ROAD POTHOLE PREDICTION USING CNN

A PROJECT REPORT

Submitted by

SRIRAM S	**(211418104267)**
SHANMUGASUNDARAM V	**(211418104245)**
SIDDARTHAN E	**(211418104249)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MAY 2022

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"ROAD POTHOLE PREDICTION USING CNN"** is the bonafide work of "**SRIRAM S (211418104267), SHANMUGASUNDARAM V (211418104245), SIDDARTHAN E (211418104249)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.S.MURUGAVALLI M.E.,Ph.D.,	**Dr.M.RAJENDIRAN, M.E.,Ph.D**
HEAD OF THE DEPARTMENT	**PROFESSOR**
DEPARTMENT OF CSE,	DEPARTMENT OF CSE,
PANIMALAR ENGINEERING COLLEGE,	PANIMALAR ENGINEERING COLLEGE,
NASARATHPETTAI,	NASARATHPETTAI,
POONAMALLEE,	POONAMALLEE,
CHENNAI-600 123.	CHENNAI-600 123.

Certified that the above mentioned students were examined in End Semester project viva-voice held on ________________________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **“SRIRAM S (211418104267), SHANMUGASUNDARAM V (211418104245) ,SIDDARTHAN E (211418104249)”** hereby declare that this project titled “**ROAD POTHOLE PREDICTION USING CNN ”**,under the guidance of “**DR.M.RAJENDIRAN” M.E,Ph.D** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI,M.A.,Ph.D.**, for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHI KUMAR,M.E.,Ph.D.**, and **Dr.SARANYA SREE SAKTHI KUMAR ,B.E.,MBA,Ph.D.**, for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.MANI ,M.E.,Ph.D.**, for his timely concern and encouragement provided to us throughout the course.

We thank the Head of the CSE Department, **Dr. S.MURUGAVALLI,M.E.,Ph.D.**, for the support extended throughout the project.

We would like to thank my **Project Guide Dr.M.RAJENDIRAN M.E,Ph.D**, and **Project Coordinator Dr.N.PUGHAZENDI M.E.,Ph.D.,** all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

Lastly we take the opportunity to thank all our staff members of the department of computer science and engineering. Regards to our family and friends who offered an unflinching moral support for the completion of this project.

SHANMUGASUNDARAM V

SIDDARTHAN E

SRIRAM S

ABSTRACT

Potholes are one of the extreme classes of road damage which can not only cause misalignment of the vehicles from their intended path but also damage vehicle structure and components. Both of these can lead to accidents. The damage caused is also dependent on the speed of the vehicle. Therefore it becomes necessary to avoid them which calls for their detection. This project is to train a Deep Learning algorithm capable of classifying images of different Road Pothole, such as a alphabet letter, and numeric A comparison of the proposed and current algorithms reveals that the accuracy hand gesture types classification based on CNNs is higher than other algorithms. It is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully fruits types on image. The problem statement can be given as follows. This system consists of two components one is mobile node and other is the access point. Access points responsible for storing the information about potholes in its vicinity, taking the feedback from vehicles, updating the information in repository and broadcasting the information to other vehicles.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

S.NO	ACRONYMS	EXPANSION
1.	CNN	Convolutional Neutral Network
2.	FC	Fully Connected Layer
3.	CNTK	Theano or Cognitive Toolkit
4.	GUI	graphical user interface
5.	CT	Computerized Tomography
6.	UML	Unified Modelling Language
7.	ReLU	Rectified Linear Unit

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

Potholes are one of the extreme classes of road damage which can not only cause misalignment of the vehicles from their intended path but also damage vehicle structure and components. Both of these can lead to accidents. The damage caused is also dependent on the speed of the vehicle. Therefore, it becomes necessary to avoid them which calls for their detection. With the onset of autonomous vehicles employment in passenger travel, the accurate detection is important for taking evasive measures to prevent potential accidents and reduced vehicle life by preventing subjection to unnecessary high stresses. It can also be conducive to road maintenance body in cities. Use of convolutional neural networks (CNNs) for image classification tasks is quite a popular practice due to their highly efficient results. This study focuses on using convolutional neural networks to come up with a robust model to detect potholes and suggest some unprecedented applications. Detecting potholes manually is a labor-intensive and time-consuming task. Many techniques have been implemented to detect potholes like vibration-based methods, 3D reconstruction based methods, and vision-based methods. But each of these techniques has some limitations. Thus, we have tried to address this problem of potholes using a novel technique of thermal imaging that uses the infrared rays emitted by objects to form images based on the temperature differences.

1.2 PROBLEM DEFINITION

Pothole detection system is a system that aims at warning the driver about the uneven roads and potholes in its path. We study the different ways in which goal of the system can be achieved. We justify the methods we have chosen in this projects. And then we give details about the working of the different subsystems. The problem statement can be given as follows. This system consists of two components one is mobile node and other is the access point. Access points responsible for storing the information about potholes in its vicinity, taking the feedback from vehicles, updating the information in repository and broadcasting the information to other vehicles. Whereas Mobile node which is the small device placed in vehicle is responsible for sensing those potholes which it did not have previous information about, locating and warning the driver about the potholes which it has information about, and giving the data about newly sensed pothole to access point. The whole scenario works as follows. While deploying the access point we feed in some initial data about potholes to it. Then it keeps on broadcasting the data. Vehicle equipped with the client device catches that data. Now the device has the information about the locations of potholes. The device is responsible for warning the driver about occurrences of pothole. But new potholes may always be formed because of environment or fatigue. So client device also acts as a sensor and finds out the occurrence of newly formed potholes on the road. If it finds out any new potholes it gives data of new pothole to Access point in terms of the feedback. Access points updates this information to its data store and then adds it to the information broadcast.

LITERATURE SURVEY

2. LITERATURE SURVEY

N. Hoang (Hoang, 2018) [1]

He proposed an artificial intelligence model for pothole detection and trained it using two machine learning algorithms including the least squares support vector machine (LS-SVM) and the artificial neural network (ANN). This work achieved the classification accuracy rate of approximately 89% using the LS-SVM algorithm and roughly 86% using ANN.

K. An *et al.* (2018) [2]

He experimented with several pre-trained mod- els for pothole detection. They achieved the classification accuracy rate of 97% using coloured images and 97.5% using grayscale images.

S. Ryu *et al.* (2015) [3]

He designed a pothole detection system to col- lect road images through an optical device mounted on a vehicle and then proposed an algorithm to detect potholes from the col- lected data. According to the proposed algorithm; first, a histogram and the closing operation of a morphology filter are used for extracting dark regions for the pothole. Next, candidate regions of a pothole are extracted using various features, such as size and compactness. Finally, a decision is made whether candidate regions are potholes or not, by comparing pothole and background features. Using this technique, they achieved the classification accuracy rate of 73.5%.

Akagic *et al.* (2017) [4]

He proposed an unsupervised vision-based method for pothole detection. According to the proposed method, first, the pothole areas are extracted from the

RGB colour space and then, image segmentation is performed on them. Afterwards, the search is done in the extracted region of interest (ROI) only. Their method is suitable as a pre-processing step for other super- vised methods. The effectiveness of their method depends on the extraction of ROI accuracy and they achieved an accuracy of 82%.

Vinayakumar Ravi Et al [5]

Most of the pretrained CNN models achieved above 99% accuracy and less than 0.005 loss with 15 epochs during the training. All 7 different types of EfficientNet (ENet)-based CNN models performed better in comparison to other models in terms of accuracy, average and macro precision, recall, F1 score. Moreover, the proposed ENet-based CNN models performed better than other existing methods such as VGG16 and ResNet-50 for pothole classification tasks.

Yu and Salari [6]

The 3D reconstruction approaches are categorized based on the technology used: laser-based or stereo-vision based techniques. The 3D laser scanner utilizes reflected laser pulses to create accurate digital models of objects [13–15]. These lasers could be used to detect potholes depth in real-time. Yu and Salari proposed a method [14] that involves the use of a light source to project a pattern of laser beams on the pavement, a camera to capture the pavement illuminated with the laser beams, and image processing on the captured images to identify potholes. Different approaches such as Multi-window Median filtering, Tile Partitioning with common thresholding [52], Laser line deformation, and Template matching were explored. The laser-based pothole detection techniques can detect potholes in real time.

Ouma et al [7]

The image processing object detectors are dependent on hand-crafted representations to extract low-level features. There were several previous image-processing research to detect potholes in a single image/frame [21–24], and other video-based methods were proposed to detect potholes and count their number over a series of frames [21,25–28]. The authors in [24] collected different frames and converted the frames into blurring grayscale images and then applied morphological and edge detection methods [55] to identify contours that are run through a Hough transform algorithm to extract features. Ouma et al. [56] applied fuzzy c-means clustering algorithm and morphological reconstruction techniques to 2D color images to detect potholes on asphalt pavement. In addition, Nienaber et al used image processing to identify the potholes on roads and reject unwanted objects such as vehicle and plants from the image [22].

Silvister et al [8]

There is an increasing tendency of applying machine learning (ML) methods to generate trained models to detect potholes in 2D digital images. Support vector machine (SVM) was used as a ML algorithm for road information analysis and pothole detection [29]. Texture measure based on histograms was used as the feature of the image and non-linear SVM was used to detect whether the image includes potholes. The authors in [30] created a SVM trained by a set of scale-invariant feature transform (SIFT) features for recognizing potholes in labeled images. These methods achieved accuracy of 91.4% for detecting potholes. Hoang [31] used least squares SVM and neural network with steerable filter-based feature extraction and achieved a pothole detection accuracy rate of roughly 89%. Recently, Hoang et al. [32] integrated the SVM and the forensic-based investigation (FBI) metaheuristic to optimize the detection accuracy, and their experiments achieved an accuracy of 94.833% for detecting potholes.

Amita Dhiman and Reinhard Klette [9]

Techniques for identifying potholes on road surfaces aim at developing strategies for real-time or offline identification of potholes, to support real-time control of a vehicle (for driver assistance or autonomous driving) or offline data collection for road maintenance. For these reasons, research around the world has comprehensively explored strategies for the identification of potholes on roads. This paper starts with a brief review of the field; it classifies developed strategies into several categories. We, then, present our contributions to this field by implementing strategies for automatic identification of potholes. We developed and studied two techniques based on stereo-vision analysis of road environments ahead of the vehicle; we also designed two models for deep-learning-based pothole detection. An experimental evaluation of those four designed methods is provided, and conclusions are drawn about particular benefits of these methods.

Panop Khumsap [10]

Road surface inspection for cracks, distortion, and disintegration together with appropriate surface treatments are mandatory in maintaining the ride quality and safety of the highways. Due to especially high occurrences of ‘reflection’, ‘alligator cracks’ and ‘potholes’ in Thailand, and the fact that they require markedly different treatment methods, a classifier that can distinguish among those two types of bad surface is most desirable.

