Natural Disasters Intensity Analysis And Classification Using IBM Watson

1. INTRODUCTION

1.1. PROJECT DESCRIPTION :-

–

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 convolution 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. Deep learning method for the reconstruction of two-dimensional cardiac magnetic resonance images was proposed to enhance the image data acquisition process. Cascade deep convolutional neural networks use a 10 -fold method to reconstruct the feature map for the MR images. In this way, feature extraction sequence becomes very fast and it takes less than 5 to 10s to extract the feature matrix.

1.2. Technical Architecture :-

Fig-1.1- Technical Architecture

1. <u>AIM AND SCOPE OF THE PRESENT INVESTIGATION</u>

2.1.AIM OF THE PROJECT :-

Intensity Analysis and Classification of Natural Disaster by using Artificial Intelligence. Which detects Disaster occurred.

2.2 SCOPE OF THE PROJECT :-

1. Let start the loading and viewing of the dataset.
2. Implement out training script with keras.
3. Import necessary packages.
4. Train the network on our full dataset
5. Finding our Initial learning rate using CNN.

6. Generate the Model building applications and run the source code.

7. Then detect the disaster through the cam.

2. EXPERIMENTAL OR MATERIALS AND METHODS, NEURAL NETWORK USED

3.1 Project Objectives :-

By the end of this project you will:

1. know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks

2. Gain a broad understanding of image data.

3. Work with Sequential type of modeling

4. Work with Keras capabilities

5. Work with image processing techniques

6. Work with OpenCV

3.2. Project Flow :-

1. The user interacts with the UI (User Interface) to open the integrated webcam.
2. The video frames are captured and analyzed by the model which is integrated with flask application.
3. 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

1. Data Collection.
    a. Collect the dataset or Create the dataset
1. Data Preprocessing.
2. Import the ImageDataGenerator library
3. Configure ImageDataGenerator class
4. Apply ImageDataGenerator functionality to Trainset and Testset
5. Model Building
    a. Import the model building Libraries

Create a Project folder which contains files as shown below

Fig-3.1- Project Flow

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

3.3. <u>Pre-requirites</u> :-

3.3.1.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.

Fig:-3.2- Packages used in Project

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

1. **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

1. **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.

1. **OpenCV:**

- 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”

1. **Flask:**

   Web framework used for building Web applications

3.3.2.1.Python packages:

- 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.

3.3.2.2.Collection Of Dataset:

This milestone lets you create the dataset or download the dataset.

2. **Download The Dataset:**

Collect images of disaster-prone areas organized into subdirectories based on their respective names as shown in the project structure.

Create folders of types of disasters that need to be recognized.

In this project, we have collected images of 4 types of natural disasters Cyclone, Earthquake, Flood, and wildfire and they are saved in the respective subdirectories with their respective names.
Note: For better accuracy train on more images

Drive link:-

https://drive.google.com/file/d/1kLNL5VbyeRnp8kFx4i4_R4i3yDLdOe9A/view?usp=share_link

3.4. <u>DATASET</u> :-

There are two types of sets one is for Training and another for Testing. In training there has 4 type of disaster they are

1. Earthquake
2. Cyclone
3. Flood
4. Wild Fire

Let see the inside set images like

3.4.1.<u>Earthquake</u> :-

Fig:-3.3- Earthquake Dataset

3.4.2.<u>Cyclone</u> :-

Fig:-3.4- Cyclone Dataset

3.4.3.Flood :-

Fig:-3.5- Flood Dataset

_

3.4.4.<u>Wild Fire</u> :-

Fig:- 3.6- Wild-Fire Dataset

3.5. <u>Image Preprocessing</u> :-

Image Pre-processing includes the following main tasks

1. Import ImageDataGenerator Library.
2. Configure ImageDataGenerator Class.
3. 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.

3.5.1.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.

Fig:-3.7- Import ImageDataGenerator

3.5.2.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:

1. Image shifts via the width_shift_range and height_shift_range arguments.
2. The image flips via the horizontal_flip and vertical_flip arguments.
3. Image rotations via the rotation_range argument
4. Image brightness via the brightness_range argument.
5. Image zoom via the zoom_range argument.

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

Fig:-3.8- DataGenerator Class

3.5.3.Apply ImageDataGenerator Functionality To Trainset And Testset:

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, }

3.5.3.1.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).

Fig:-3.9- Dataset Adding

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

3.6 . <u>Model Building</u> :-

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

3.7. Train Test And Save Model :-

Activity 1: Import the Libraries:

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

Fig:- 3.10- 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.

Fig:- 3.11-

Initializing the model

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 down sample 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.

Fig:- 3.12- Adding CNN Layers

Activity 5: Adding Dense Layers:

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

Fig:- 3.13- Adding Dense Layers

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.

Fig:-3.14- Model Summary

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.

Fig:-3.15- Learning 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 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.

Fig:-3.16- Train the Model

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.

Fig:-3.17- Save the Model

Activity 9: Test The model:

1. 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

Fig:- 3.18-Test the Model

1. Taking an image as input and checking the results

Fig:- 3.19- Input for Checking

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

Fig:- 3.20- Result for Checking Input

4.<u>APPLICATION BUILDING</u>

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

1. Building HTML Pages

2. Building server-side script

4.1.Build HTML Pages :-

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

   For more information regarding HTML
   https://www.w3schools.com/html/

1. We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

2. Link :CSS , JS

4.1.1.Building Html Pages :-

Let’s create our HTML structure.

