Broken Tree Detection on road

A project report submitted in partial fulfillment of the requirements for the award of the degree of

Master of Computer Applications

in

Computer Applications

By, Asish Chowdhury (205120023)

DEPARTMENT OF COMPUTER APPLICATIONS

NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI 620015

DECEMBER 2022

BONAFIDE CERTIFICATE

This is to certify that the project **“Broken tree detection on road”** is a project work successfully done by **Asish Chowdhury (205120023)** in partial fulfilment of the requirements for the award of the degree of Master of Computer Applications from National Institute of Technology, Tiruchirappalli, during the academic year 2022-2023 (5th Semester – CA749 Mini Project Work).

Dr. P.J.A. Alphonse	Dr. P.J.A. Alphonse
Project Guide	Head of the Department

Project viva-voce held on …………………………….

Internal Examiner **External Examiner**

Abstract

This project mainly assists the city concern authorities in detecting whether a tree is broken and blocking the road. Due to the storm or other reasons, a tree may fall on the road. It can block the entire route and may cause severe inconvenience to the people. This situation can create a big issue for transportation, ambulance patients, broken electric wires, etc. So as soon as our application detects any fallen tree, it will notify the concerned authorities, and they will instantly take action.

Acknowledgements

I am grateful to Almighty God for giving me the strength, knowledge, and understanding to complete this project. His love has been more than sufficient to keep and sustain me.

My profound gratitude goes to my excellent project guide and head of the department, Dr. P.J.A. Alphonse, for his invaluable support, patience, time, and guidance in seeing me to the completion of this project work.

Also, my gratitude goes to my mentor, Dr. S.R. Balasundaram, who patiently saw me complete this project.

I also extend gratitude and appreciation to my lecturers in the Computer Applications department, who have taught me at one point or another. May God continue to bless and protect you all.

I also wish to acknowledge the incredible support of my parents. They are the inspiration for my academic pursuit. God bless you both.

TABLE OF CONTENTS

1. Introduction

We all know the importance of transportation and connectivity in today’s world, but due to some reasons like a storm, hurricanes, typhoons, cyclones, or heavy rain, many trees fall on the road and blocks connectivity. In this situation, we wait for governmental action to clear the route, which may take an enormous amount of time.

So, my project solves this issue. If anyone finds a situation where a tree is fallen and blocks a route, they can take a photo of it and post it to the concerned authority portal. Therefore our application will predict if the image is authentic and if it anticipates a fallen tree, then it will inform the concerned authorities, and they will instantly take action to clear the route.

2.1 Problem statement

Due to some causes like a gale, storms, typhoons, cyclones, or heavy rain, many trees fall on the road and block connectivity routes.

Suppose a tree is blocking a road, then we must call an emergency phone number to fix this very soon. But the disadvantage is that you should know the emergency helpline since emergency responders may need to use this road and alternate access or remove the tree—the same with electrical wires down. If the dispatcher gives you a hard time, you should mention this. But still, you have to wait until governmental aid is provided. Suppose someone has a medical emergency, and they have to cross the route as quickly as possible. So this a very crucial problem to solve.

2.2 Requirements

- **Hardware Platform**

RAM	: 4 GB
PROCESSOR	: INTEL-CORE I3
PROCESSOR	: 2.2 GHz
HARD DISK	: 1TB
MOUSE	: SCROLLING MOUSE
KEY BOARD	: MM KEY BOARD

- **Software Platform**

  - Operating System : Windows 7/8/10

- **Library**

Numpy

Numpy is a general-purpose array-processing package. It provides a high- performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

Pandas

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.

Open-cv

OpenCV is a pre-built, open-source CPU-only library (package) that is widely used for computer vision, machine learning, and image processing applications. It supports a good variety of programming languages including Python.

Scikit-learn (Sklearn)

It is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python.

Joblid
joblib. dump() and joblib. load() provide a replacement for pickle to work efficiently on arbitrary Python objects containing large data, in particular large numpy arrays.

Matplotlib
Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- **Plateform**

  TensorFlow
  TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

  Keras
  Keras is a high-level, deep learning API developed by Google for implementing neural networks. TensorFlow has adopted Keras as its official high-level API. Keras is embedded in TensorFlow and can perform deep learning fast as it provides inbuilt modules for all neural network computations. At the same time, calculations involving tensors, computation graphs, and sessions can be custom-made using the Tensorflow Core API, which gives you total flexibility and control over your application and lets you implement your ideas relatively quickly.