SYSTEM ANALYSIS

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

We present a novel deep learning framework named the Iteratively Optimized Patch Label Inference Network (IOPLIN) for automatically detecting various pavement distresses that are not solely limited to specific ones, such as cracks and potholes. IOPLIN can be iteratively trained with only the image label via the Expectation-Maximization Inspired Patch Label Distillation (EMIPLD) strategy, and accomplish this task well by inferring the labels of patches from the pavement images. IOPLIN enjoys many desirable properties over the stateof-the-art single branch CNN models. It is able to handle images in different resolutions, and sufficiently utilize image information particularly for the high-resolution ones, since IOPLIN extracts the visual features from unrevised image patches instead of the resized entire image. Moreover, it can roughly localize the pavement distress without using any prior localization information in the training phase. In order to better evaluate the effectiveness of our method in practice, we construct a large-scale Bituminous Pavement Detection dataset named CQU-BPDD consisting of 60,059 high-resolution pavement images, which are acquired from different areas at different times. Extensive results on this dataset demonstrate the superiority of IOPLIN over the state- of the-art image classification approaches in automatic pavement distress detection.

Drawback

- It has not used on Deep Neural network in keras and TensorFlow as classifier.
- They are not using CNN and OpenCV computer vision technique
- It has not focused on increasing the recognition rate and classification of road pothole.

PROPOSED SYSTEM

We are proposing road pothole using Deep CNN(convolutional neural network) for deep learning technique. After collecting a suitable amount of data containing the images of potholes under various conditions and weather, and implementing CNN approach of deep learning has been adopted, that is a new approach in this problem domain using pothole imaging. Also, a comparison between the self-built convolutional neural model and some of the pre-rained models has been done.
the proposed method for this project is to train a Deep Learning algorithm capable of road pothole classification, This particular classification problem can be useful for road pothole detection. The using Deep Learning with the help of Convolution Neural Networks based on TensorFlow and Keras.
we proposed a deep learning (dl) based road pothole dataset to build classification method to prevent the pothole. the deep learning method used in the study is the Convolutional neural network (CNN). it is predicted that the success of the obtained results will increase if the CNN method is supported by adding extra feature extraction methods and classify successfully road pothole.

Advantages

- To classify road pothole image used on artificial neural network.
- It is best model for deep learning technique to easily road pothole.

3.3 FEASIBILITY STUDY

3.3.1 Introduction

The data use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. It has the test dataset (or subset) in order to test our models and it will do this using Tensor flow library in Python using the Keras method.

Deep learning needs data gathering have lot of past image data's. Training and testing this model working and predicting correctly.

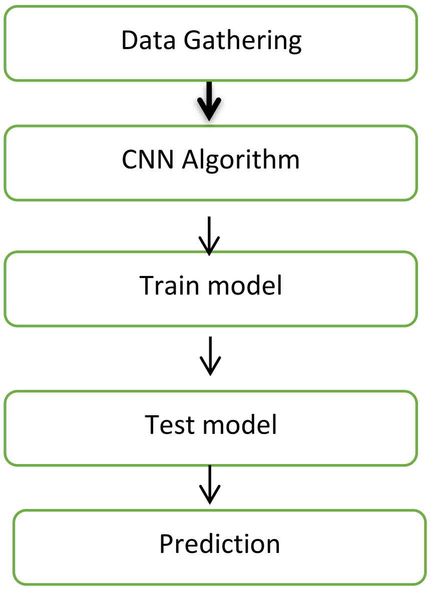


Fig:3.3.1 Steps of dataflow diagram

3.3.2 Economical Feasibility

Being a desktop application it will have no cost .In this, we don't need any server because it's a desktop application which can be installed easily in a couple of hours and does not require an internet connection. It reduces paper and printing cost and also the usage of paper is declined. Since, it is one time investment it will become profitable. It will apply in the local scan centers and Hospitals. From these it's clear that this project is financially feasible.

3.3.3 Source Feasibility

This project proposes an desktop application based on Anaconda Navigator. It is a desktop graphical user interface (GUI) which helps users to easily identify the presence of Potholes and also Drivers for faster identification of Potholes in roads. The Graphical user interface of our project makes the application easy to handle for any kind of age groups.

3.4 SOFTWARE ENVIRONMENT

Operating System	: Windows / Linux / Ubuntu
Simulation Tool	: Anaconda with Jupyter Notebook

3.5 HARDWARE ENVIRONMENT

Processor	: Intel core i5 / 3.2 GHz Processor
Hard disk	: 512 GB SSD HDD
RAM	: 32GB DDRA RAM

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1 Data Flow Diagram

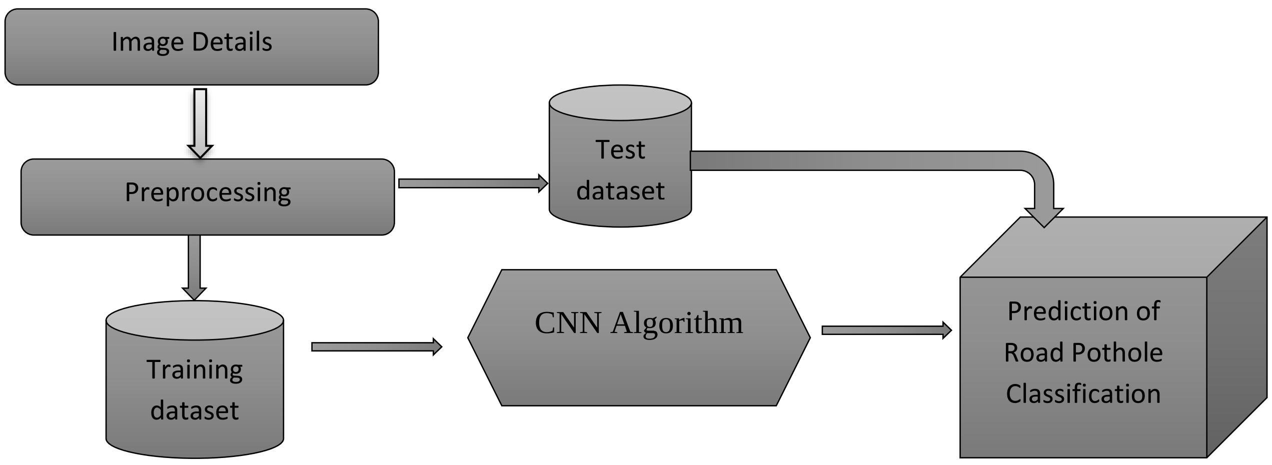
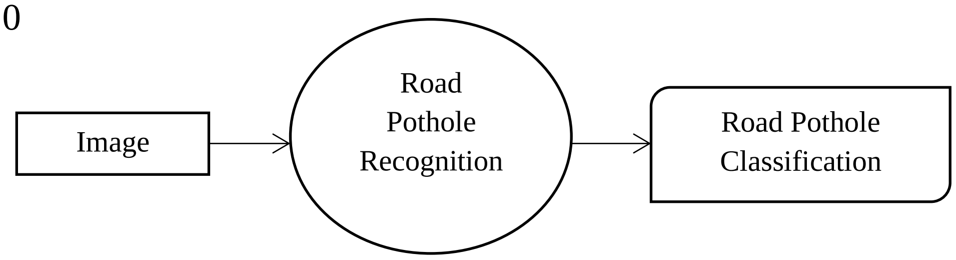


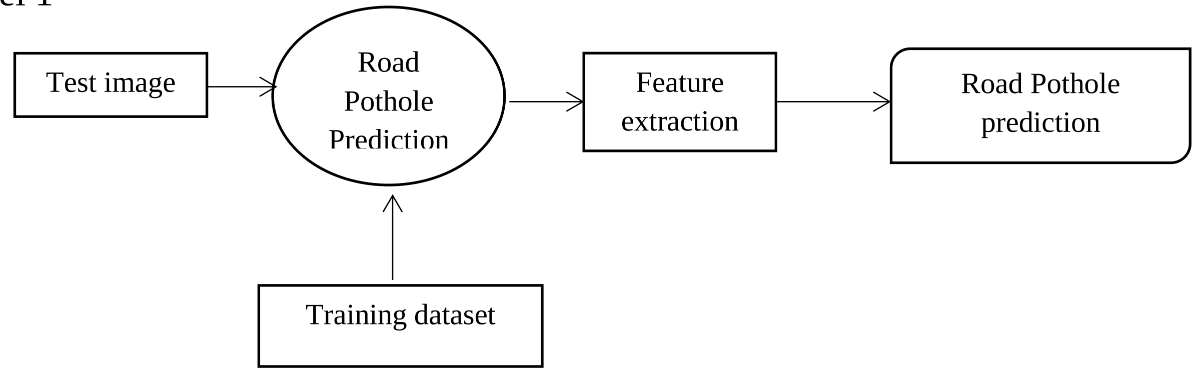
Fig:4.1 Process of dataflow diagram

A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

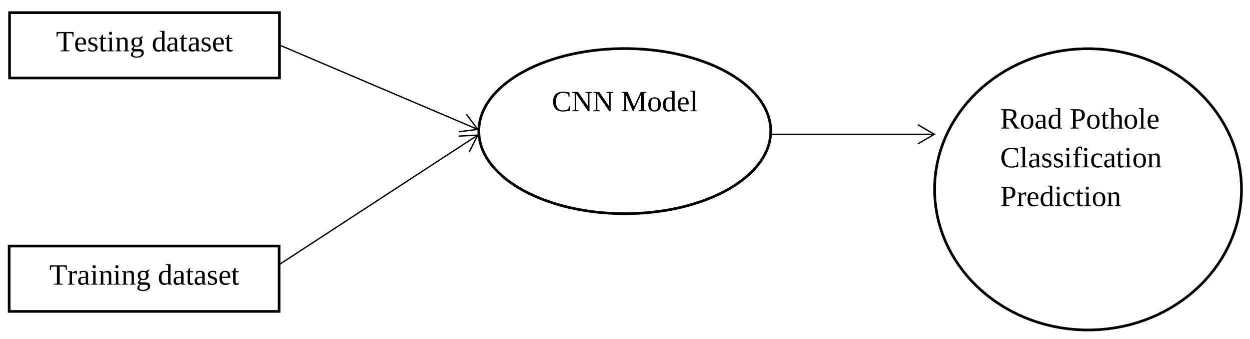
Level 0



Level 1



Level 2



Level 3:

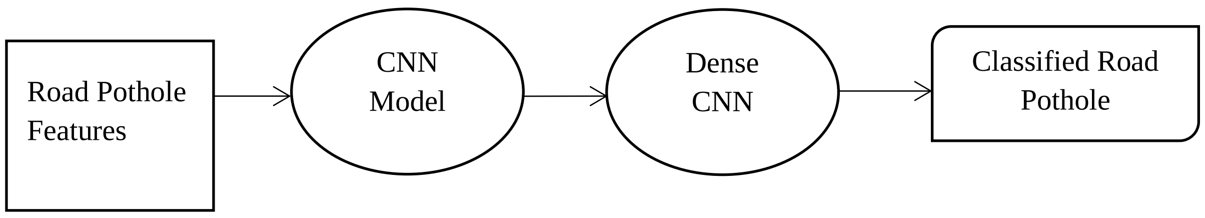


Fig:4.1 Data flow diagram

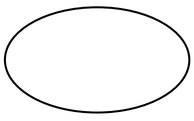
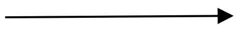
Symbol Name	**Symbol**	**Description**
Entity		An entity is represented by a rectangle which contains the entity's name.
Attribute		In the Chen notation, each attribute is represented by an oval containing attribute's name
One or More		It represents One or More

Table 4.1 Dataflow Diagram Symbol Description

4.2 ER Diagram

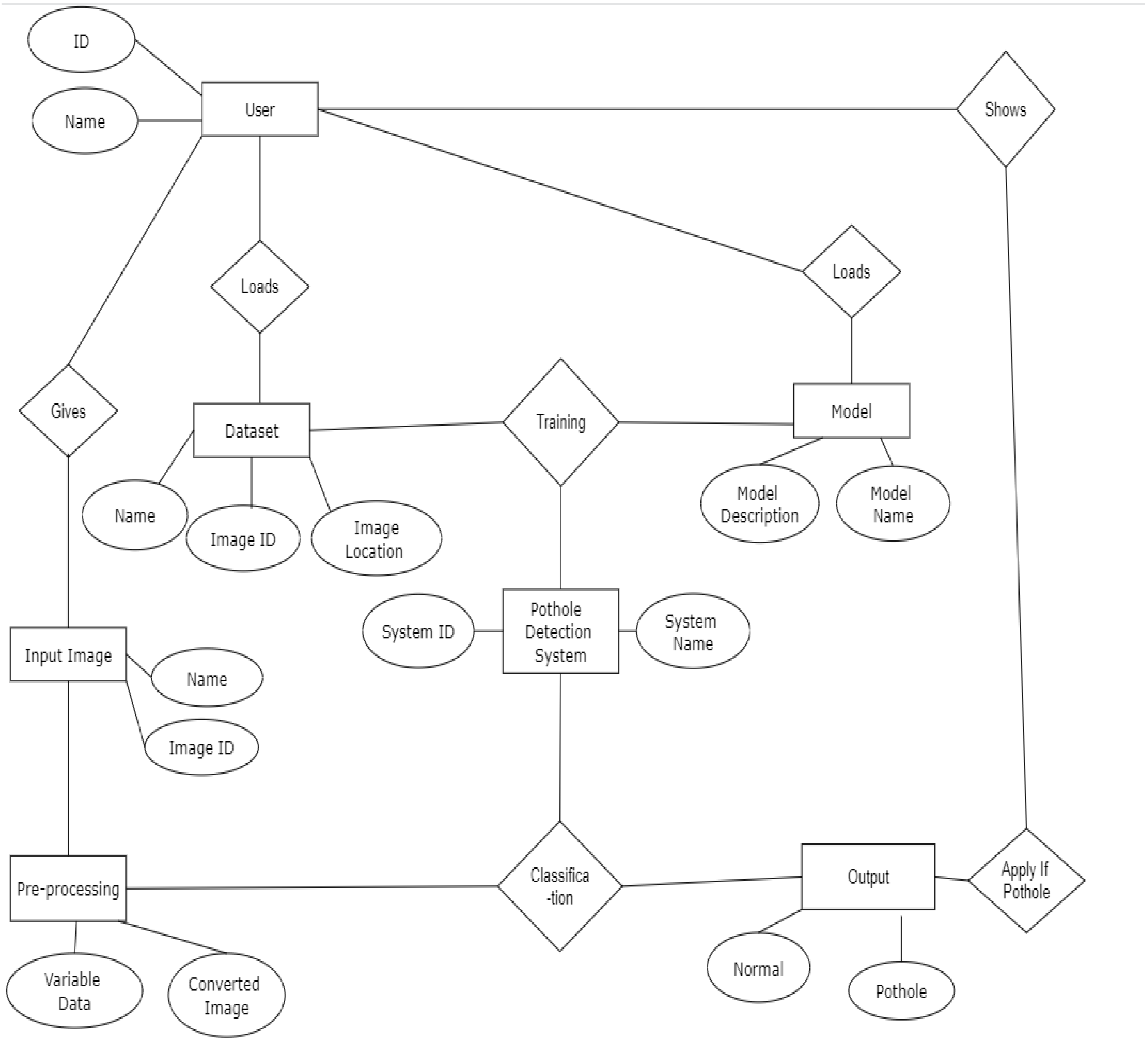


Fig:4.2 ER Diagram

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database.

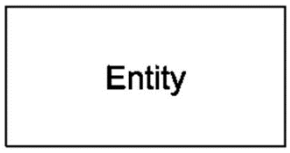
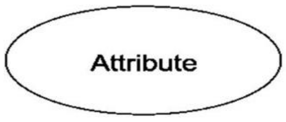
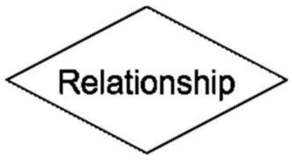
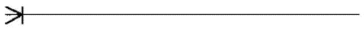
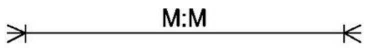
Symbol Name	Symbol	Description
Entity		An entity is represented by a rectanglewhich contains the entity’s name.
Attribute		In the Chen notation, each attribute is represented by an oval containing attribute’s name
Strong Relationship		A relationship where entity is existence- independent of other entities, and PK of Child doesn’t contain PK component of Parent Entity. A strong relationship is represented by a single rhombus
One or More		It represents One or More
Many - to - Many		It represents a one through many on bothsides of a relationship

Table 4.2 E-R Diagram Symbol Description

4.3 UML DIAGRAM

4.3.1 Use case diagram

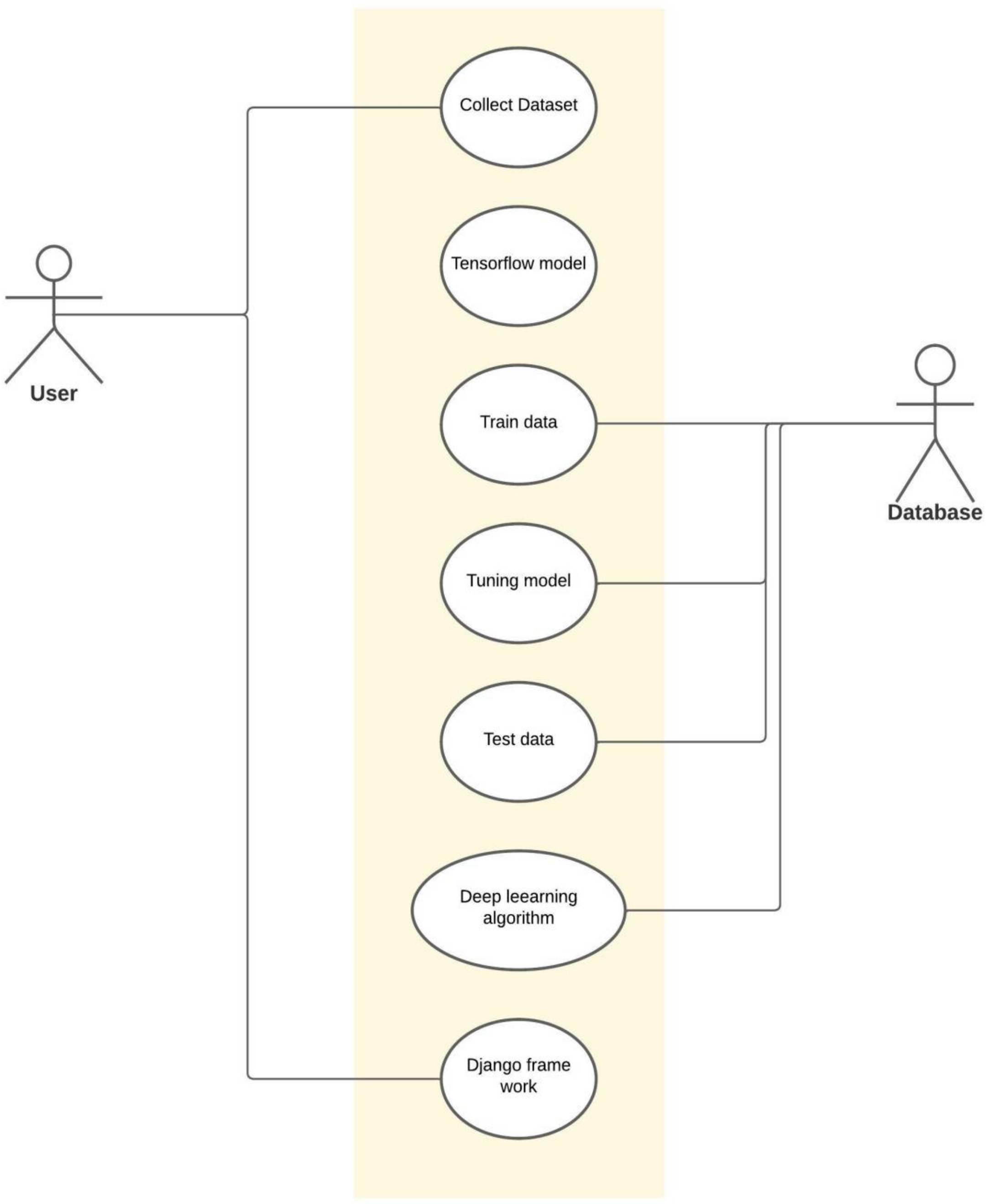


Fig:4.3.1 Use case diagram

Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

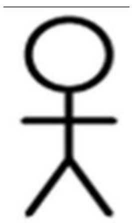
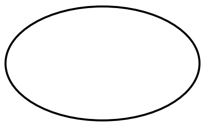
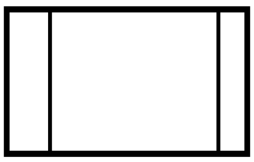
Symbol Name	Symbol	Description
Actor		Actors are the users of a system.
Usecase		Label the ovals with verbs thatrepresent the system's functions.
Data Source		A Non-Human Actor is represented by this symbol.