The **Hypertext Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes, and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as directly introduce content into the page. Other tags such as surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium, former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. A form of HTML, known as HTML5, is used to display video and audio, primarily using the element, in collaboration with JavaScript. Now create a form with the class "credit-form." Then divide our form in two sections. The first section is the form header where will have our form title, and the second is the form body where will have all the form elements and buttons.

Here three types of html pages are used they are :

1. Home.html
2. Intro.html
3. Upload.html

Let see

4.1.1.1.<u>Home.html</u> :-

Fig:-4.1- Home.html

Fig:-4.1- Home.html

4.1.1.2.Intro.html :-

Fig:-4.2- Intro.html

4.1.1.3.Upload.html :-

Fig:-4.3- Upload.html

4.1.1.4.<u>Home page</u>:-

1. The below image is the home page of the AI based Natural Disasters Intensity Analysis.

Fig:-4.4- Home Page

1. And there has introduction link, which gives the intro of AI based Natural Disasters Intensity Analysis.

4.1.1.5.__Intro Page__ :-

Fig:-4.5- Intro Page

1. It contains Cyclone, Earthquake, Flood and Wild Fire detections for every disaster there has their Wikipedia link , which gives the complete information about disaster.

4.1.1.6.__Cyclone__:-

Fig:-4.6- Cyclone Wikipedia

4.1.1.7.<u>Earthquake</u>:-

Fig:-4.7-Earthquake Wikipedia

4.1.1.8.<u>**Flood**</u>:-

Fig:-4.8-Flood Wikipedia

_

4.1.1.9.<u>**Wild Fire**</u> :-

Fig:-4.19- Wild-Fire Wikipedia

1. And there has another link that is upload, which can scan the disaster trough the cam.

4.1.1.10.<u>Upload Page</u>:-

Fig:-4.10- Upload Page

-

-

4.2.<u>Build Python Code</u> :-

- 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: <u>Importing Libraries</u>

The first step is usually importing the libraries that will be needed in the program.

Fig:-4.11-Import Operators

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**

Fig:-4.12-Loading the model

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 web server 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

Fig:-4.13-Routing the Html page

When “Open Web Cam “ is clicked on the UI, predict function is executed.

Fig:-4.14-Function Prediction

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.

Fig:-4.15-Loop over frame

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.

Fig:-4.16-Resize the frame

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.

Fig:-4.17-Predicting the Results

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.

Fig:-4.18-Displaying the Result

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.

We can use the web cam with an simple python program it has connected with the html files and it can scan the disaster easily and detect the disaster is which type through given dataset.

And in this simple program given frame size of the web cam is length is 64. And in this simple program given frame size of the web cam is bearth is 64.

Totally frame is square shape at l*b frame such as 64*64 sized frame contain the web cam.

Fig:-4.19-Frame size of webcam

This program for disaster upload, which disaster is upload and it is detected successfully or not. Let see the program

Fig:-4.20-Program for Disaster upload

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

Fig:-4.21-Run the application

a. .<u>RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS</u>

5.1.<u>Run The Application</u> :-

- 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.

Fig:-5.1-Kernal

Running on http://127.0.0.1:5000

Fig:-5.2.After Run the Application

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.

-

-

5.2.Output screenshots :-

5.2.1.Wild fire :-

Output during Wild fire.

Fig:-5.3- During Wild-Fire

5.2.2.<u>Cyclone</u> :-

Output during Cyclone.

Fig:-5.4- During Cyclone

5.2.3.<u>Earthquake </u>:-

Output during Earthquake.

Fig:-5.5- During Earthquake

5.2.4.<u>Flood’s</u> :-

Output during Flood’s

Fig:-5.6- During Floods

6.SUMMARY AND CONCLUSION

—

—

SUMMARY :-

In this tutorial, you learned how to use computer vision and the Keras deep learning library to automatically detect natural disasters from images.

To create our natural disaster detector we fine-tuned VGG16 (pre-trained on ImageNet) on a dataset of 940 images belonging to four classes:

Cyclone/hurricane

Earthquake

Flood

Wildfire

After our model was trained we evaluated it on the testing set, finding that it obtained 95% classification accuracy.

Using this model you can continue to perform research in natural disaster detection

6.2.CONCLUSION :-

1. Many researchers have attempted to use different AI 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.

2. The model is tested on 198 natural images and training with 742 natural images and performance is calculated and expressed as different statistical values: Execution time- 29s 194ms/step And Accuracy- 93% at 18th Epoch out of 20, which is competitive and comparable with state-of-the-art algorithms.

6.3.<u>REFERENCES</u> :-

1. Refer the link below to downloadanaconda navigator.

   Link : https://youtu.be/1ra4zH2G4o0

2. Please refer to the link given below to download the data set andto know about the dataset .

   Link:
   https://drive.google.com/file/d/1kLNL5VbyeRnp8kFx4i4_R4i3yDLdOe9A/view?usp=share_link

3. Detail information about keras.

   Link: Keras ImageDataGenerator and Data Augmentation - PyImageSearch

4. For more information regarding HTML
   Link : https://www.w3schools.com/html