- **Browser**

  Google Chrome

  It is used to open jupyter notebook and to build the frontend of this project.

- **<u>Tools</u>**

  <u>Jupyter notebook</u>

  Jupyter notebooks basically provide an interactive computational environment for developing Python based Data Science applications. They are formerly known as ipython notebooks. The following are some of the features of Jupyter notebooks that make it one of the best components of Python ML ecosystem −

  Jupyter notebooks can illustrate the analysis process step by step by arranging the stuff like code, images, text, output etc. in a step by step manner.

  It helps a data scientist to document the thought process while developing the analysis process.

  One can also capture the result as the part of the notebook.

  With the help of jupyter notebooks, we can share our work with a peer also.

  <u>Spider</u>

  Spyder is an open-source cross-platform IDE. The Python Spyder IDE is written completely in Python. It is designed by scientists and is exclusively for scientists, data analysts, and engineers. It is also known as the Scientific Python Development IDE and has a huge set of remarkable features which are discussed below.

- **<u>Languages</u>**

  Python
- **<u>Product Version</u>**

  Python 3.11.0(64-bit)

3. Methodology

3.1 Dataset collection

First of all dataset is collected from Kaggle datasets. I have collected different classes from different datasets. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges. So it is a trustable source of datasets.

3.2 Pre-processing

Pre-processing refers to the transformations applied to our data before feeding it to the model. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. The aim of pre-processing is an improvement of the image data that suppresses unwilling distortions or enhances some image features important for further processing. So pre-processing is required for proceeding to any other processes. Used pre-processing are –

Removing duplicate images, water marked images, very low resolution images; resize all images into a fixed size so that our model perform well.

3.3 Image splitting

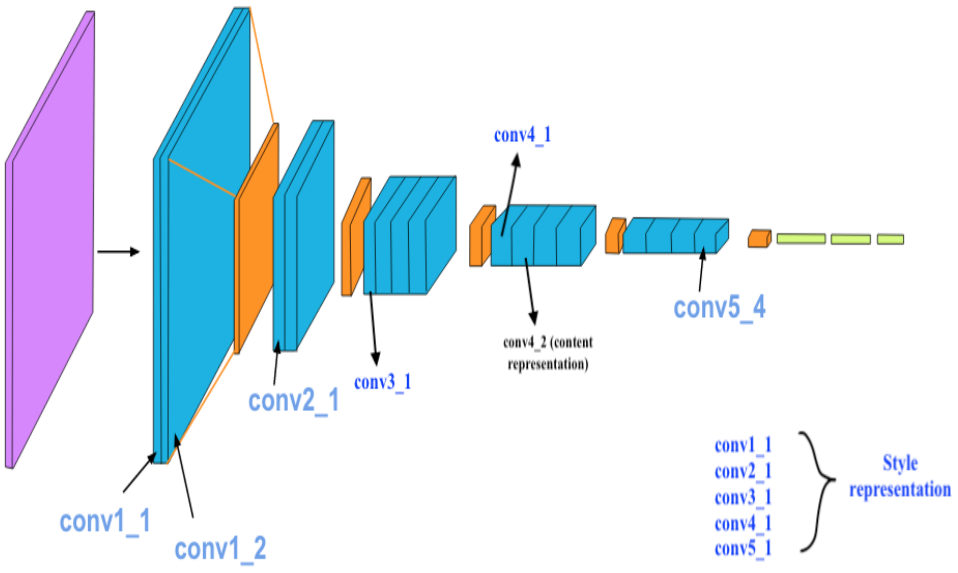
In this problem, we have 2 types of images. One is for normal road and another is for broken tree, so for training and validation purpose we have to split our image datasets into 2 sets. These sets are called classes. So both training and validation folders will contain set of images of these two classes

3.4 Model creation

Once, our data dataset is pre-processed and we have separated our classes then we can start to build our model.

I have followed the architecture of vgg-19 model but it is implemented from scratch. The model contains 5 blocks.

First and second block has two convolution layers followed by a polling layer. Similarly, third, fourth and fifth block has four convolution layers and followed by one polling layer and at last we have a flatten layer followed by three Dense layers. The overall architecture is given bellow-



3.5 ***Train & Test model***

Training in machine learning is the method of providing an ML algorithm with data to aid in identifying and learning good values for all attributes involved. There are several kinds of machine learning models, of which the most common ones are supervised and unsupervised learning. In this project, supervised learning is used and I have used around 1908 images for training and 24 images for testing purpose.

