZMvAVU_zudzgTS30UktFmb1RA1  
  
And the upload route is used for prediction and it contains all the codes which are used for predicting our results.

The tasks involved are

Grab the frames from the web cam.

Loop over the frames from the video stream

Convert the image from BGR to RGB

Predicting our results

Displaying the result

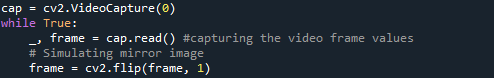
Run the application

Grab the frames from the webcam

To recognize the type of disaster we have to capture the video stream. There are two ways we can capture the input video

1.using in-built webcam

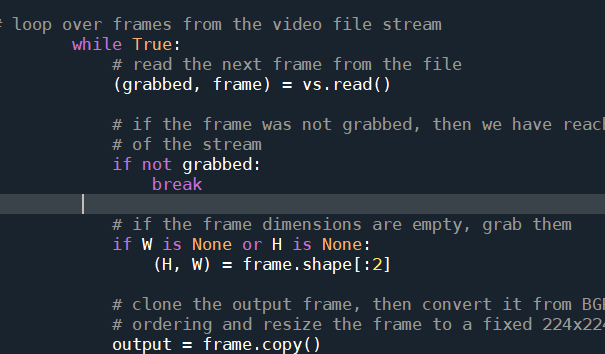
2. using video file residing on the disk



we use the VideoCapture module in the OpenCV library to capture a live video. We create a VideoCapture object using the constructor provided in the module. The argument to the constructor can be either a device index or the path of the video file. A device index is a number used to identify the webcam and, in most cases, the value is 0 The object of the VideoStream module enables us to capture frame-by-frame video data.

Loop over the frames from the video stream

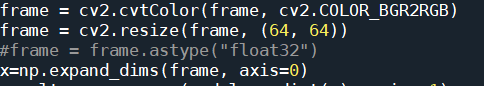
Let us grab the video frames from the video by looping over the frames and check if the frame was not grabbed, then we have reached the end of the stream. clone the output frame for showcasing the output



Convert the image from BGR to RGB

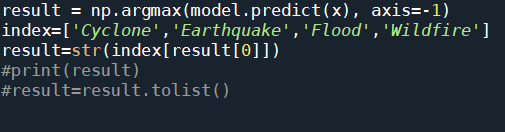
The frame we have captured is a 3-channel RGB colored image.

convert it from BGR to RGB , resize the frame to a fixed 64x64 and expand the dimensions to give it to the model for prediction.



Predicting the results

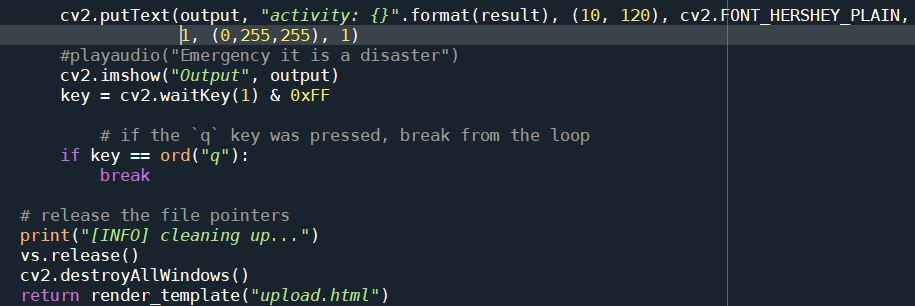
We then proceed to detect all type of disaster in the input image using model.predict function and the result is stored in result variable.



Displaying the result

After we recognize the type of disaster, we have to display the same on the live video stream for visualization.

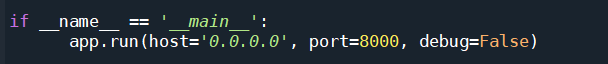
The cv2.imshow() function always takes two more functions to load and close the image. These two functions are cv2.waitKey() and cv2.destroyAllWindows(). Inside the cv2.waitKey() function, you can provide any value to close the image and continue with further lines of code.



Note: Press q on the keyboard to close the webcam which is opened after we grab the input and the application recognizes the input image.

Finally, Run the application

This is used to run the application in a local host. The local host runs on port number 8000.(We can give different port numbers)

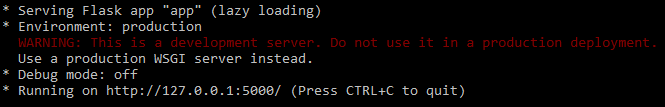


### Run The Application

* Open the anaconda prompt from the start menu.
* Navigate to the folder where your app.py resides.
* Now type “python app.py” command.
* It will show the local host where your app is running on http://127.0.0.1.8000/
* Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
* Enter the values, click on the predict button and see the result/prediction on the web page.

https://lh6.googleusercontent.com/pV9pRN1WAMfyMsGNhLx9zUq2PTqk2JPu9qOduLgP463Qn_MeemYmf4fDUEYXNYdadF7sP8n71mz33bWHPeSY7iqT-lJFvD4EULu-CISsCWY_WzSrUrgByRkPH8IT9XoR91Yp35H_

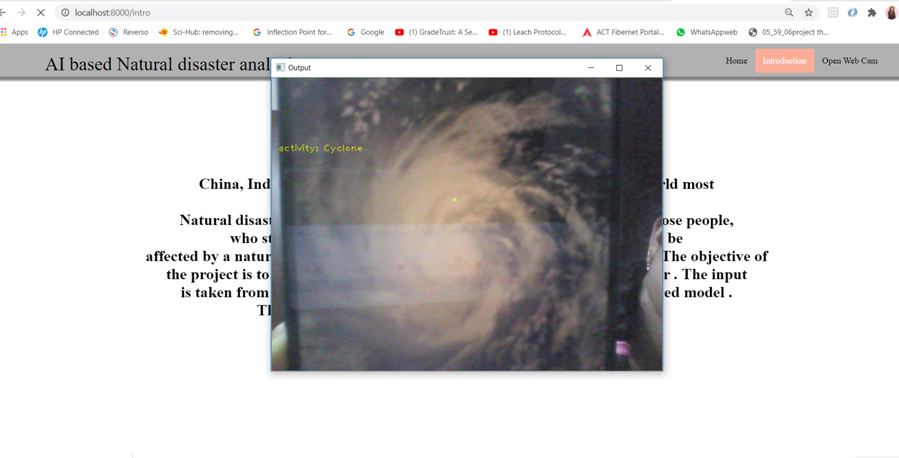
Then it will run on localhost:8000



Navigate to the localhost (<http://127.0.0.1:8000/>)where you can view your web page.

Click on open webcam and then you can see another spyder window which is opened to view the opened webcam.

Output screenshots:



## Conclusions

Many researchers have attempted to use different deep learning methods for detection of natural disasters. However, the detection of natural disasters by using deep learning techniques still faces various issues due to noise and serious class imbalance problems. To address these problems, we proposed a multilayered deep convolutional neural network for detection and intensity classification of natural disasters. The proposed method works in two blocks—one for detection of natural disaster occurrence and the second block is used to remove imbalanced class issues. The results were calculated as average statistical values: sensitivity, 97.54%; specificity, 98.22%; accuracy rate, 99.92%; precision, 97.79%; and F1-score, 97.97% for the proposed model. The proposed model achieved the highest accuracy as compared to other state-of-the-art methods due to its multilayered structure. The proposed model performs significantly better for natural disaster detection and classification, but in the future the model can be used for various natural disaster detection processes.



THANK YOU