Table 4.3.1 Usecase Diagram Symbol Description

4.3.2 Class diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system.

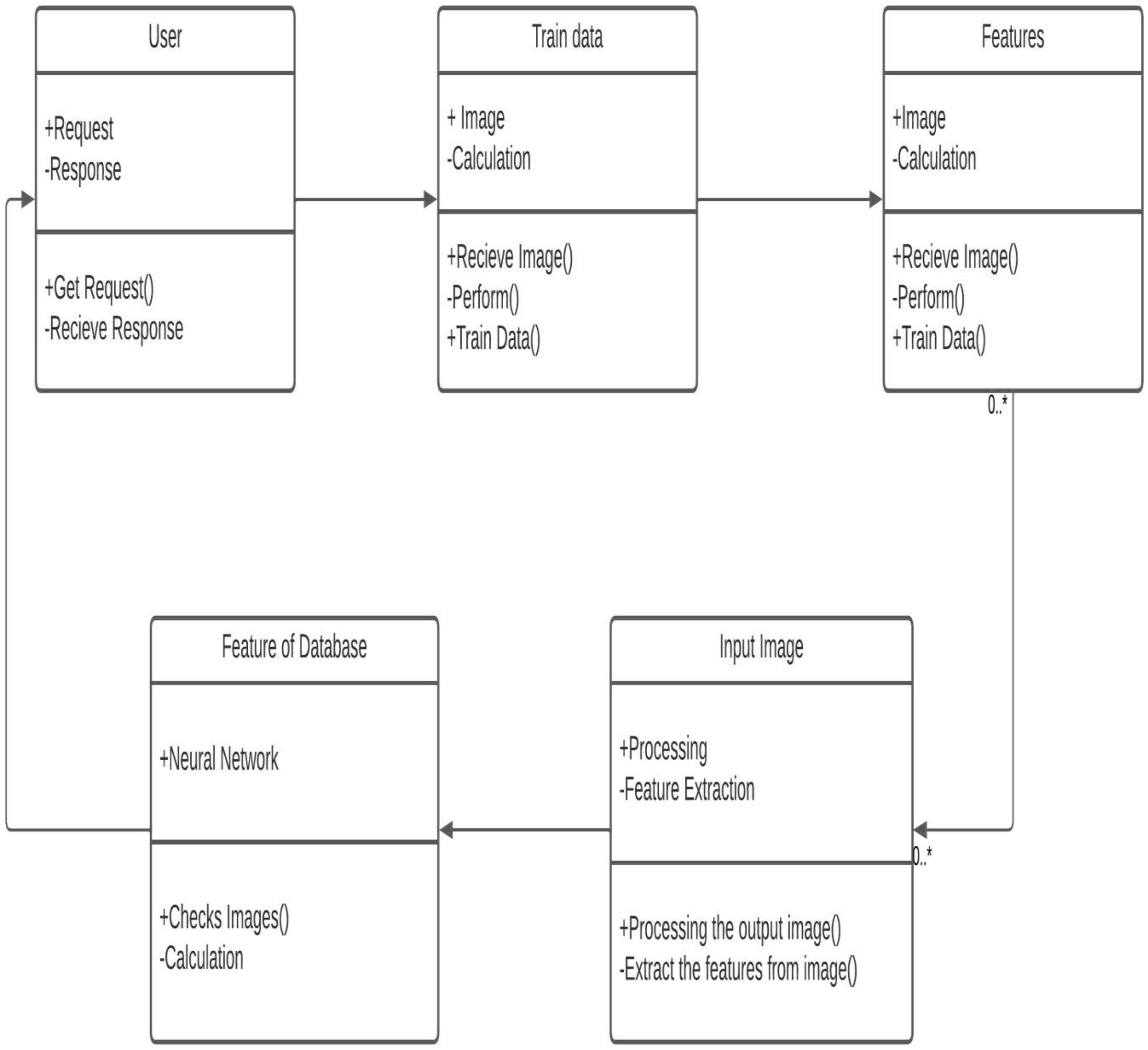


Fig:4.3.2 Class diagram

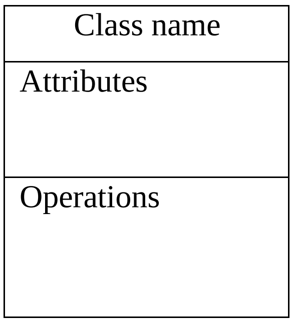
Symbol Name	Symbol	Description
Association	———————	Associations are relationships between classes in a UML Class Diagram. They are represented by a solid line between classes.
Dependency	---------------->	An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modeled as a dependency relationship.
Class		A class represent a concept which encapsulates state (**attributes**) and behavior (**operations**). Each attribute has a type.

Table 4.3.2 Class Diagram Symbol Description

4.3.3 Activity diagram

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart.

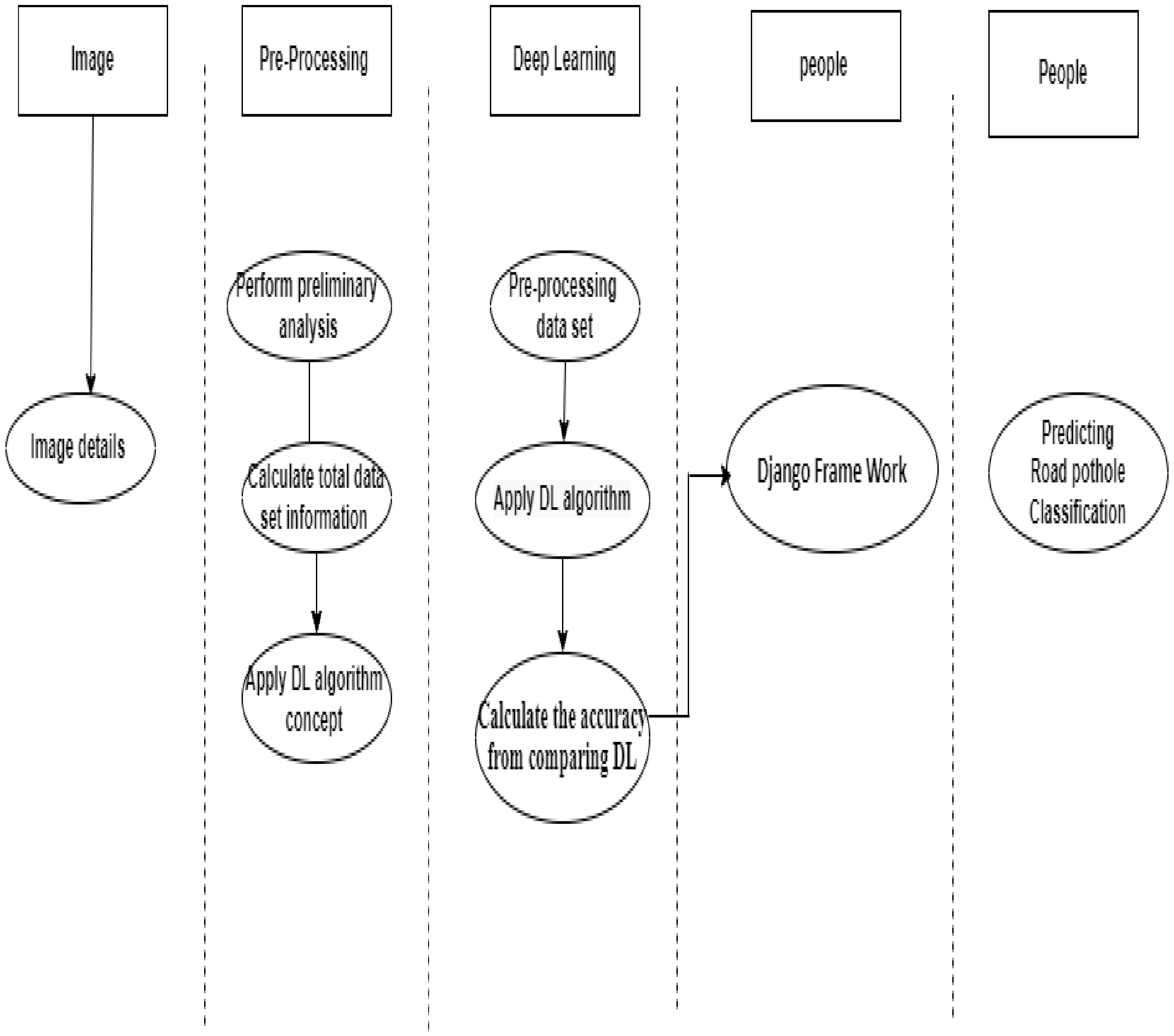


Image
Pre-Processing
Deep Learning
people
People
Image details
Perform preliminary analysis
Calculate total data set information
Apply DL algorithm concept
Pre-processing data set
Apply DL algorithm
Calculate the accuracy from comparing DL
Django Frame Work
Predicting Road pothole Classification

Fig:4.3.3 Activity diagram

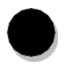
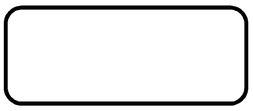
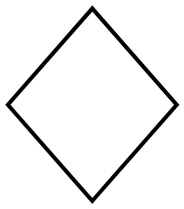
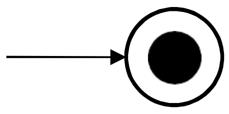
Symbol Name	Symbol	Description
Start/Initial State		A small filled circle followed by an arrow represents start point for any activity diagram.
Activity State		An action state representsthe non-interruptible action of objects.
Decisions and Branching		A diamond represents a decision with alternate paths. The outgoing alternates should be labelled with a condition or guard expression. You can also label one of the paths "else."
Final State		An arrow pointing to a filled circle nested insideanother circle represents the final action state.