3.6 ***Prediction***

After building a well-trained model, next step is to examine the prediction and the accuracy of the model. If classification is about separating data into classes, prediction is about fitting a images or data that gets as close to the data as possible.

4. Deep-Learning concepts

4.1 Image Classification

Image classification is where a computer can analyse an image and identify the 'class' the image falls under. (Or a probability of the image being part of a 'class'.) A class is essentially a label, for instance, 'car', 'animal', 'building' and so on.

For example, you input an image of a sheep. Image classification is the process of the computer analysing the image and telling you it's a sheep. (Or the probability that it's a sheep.)

For us, classifying images is no big deal. But it's a perfect example of Moravec's paradox when it comes to machines. (That is, the things we find easy are difficult for AI.)

Early image classification relied on raw pixel data. This meant that computers would break down images into individual pixels. The problem is that two pictures of the same thing can look very different. They can have different backgrounds, angles, poses, etcetera. This made it quite the challenge for computers to correctly 'see' and categorise images.

4.2 *Deep-Learning*

Deep learning is a type of machine learning; a subset of artificial intelligence (AI) that allows machines to learn from data. Deep learning involves the use of computer systems known as neural networks.

In neural networks, the input filters through hidden layers of nodes. These nodes each process the input and communicate their results to the next layer of nodes. This repeats until it reaches an output layer, and the machine provides its answer.

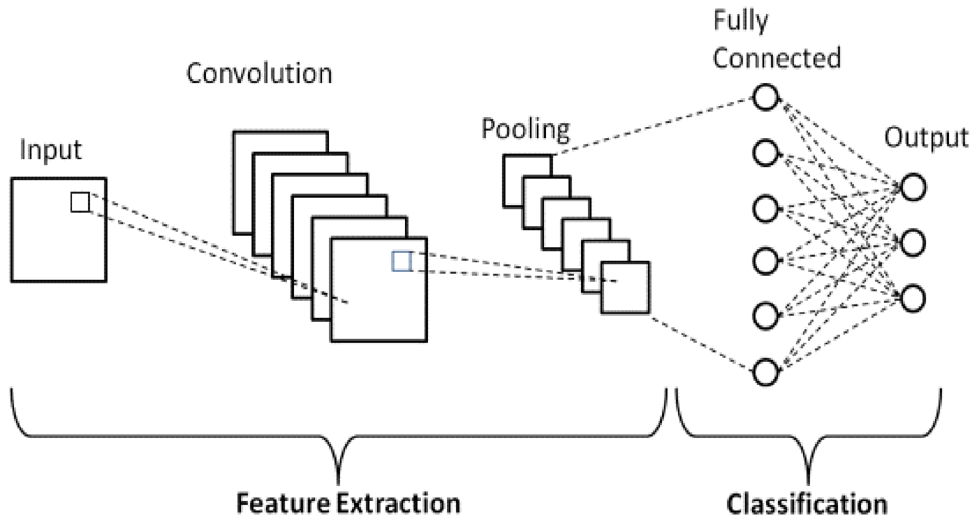
There are different types of neural networks based on how the hidden layers work. Image classification with deep learning most often involves convolutional neural networks, or CNNs. In CNNs, the nodes in the hidden layers don't always share their output with every node in the next layer (known as convolutional layers).

Deep learning allows machines to identify and extract features from images. This means they can learn the features to look for in images by analysing lots of pictures. So, programmers don't need to enter these filters by hand.