Table 4.3.3 Activity Diagram Symbol Description

SYSTEM ARCHITECTURE

5. SYSTEM ARCHITECTURE

5.1 ARCHITECTURE DIAGRAM

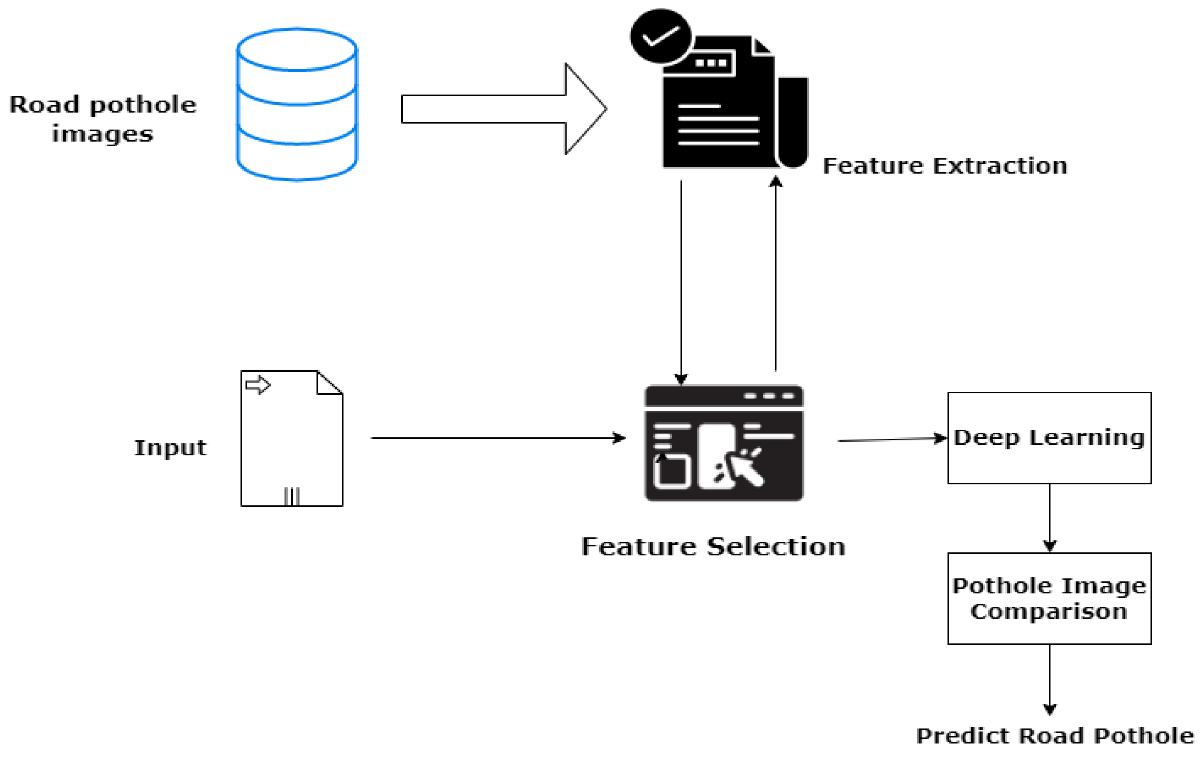


Fig:5.1 Architecture diagram

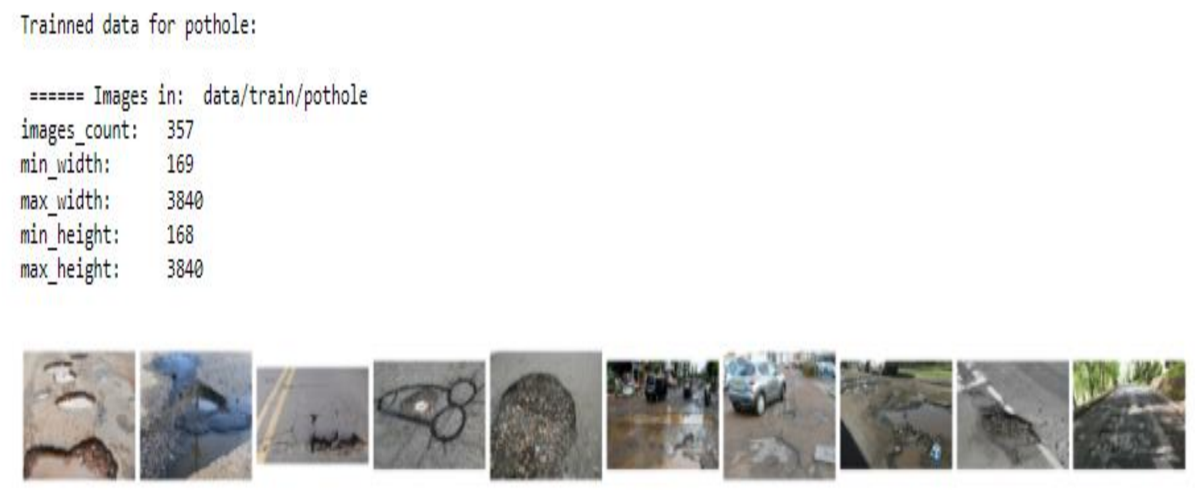
The system get an input image from the user. After getting the image the system pre-processes the input image using medium filter, After that feature Extraction and selection Now the database image is taken for CNN algorithm.

In the CNN classification, we use google net (image processing) to train the system and will classify whether the image is normal or tuberculosis image. If the system confirms a normal image then it gives the result as normal or if the system confirms that tuberculosis is present in that image then output image is show in a new window and the result of that image (pothole is present) is displayed.

5.2 MODULE DESCRIPTION

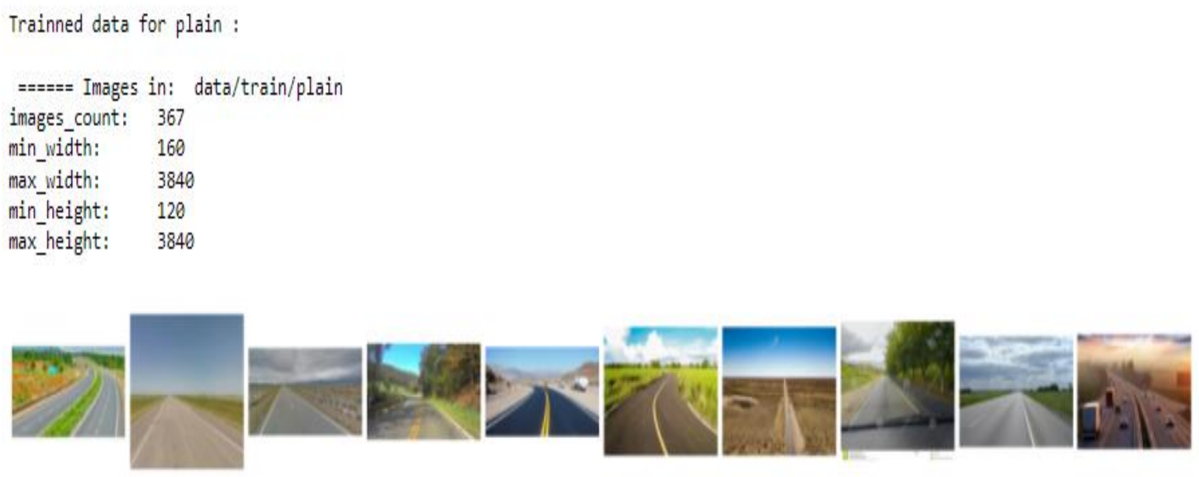
5.2.1 Import the given image from dataset

We have to import our data set using keras preprocessing image data generator function also we create size, rescale, range, zoom range, horizontal flip. Then we import our image dataset from folder through the data generator function. Here we set train, test, and validation also we set target size, batch size and class-mode from this function we have to train using our own created network by adding layers of CNN.



```
Trainned data for pothole:

 ====== Images in:  data/train/pothole
images_count:   357
min_width:      169
max_width:      3840
min_height:     168
max_height:     3840
```



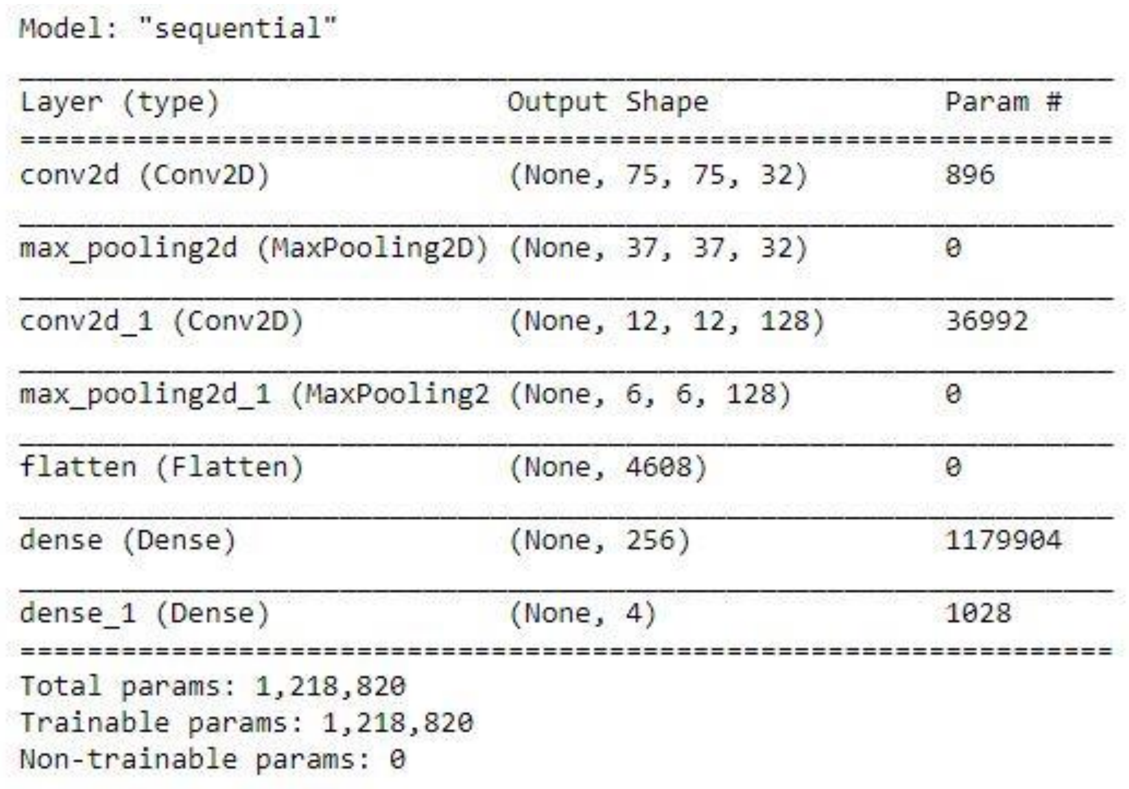
```
Trainned data for plain :

 ====== Images in:  data/train/plain
images_count:   367
min_width:      160
max_width:      3840
min_height:     120
max_height:     3840
```

Fig:5.2.1 Pre-trained image

5.2.2 To train the module by given image dataset

To train our dataset using classifier and fit generator function also we make training steps per epoch's then total number of epochs, validation data and validation steps using this data we can train our dataset.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 75, 75, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 12, 12, 128)       36992
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 1028
=================================================================
Total params: 1,218,820
Trainable params: 1,218,820
Non-trainable params: 0
```

Fig:5.2.2 CNN Model Summary details

5.2.3 Working process of layers in CNN model

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

Input layer in CNN contain image data. Image data is represented by three dimensional matrixes. It needs to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, it need to convert it into 784 x 1 before feeding into input.

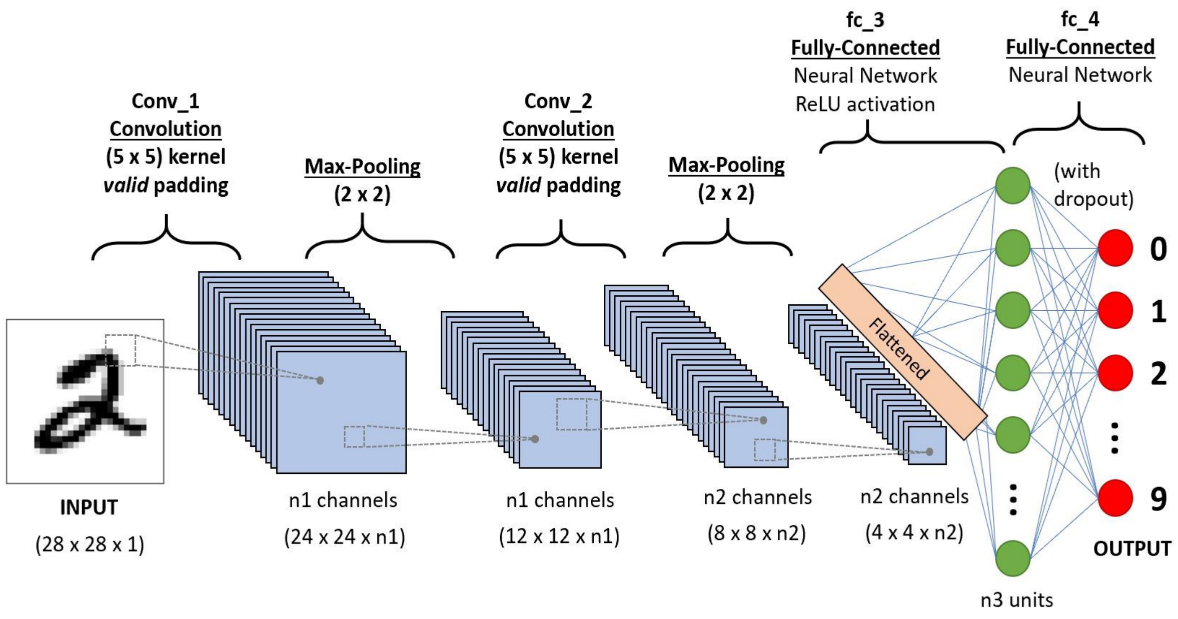


Fig:5.2.3 CNN Algorithm diagram

Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field (it is a local region of the input image that has the same size as that of filter) and the filter.

Pooling Layer

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layers. If it applies FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive.. It has applied max pooling in single depth slice with Stride of 2. It can observe the 4 x 4 dimension input is reducing to 2 x 2 dimensions.

Fully Connected Layer (FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training.

Softmax / Logistic Layer

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

Output Layer

Output layer contains the label which is in the form of one-hot encoded. Now you have a good understanding of CNN.

5.2.4 Road pothole classification identification

We give input image using keras preprocessing package. That input Image converted into array value using pillow and image to array function package. We have already classified Road Pothole image dataset. It classifies what are the Road Potholes. Then we have to predict our Road Pothole using predict function.

The Road Pothole recognition method is based on a two-channel architecture that is able to recognize classification of Road Potholes. The Road Pothole images are used as the input into the inception layer of the CNN.

5.2.5 Libraries Required

- ✓ **tensorflow**: Just to use the tensor board to compare the loss and adam curve our result data or obtained log.

- ✓ **keras**: To pre-process the image dataset.
- ✓ **matplotlib**: To display the result of our predictive outcome.
- ✓ **os**: To access the file system to read the image from the train and test directory from our machines.

TensorFlow

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of 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.

Let us now consider the following important features of TensorFlow −

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.
- It includes a high scalable feature of computation with various data sets.

Keras

Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by

Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

Matplotlib

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. Matplotlib is written in Python and makes use of NumPy, the numerical mathematics extension of Python. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

OS

The OS module in Python comes with various functions that enables developers to interact with the Operating system that they are currently working on. In this article we'll be learning mainly to create and delete a directory/folder, rename a directory and even basics of file handling.

SYSTEM IMPLICATIONS

6. SYSTEM IMPLICATIONS

6.1 CLIENT-SIDE CODING

Module 1

To build a model for training and testing:

```
import os
import numpy as np # linear algebra
import matplotlib.pyplot as plt


# Dl framwork - tensorflow, keras a backend
import tensorflow as tf
import tensorflow.keras.backend as K
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout, BatchNormalization
from tensorflow.keras.layers import Conv2D, SeparableConv2D, MaxPool2D,
LeakyReLU, Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,
EarlyStopping
from IPython.display import display
from os import listdir
from os.path import isfile, join
from PIL import Image
import glob
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Convolution2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense


import warnings
warnings.filterwarnings('ignore')


dir_name_train_plain = 'data/train/plain'
dir_name_train_pothole = 'data/train/pothole'
```

```
def plot_images(item_dir, n=6):
    all_item_dir = os.listdir(item_dir)
    item_files = [os.path.join(item_dir, file) for file in all_item_dir][:n]

    plt.figure(figsize=(80, 40))
    for idx, img_path in enumerate(item_files):
        plt.subplot(7, n, idx+1)
        img = plt.imread(img_path)
        plt.imshow(img, cmap='gray')
        plt.axis('off')

    plt.tight_layout()


def Images_details_Print_data(data, path):
    print(" ====== Images in: ", path)
    for k, v in data.items():
        print("%s:\t%s" % (k, v))

def Images_details(path):
    files = [f for f in glob.glob(path + "**/*.*", recursive=True)]
    data = {}
    data['images_count'] = len(files)
    data['min_width'] = 10**100  # No image will be bigger than that
    data['max_width'] = 0
    data['min_height'] = 10**100  # No image will be bigger than that
    data['max_height'] = 0



    for f in files:
        im = Image.open(f)
        width, height = im.size
        data['min_width'] = min(width, data['min_width'])
        data['max_width'] = max(width, data['max_height'])
        data['min_height'] = min(height, data['min_height'])
        data['max_height'] = max(height, data['max_height'])

    Images_details_Print_data(data, path)


print("")

print("Trainned data for plain :")
print("")

Images_details(dir_name_train_plain)

print("")

plot_images(dir_name_train_plain, 10)



print("")

print("Trainned data for pothole:")

print("")

Images_details(dir_name_train_pothole)

print("")

plot_images(dir_name_train_pothole, 10)


Classifier=Sequential()
Classifier.add(Convolution2D(32,(3,3),input_shape=(128,128,3),activation='relu'))
Classifier.add(MaxPooling2D(pool_size=(2,2)))
Classifier.add(Flatten())
Classifier.add(Dense(38, activation='relu'))


Classifier.add(Dense(4, activation='softmax'))
Classifier.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accurac
y'])


train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,ho
rizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)


training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(128,128),bat
ch_size=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('dataset/Test',target_size=(128,128),batch_siz
e=32,class_mode='categorical')
```

```
img_dims = 150
epochs = 10
batch_size = 32


#### Fitting the model
history = Classifier.fit_generator(
        training_set, steps_per_epoch=training_set.samples // batch_size,
        epochs=epochs,
        validation_data=test_set,validation_steps=test_set.samples // batch_size)


def graph():
  #Plot training & validation accuracy values
  plt.plot(history.history['accuracy'])
  plt.plot(history.history['val_accuracy'])
  plt.title('Model accuracy')
  plt.ylabel('Accuracy')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Test'], loc='upper left')
  plt.show()

  # Plot training & validation loss values
  plt.plot(history.history['loss'])
  plt.plot(history.history['val_loss'])
  plt.title('Model loss')
  plt.ylabel('Loss')
  plt.xlabel('Epoch')
  plt.legend(['Train', 'Test'], loc='upper left')
  plt.show()
  graph()
```

Module 2:

```
# Dl framwork - tensorflow, keras a backend

import tensorflow as tf

import tensorflow.keras.backend as K

from tensorflow.keras.models import Model

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import LeakyReLU

from tensorflow.keras.layers import Activation

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ModelCheckpoint

from tensorflow.keras.callbacks import ReduceLROnPlateau

from tensorflow.keras.callbacks import EarlyStopping

import warnings
warnings.filterwarnings('ignore')

model = Sequential()
# 1st Convolutional Layer
model.add(Conv2D(filters=96, input_shape=(224,224,3), kernel_size=(11,11),
strides=(4,4), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# 2nd Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# 3rd Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
model.add(Activation('relu'))
# 4th Convolutional Layer
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='valid'))
model.add(Activation('relu'))
# 5th Convolutional Layer
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), padding='valid'))
model.add(Activation('relu'))
# Max Pooling
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
# Passing it to a Fully Connected layer
model.add(Flatten())
# 1st Fully Connected Layer
model.add(Dense(4096, input_shape=(224*224*3,)))
model.add(Activation('relu'))
# Add Dropout to prevent overfitting
model.add(Dropout(0.4))
# 2nd Fully Connected Layer
model.add(Dense(4096))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# 3rd Fully Connected Layer
model.add(Dense(1000))
model.add(Activation('relu'))
# Add Dropout
model.add(Dropout(0.4))
# Output Layer
model.add(Dense(4))
model.add(Activation('softmax'))
model.summary()

# Compile the model
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

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

```
training_set=train_datagen.flow_from_directory('dataset/Train',target_size=(224,224),bat
ch_size=32,class_mode='categorical')
test_set=test_datagen.flow_from_directory('dataset/Test',target_size=(224,224),batch_siz
e=32,class_mode='categorical')

img_dims = 150
epochs = 1
batch_size = 32

#### Fitting the model
history = model.fit(
        training_set, steps_per_epoch=training_set.samples // batch_size,
        epochs=epochs,
        validation_data=test_set,validation_steps=test_set.samples // batch_size)

import matplotlib.pyplot as plt

def graph():
   #Plot training & validation accuracy values
   plt.plot(history.history['accuracy'])
   plt.plot(history.history['val_accuracy'])
   plt.title('Model accuracy')
   plt.ylabel('Accuracy')
   plt.xlabel('Epoch')
   plt.legend(['Train', 'Test'], loc='upper left')
   plt.show()

   # Plot training & validation loss values
   plt.plot(history.history['loss'])
   plt.plot(history.history['val_loss'])
   plt.title('Model loss')
   plt.ylabel('Loss')
   plt.xlabel('Epoch')
   plt.legend(['Train', 'Test'], loc='upper left')
   plt.show()
graph()

print("[INFO] Calculating model accuracy")
scores = model.evaluate(test_set)
print(f"Test Accuracy: {scores[1]*100}")
```

PyCharm

Views.py

```
from django.shortcuts import render

from django.http import HttpResponseRedirect
from django.urls import reverse_lazy
from django.views.generic import TemplateView
from employee.forms import EmployeeForm

from django.views.generic import DetailView
from employee.models import Employee

class EmployeeImage(TemplateView):
    form = EmployeeForm
    template_name = 'emp_image.html'

    def post(self, request, *args, **kwargs):
        form = EmployeeForm(request.POST, request.FILES)

        if form.is_valid():
            obj = form.save()

            return HttpResponseRedirect(reverse_lazy('emp_image_display', kwargs={'pk':
obj.id}))

        context = self.get_context_data(form=form)
        return self.render_to_response(context)

    def get(self, request, *args, **kwargs):
        return self.post(request, *args, **kwargs)

class EmpImageDisplay(DetailView):
    model = Employee
    template_name = 'emp_image_display.html'
    context_object_name = 'emp'
```

```
def Road Pothole(request):
    result1 = Employee.objects.latest('id')
    import numpy as np
    import tensorflow as tf
    models = keras.models.load_model('C:/Users/SPIRO73-
PYTHON/Desktop/smb/Deep_Learning/Road Pothole
Classification/Deploy/employee/e.h5')
    from tensorflow.keras.preprocessing import image
    test_image = image.load_img('C:/Users/SPIRO73-
PYTHON/Desktop/smb/Deep_Learning/Road Pothole Classification/Deploy/media/' +
str(result1), target_size=(225, 225))
    test_image = image.img_to_array(test_image)
    test_image = np.expand_dims(test_image, axis=0)
    result = models.predict(test_image)
    prediction = result[0]
    prediction = list(prediction)
classes=['plain', 'pothole']

    output = zip(classes, prediction)

    output = dict(output)

if output['plain']==1.0:

    print("plain")

elif output['pothole']==1.0:

    print("pothole")

    return render(request, "result.html", {"out": a})
```