4.3 *Convolutional Neural Network (CNN)*

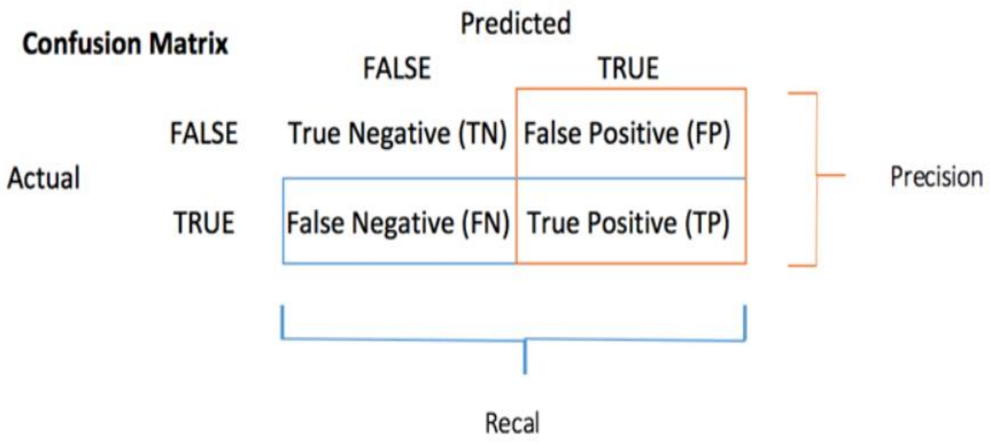
A CNN is a framework developed using machine learning concepts. CNNs are able to learn and train from data on their own without the need for human intervention. In fact, there is only some pre-processing needed when using CNNs. They develop and adapt their own image filters, which have to be carefully coded for most algorithms and models. CNN frameworks have a set of layers that perform particular functions to enable the CNN to perform these functions.

CNN layers can be of four main types: Convolution Layer, ReLu Layer, Pooling Layer, and Fully-Connected Layer. Convolution Layer: A convolution is the simple application of a filter to an input that results in activation. The convolution layer has a set of trainable filters that have a small receptive range but can be used to the full-dept of data provided. Convolution layers are the major building blocks used in convolutional neural networks. ReLu Layer: ReLu layers, also known as Rectified linear unit layers, are activation functions applied to lower overfitting and build the accuracy and effectiveness of the CNN. Models that have these layers are easier to train and produce more accurate results. Pooling Layer: This layer collects the result of all neurons in the layer preceding it and processes this data. The primary task of a pooling layer is to lower the number of factors being considered and give streamlined output. Fully-Connected Layer: This layer is the final output layer for CNN models that flattens the input received from layers before it and gives the result.



4.4 *Confusion matrix*

A confusion matrix describes the performance of the classification model. In other words, confusion matrix is a way to summarize classifier performance. The following figure shows a basic representation of a confusion matrix:



4.5 *Accuracy*

Accuracy is one measure for evaluating classification models. Informally, accuracy is the fraction of predictions our model got correct. Formally, accuracy has the following meaning:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

4.6 Precision

Precision is defined as the ratio of correctly classified positive samples (True Positive) to a total number of classified positive samples (either correctly or incorrectly).

1. Precision = True Positive/True Positive + False Positive
2. Precision = TP/TP+FP

- TP- True Positive
- FP- False Positive

4.7 Recall

The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples. The ***recall measures the model's ability to detect positive samples***. The higher the recall, the more positive samples detected.

1. Recall = True Positive/True Positive + False Negative
2. Recall = TP/TP+FN
    - TP- True Positive
    - FN- False Negative
    - Recall of a machine learning model will be low when the value of; TP+FN (denominator) > TP (Numerator)
    - Recall of machine learning model will be high when Value of; TP (Numerator) > TP+FN (denominator)

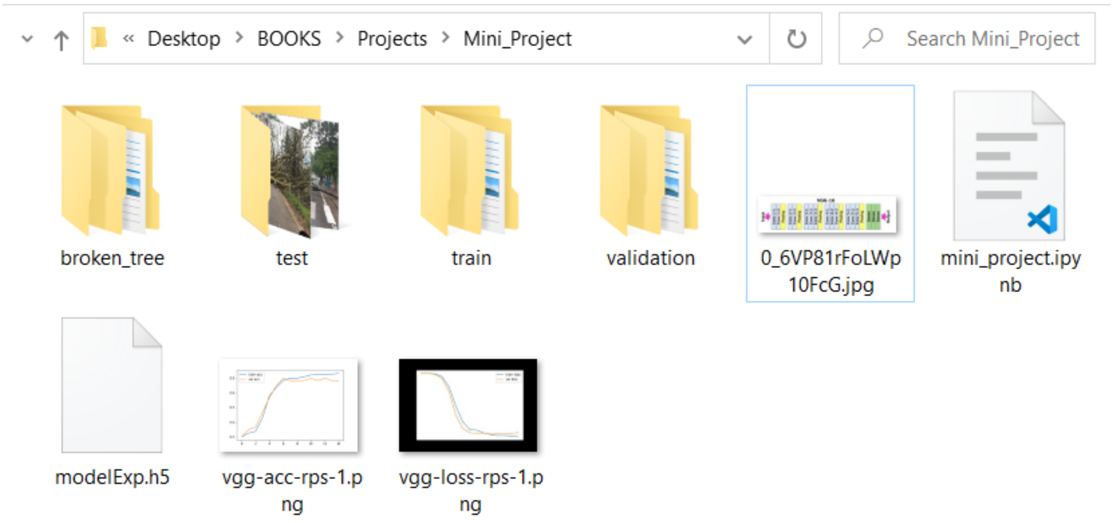
      Unlike Precision, Recall is independent of the number of negative sample classifications. Further, if the model classifies all positive samples as positive, then Recall will be 1.