6.2 SERVER-SIDE CODING

result.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ROAD POTHOLE CLASSIFICATION OUTPUT RESULT</title>
```

```
</head>
<style>
  label
{
font-size: 20px;
color:red;
font-family:  Algerian;
}
body
  {
    background: url(../static/image/emo1.png);
    background-repeat: no-repeat;
    background-position: center;
    background-size: cover;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    min-height: 100vh;
}
  button
{
font-size: 20px;
font-family:  wide latin;
color:black;
background-color: green;
  box-shadow: 5px 5px blue, 10px 10px red, 15px 15px green;
}
#ss
{
font-size: 20px;
color:#20fc03;
background-color: black;
font-family:  Times new roman;
}
.blink_me {
  animation: blinker 2s linear infinite;
}
```

```
@keyframes blinker {
  50% {
    opacity: 0;
  }
}
h2
{
background-color: #101010;
font-family: Algerian;
font-size: 33px;
letter-spacing: 3px;
color:#56ff00;
}
h1,h3
{
font-family: Algerian;
font-size: 33px;
letter-spacing: 3px;
color:red;
}
a
{
font-size: 20px;
color:black  ;
font-family:  Times new roman;
}
</style>
<body>
<center><h2 class="blink_me">ROAD POTHOLE CLASSIFICATION USING
ARTIFICIAL NEURAL NETWORK</h2></center><br><br><br><br>
 <marquee direction="down"><center><b><h1>{{out}}</h1></b></center></marquee>