4.8 Epoch

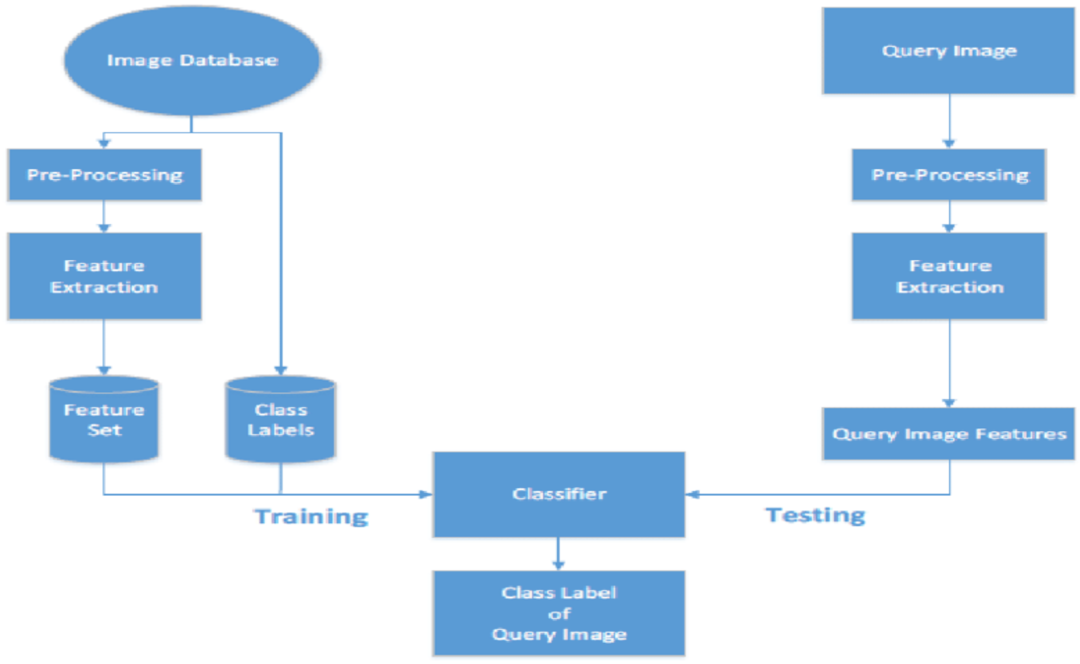
Epochs are defined as the total number of iterations for training the machine learning model with all the training data in one cycle. In Epoch, all training data is used exactly once. Further, in other words, Epoch can also be understood as the total number of passes an algorithm has completed around the training dataset. A forward and a backward pass together counted as one pass in training.

5. Design & Development

This section describes how I have developed this project and what the folder structure of my implementation is. The main folder contains six components named- **broken_tree**, **train**, **test**, **validation**, **mini_project**, and **modelExp**, where the broken_tree folder is used to make frontend, train and validation have subfolders called- 0_normal and 1_broken. The 0_normal folder contains images of all normal roads, and the 1_broken folder contains photos of fallen trees. The test folder contains images that are used for testing purposes. mini_project.ipy is our main python file.

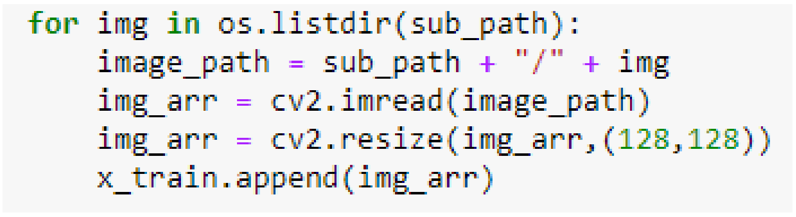


As discussed in Methodology, the steps that are followed to develop this application is given bellow,

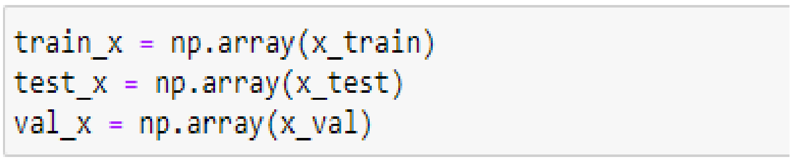


Pre-processing

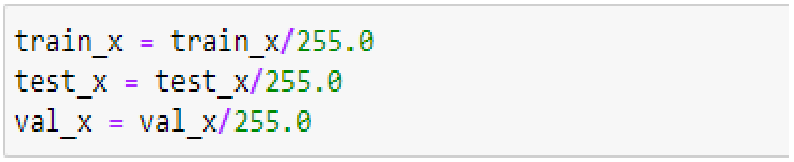
As it is already discussed in Methodology, for pre-processing, all duplicate and noisy images are removed from dataset. As input, our classification model takes all the images of the size of 128*128*3. So All the images are rescaled according to 128*128*3



```
for img in os.listdir(sub_path):
    image_path = sub_path + "/" + img
    img_arr = cv2.imread(image_path)
    img_arr = cv2.resize(img_arr,(128,128))
    x_train.append(img_arr)
```



```
train_x = np.array(x_train)
test_x = np.array(x_test)
val_x = np.array(x_val)
```



```
train_x = train_x/255.0
test_x = test_x/255.0
val_x = val_x/255.0
```

Data Augmentation

It is a process of artificially increasing the amount of data by generating new data points from existing data. The applied augmentation is shown below,

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
```

Model training

To train the model, 15 epochs are used with other parameters like steps_per_epoch, validation_steps and shuffle.