<h3><a href="{% url 'home' %}">Go Back!!!</a></h3>

</body>
</html>
```

TESTING

7. TESTING

7.1 TESTING OBJECTIVES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

7.2 TYPES OF TESTS

7.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration.

7.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

7.2.3 Functional testing

Functional testing provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input: identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions: identified functions must be exercised.

Output: identified classes of application outputs must be exercised.

7.2.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing predriven process links and integration points.

7.2.5 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirement.

7.3 TEST CASES & REPORTS

S No	Test Cases	Expected Output	Actual Output	Status
1.	Loading dataset for Training	The dataset should be trained	Dataset is trained	PASS
2.	Loading the CNN model	The model should be loaded for training the dataset	The model has been loaded successfully	PASS
3.	Training the model	The model should be trained as per the conditions	The model has been trained	PASS
4.	Inserting the input image	The input image should be successfully uploaded for further process	The input image has been uploaded	PASS
5.	Pre-processing the input image	Pre-process the image to reduce distortion and enhance the image for further processing	Displays the pre-processed image	PASS
6.	Classification - Normal	It should display the message as “Normal”	Displays the message as “Normal”	PASS
7.	Classification - Tuberculosis	It should display the message as pothole and display the image	Displays the message as pothole and image	PASS

Table 7.1 Test Cases and Possible Results

CONCLUSION

8. CONCLUSION

8.1 RESULTS AND DISCUSSION

In this project presents experimental results and discuss the suitability of the best performing representation and model over the others. The architecture of trained model is based on the pothole classification of CNN with samples of potholes and also used on road images. In this figure A.2 and A.4 contains sample image of pothole and normal classification from the Google net model. In figure A.3 and A.5 represents the output of given images. In this testing of normal and pothole sample images for the classification on pre trained model with the prediction accuracy value of normal is 1 with no loss value and prediction accuracy value of normal and pothole is 0.8 with 0.2 was prediction loss.

8.2 CONCLUSION AND FUTURE ENHANCEMENT

It focused how image from given dataset (trained dataset) in field and past data set used identify road pothole using CNN model. This brings some of the following different gestures prediction. We had applied different type of CNN compared the accuracy and saw that LeNet makes better classification and the .h5 file is taken from there and that is deployed in Django framework for better user interface.

Road Pothole prediction to connect with AI model. To automate this process by show the prediction result in web application or desktop application. To optimize the work to implement in Artificial Intelligence environment.

APPENDICES

APPENDICES

A.1 SAMPLE SCREENSHOTS

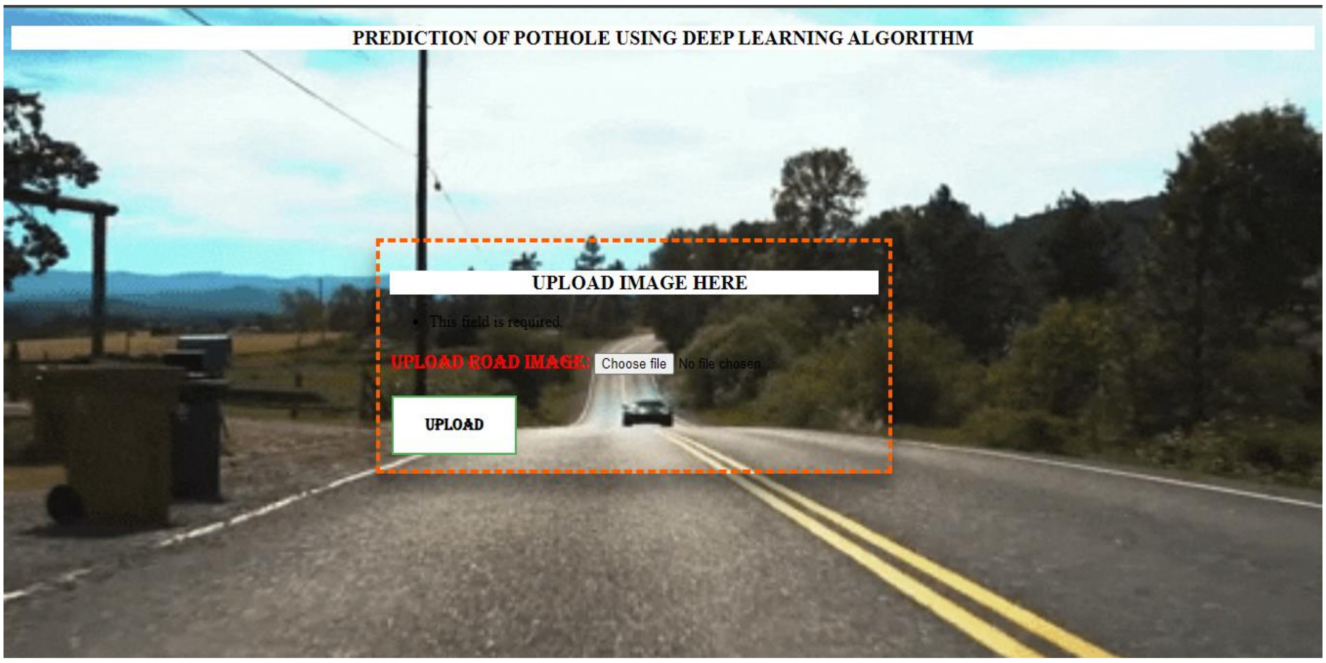


Fig: A.1 Screenshot for interface

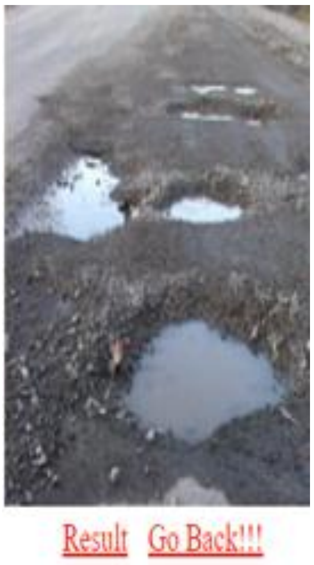


Fig: A.2 Screenshot for upload image with potholes

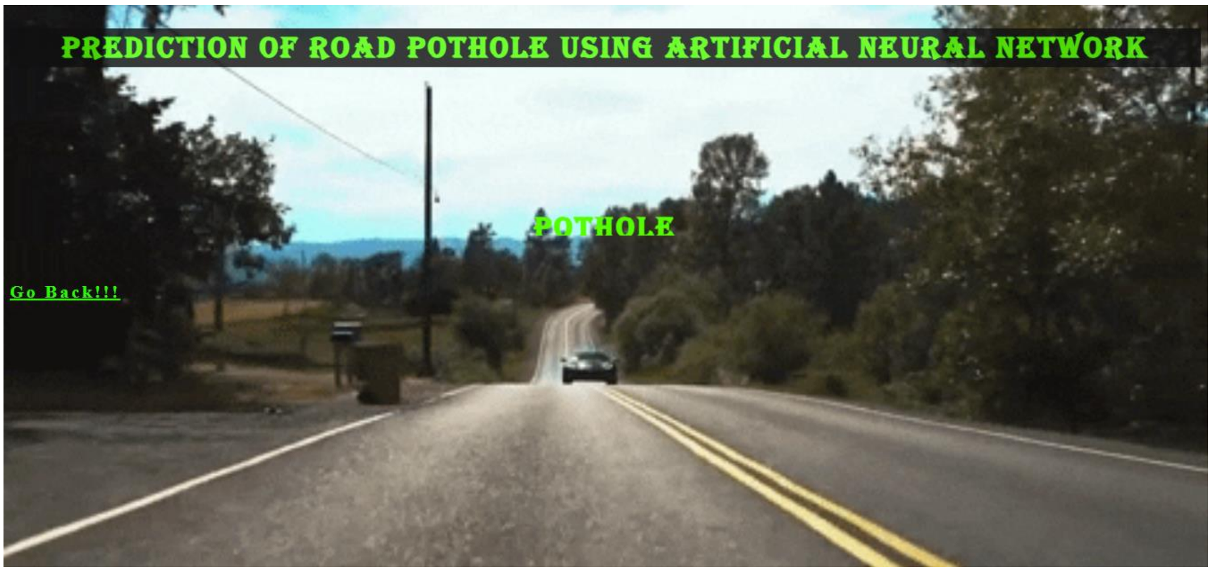


Fig: A.3 Screenshot for Result Displaying image with potholes

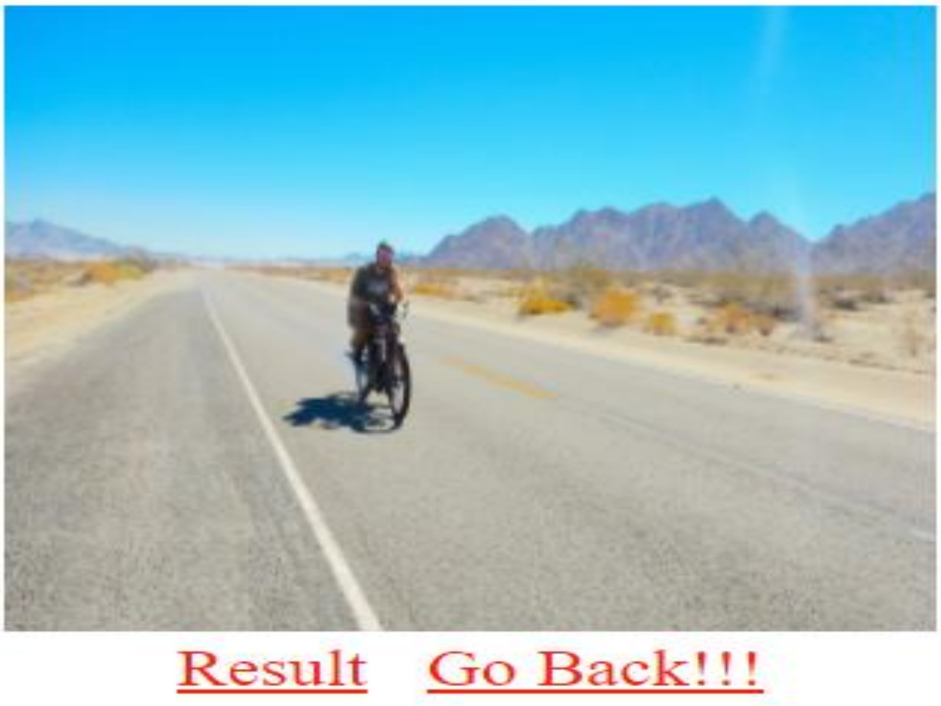


Fig: A.4 Screenshot for upload image without potholes

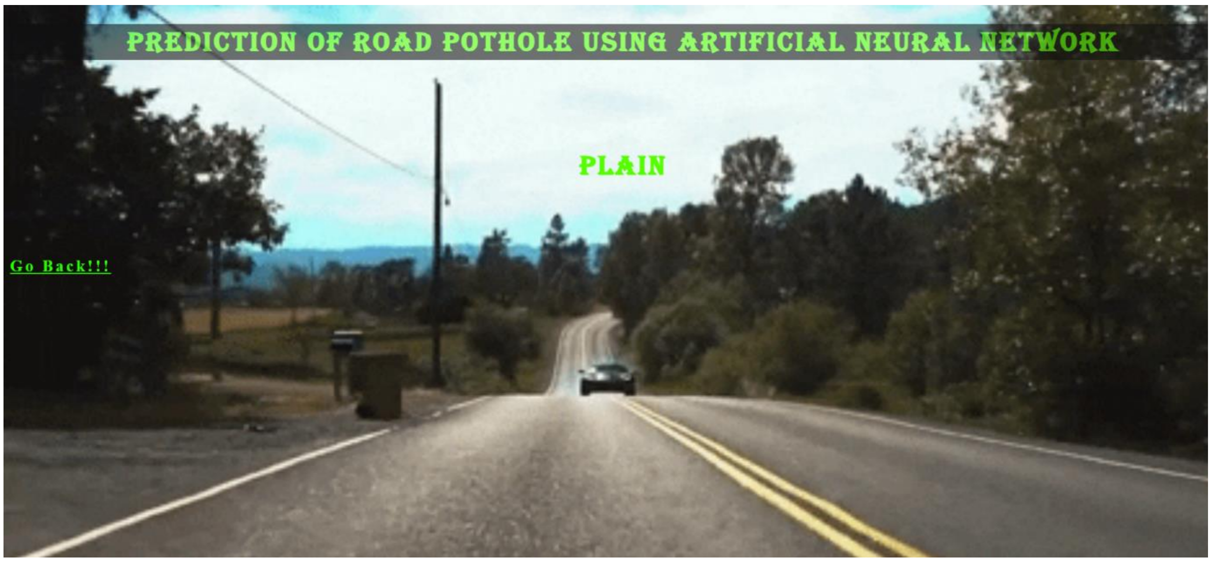


Fig: A.5 Screenshot for Result Displaying image without potholes

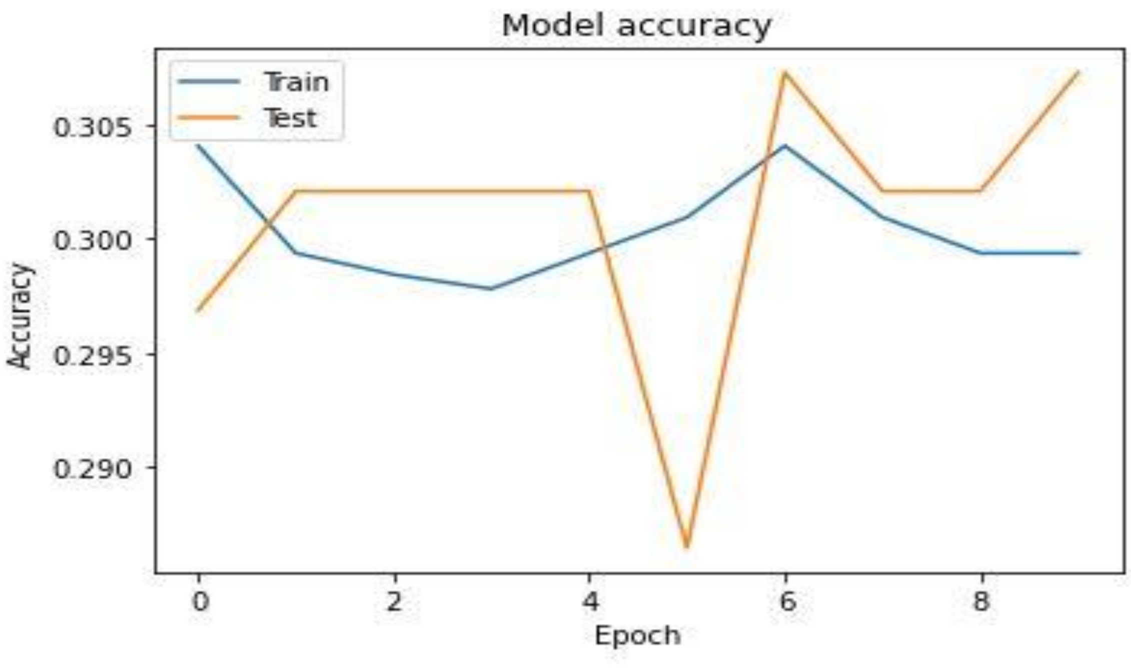


Fig: A.6 CNN model trained dataset accuracy

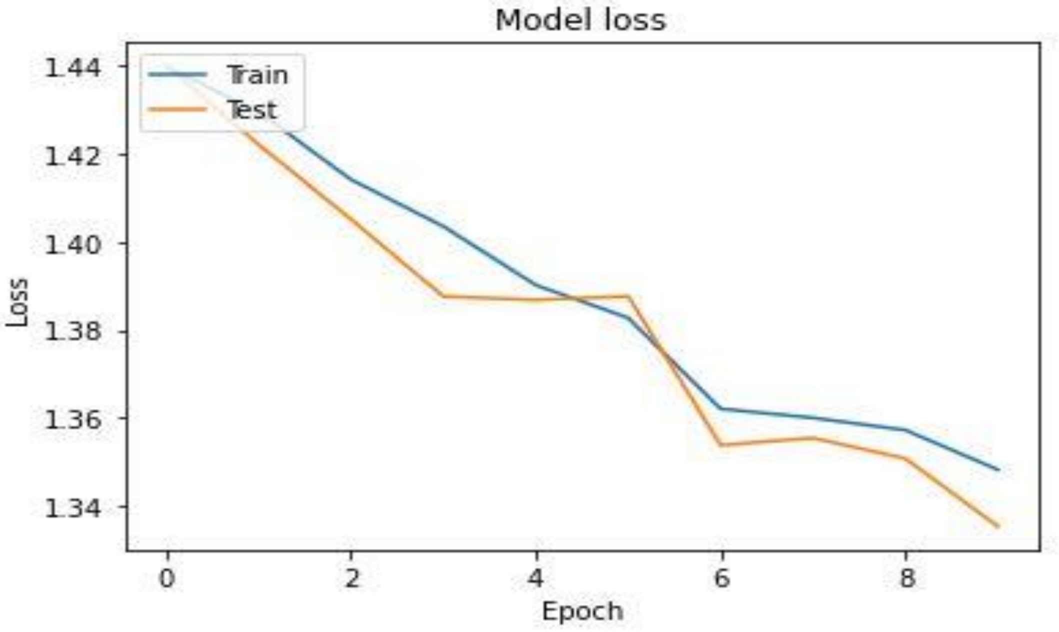


Fig: A.7 CNN model trained dataset loss values

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Hoang, N.D., (2018). An Artificial Intelligence Method for Asphalt Pavement Pothole Detection Using Least Squares Support Vector Machine and Neural Network with Steerable Filter-Based Feature Extraction. Adv. Civil Eng.

[2] K. An et al., (2018). Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving. In: Consumer Electronics (ICCE), 2018 IEEE International Conference on (pp. 1-2).

[3] S. Ryu *et al.* (2015)., Metrology and visualization of potholes using the Microsoft Kinect sensor. In: Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (2013), 1284-1291.

[4] Akagic *et al.* (2017). Convolutional neural networks: an overview and application in radiology. Insights Imaging 9(4), 611–629. Advance online publication. doi:10.1007/s13244-018-0639-9

[5] Vinayakumar Ravi Et al (2018). tagging system. In: Proceedings of the 4th Robotics and Mechatronics Conference of South Africa (2011), 1-4.

[6] Yu and Salari (2016). "Detecting Road Irregularities by Image Enhancements and Thresholding," IEEE EUROCON 2019 -18th International Conference on Smart Technologies, Novi Sad, Serbia, 2019, pp. 1-6, doi: 10.1109/EUROCON.2019.8861790.

[7] Ouma et al (2013). "Experiment of Image Processing Algorithm for Efficient Pothole Detection," 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2019, pp. 1-2, doi: 10.1109/ICCE.2019.8661973

[8] Silvister et al (2016). "Deep Integration: A Multi-Label Architecture for Road Scene Recognition," in IEEE Transactions on Image Processing, vol. 28, no. 10, pp. 4883-4898, Oct. 2019, doi: 10.1109/TIP.2019.2913079.

[9] Amita Dhiman and Reinhard Klette (2018). How Deeply to Fine-Tune a Convolutional Neural Network: A Case Study Using a Histopathology Dataset. Applied Sciences. 2020; 10(10):3359. https://doi.org/10.3390/app10103359

[10] Panop Khumsap (2016). "Classification of Paved and Unpaved Road Image Using Convolutional Neural Network for Road Condition Inspection System," 2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA), Krabi, Thailand, 2018, pp. 165-169, doi: 10.1109/ICAICTA.2018.8541284.