```
# fit the model
history = model.fit(
  training_set,
  validation_data=(val_x, val_y),
  epochs=15,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set),
  shuffle = True
)
```

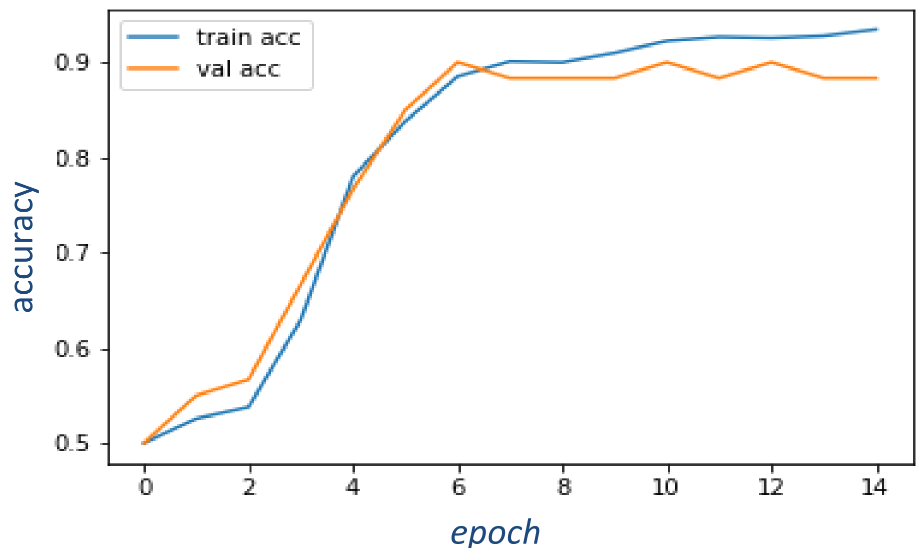
6. Performance Evaluation

In Machine Learning, the performance evaluation metrics are used to calculate the performance of the trained machine learning models. Looking up ML curves is a very crucial task because it gives us insight into the performance of our model, and if it does not goes well, and then we have to experiment with the different hyper parameters.

Interpreting a Validation Curve

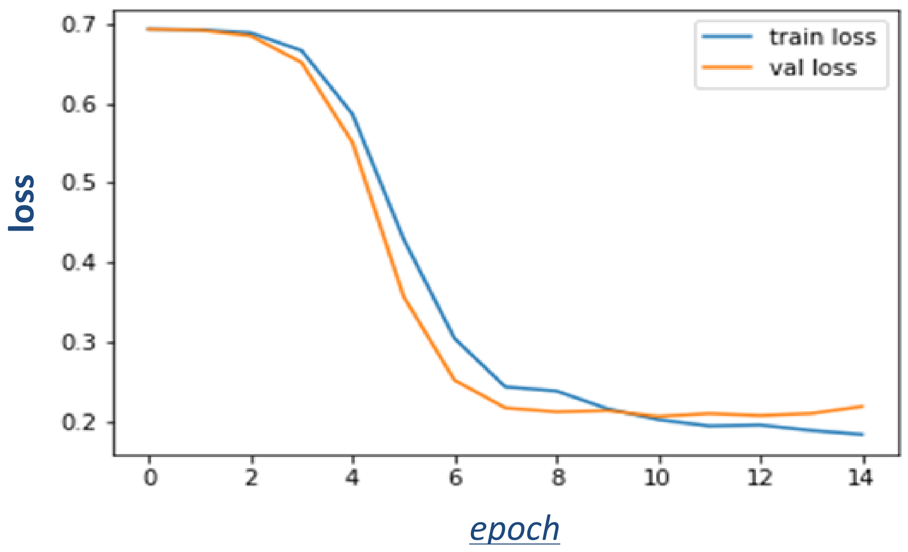
Analysing the results of a validation curve can sometimes take time and effort. Observe the following points while examining a validation curve:

- Ideally, we would want the validation and training curves to look as similar as possible.
- If both scores are low, the model will likely be underfitting. This means either the model is too simple or too few features inform it. It could also be the case that the model is regularized too much.
- If the training curve gets a relatively fast high score and the validation curve lags behind, the model is overfitting. This indicates the model is very tricky, and there is too little data, which could mean there is too small data.
- We would want the parameter value where the training and validation curves are closest to each other.

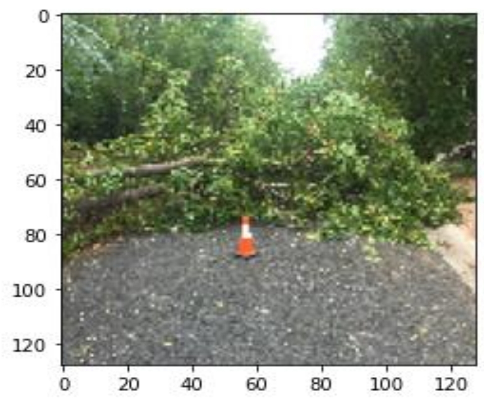


The above diagram shows that as our training accuracy is increasing our validation accuracy increases too. This is a good sign that our model is working well and does not contain underfitting or overfitting.

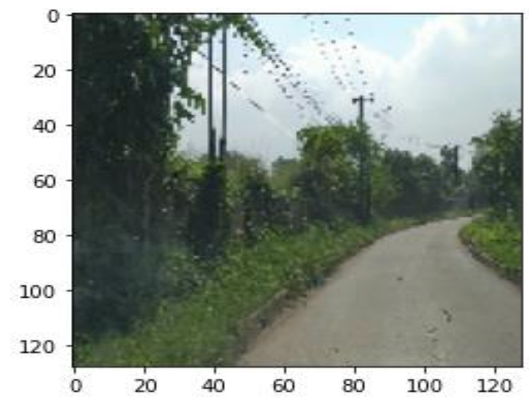
Similarly, we can analyse another curve to determine the model's performance and i.e. loss curve. The loss curve for our model bellow-



In final, our model gives around 93% accuracy on training dataset and around 88% accuracy on testing. Some of the predictions are shown below,



```
[[0. 1.]]
      Prediction : Broken Tree
```



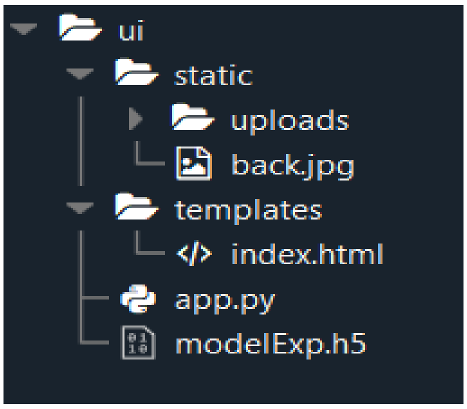
```
[[1. 0.]]
      Prediction : Normal Road
```

7. Front-end

Used Technologies

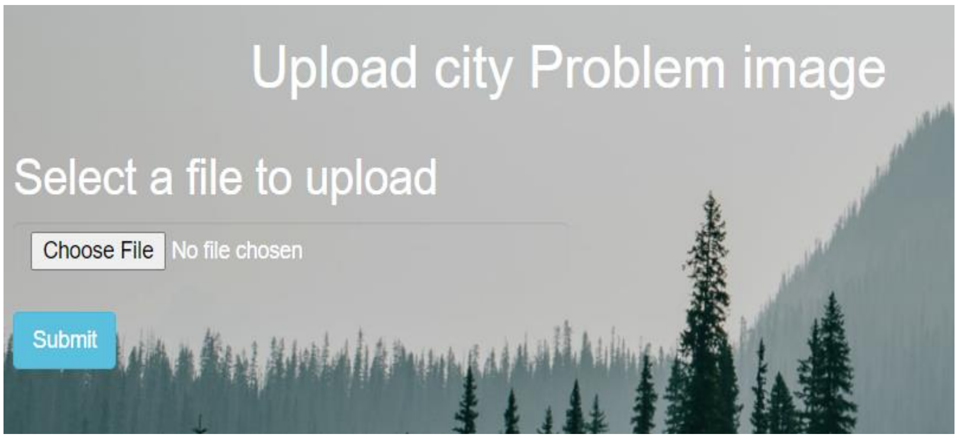
- **Flask** - Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

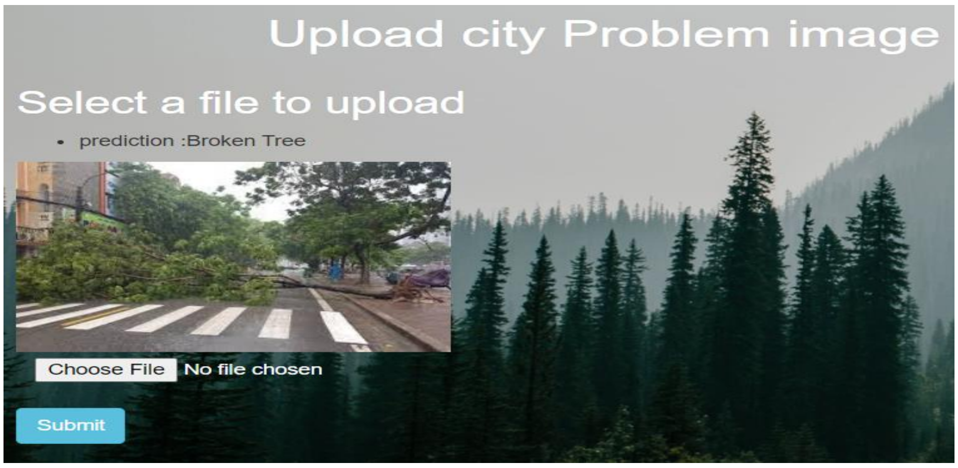
Directory structure



```
ui
  static
    uploads
    back.jpg
  templates
    index.html
  app.py
  modelExp.h5
```

User interface





8. Conclusion

This project mainly assists the city concern authorities in detecting whether a tree is broken and blocking the road. We all know the importance of transportation and connectivity in today's world, but due to some reasons like a storm, hurricanes, typhoons, cyclones, or heavy rain, many trees fall on the road and blocks connectivity. If anyone finds a situation where a tree is fallen and blocks a route, they can take a photo of it and post it to the concerned authority portal. Therefore our application will predict if the image is authentic and if it anticipates a fallen tree. This is done using Convolutional Neural Networks that extracts features from training dataset and learns to classify classes.

9. References

1. https://www.kaggle.com/datasets/tunhunhminh/demodata
2. https://www.youtube.com/watch?v=mjk4vDYOwq0&t=36s
3. https://www.youtube.com/watch?v=mRVTKrbRYi0
4. https://www.thinkautomation.com/eli5/eli5-what-is-image-classification-in-deep-learning/
5. https://www.javatpoint.com/precision-and-recall-in-machine-learning