

PROJECT REPORT

Of

Pothole Detection System

Submitted in partial fulfillment of the requirements
for the award of the degree

MASTER OF COMPUTER APPLICATIONS

Of

KLE TECHNOLOGICAL UNIVERSITY

By

Mr. Manjunath Hinglaje

SRN: 01FE20MCA013

Under the guidance of

Prof. A.K. Chikaraddi



**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
KLE TECHNOLOGICAL UNIVERSITY**

Vidyanagar, Hubballi-580031 Karnataka.

2021 – 2022

**DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS
KLE TECHNOLOGICAL UNIVERSITY**



CERTIFICATE

This is to certify that the project work entitled “Pothole Detection System” Submitted in partial fulfillment of the requirements for the award of a degree of Master of Computer Applications of **KLE Technological University, Hubballi, Karnataka** is a result of the bonafide work carried out by Mr. Manjunath Hinglaje SRN: 01FE20MCA013 During the academic year 2021-2022

Dr. A.K Chikaraddi.
Asst.Professor,
MCA Department

Dr. P R Patil. ME, Ph.D.
Professor, and Head
MCA Department

Prof. N H Ayachit
Registrar,
KLE Technological
University. Hubli.

Viva-Voce Examination

Name of the Examiners

1. _____
2. _____

Signature with Date

ACKNOWLEDGEMENT

Every successful completion of any undertaking would be complete only after we remember and thank the almighty, the parents, the teachers, and the personalities, who directly or indirectly helped and guided us during the execution of that work. The success of this work is equally attributed to all well-wishers who have encouraged and guided throughout the execution.

I express my deepest gratitude to **Prof. Ashok K Chikaraddi** for their guidance and assistance throughout the project with great interest.

I avail this opportunity to express my deepest gratitude to **Dr. P. R. Patil, HOD** for encouraging us with his valuable guidelines.

I am grateful to **Prof. N. H. Ayachit**, Registrar for his blessings and to **KLE Technological University, Hubballi** which has given me a bright future. I would like to thank all the people for their guidance and valuable suggestions throughout the project.

I also thank all teaching and non-teaching staff members of the MCA department for their invaluable cooperation.

Finally, I would like to express our sincere thanks to our **Parents and Friends** for their enormous encouragement and all others who have extended their helping hands towards the completion of our project.

Mr. Manjunath Hinglaje

ABSTRACT

Potholes are an unavoidable obstacle that all motorists in India face, once the rains have begun which carry on growing day by day. Unfortunately, there is no system to pin point the exact location of potholes, moreover fixing them is labor intensive and expensive, and hence system that will detect the potholes and other irregularities using a deep learning-based model that can detect potholes early using images and videos which can reduce the chances of an accident. This model is basically based on Transfer Learning, Faster Region-based Convolutional Neural Network(F-RCNN) and Inception-V2. There are many models for pothole detection that uses the accelerometer (without using images and videos) with machine learning techniques, but a smaller number of pothole detection models can be found which uses only machine learning techniques to detect potholes. The results of this work have shown that our proposed model out performs other existing techniques of potholes detection.

INDEX

TABLE OF CONTENTS

1. INTRODUCTION.....1-8

- 1.1 Overview
- 1.2 Motivation
- 1.3 Objectives
- 1.4 Literature Survey
- 1.5 Existing System Drawbacks

2. PROPOSED SYSTEM9-10

- 2.1 Proposed System.
- 2.2 Advantages
- 2.3 Disadvantages
- 2.4 Scope

3. SOFTWARE REQUIREMENT SPECIFICATION10-14

- 3.1 Overview
- 3.2 Requirement Specifications
 - 3.2.1 Functional Requirements
 - 3.2.2 Use case diagrams
 - 3.2.3 Nonfunctional Requirements
 - 3.2.3.1 Correctness

- 3.2.3.2 Reliability
- 3.2.3.3 Robustness
- 3.2.3.4 Maintainability
- 3.2.3.5 Portability
- 3.3 Software and Hardware requirement specifications

4. SYSTEM DESIGN..... 15-19

- 4.1 Architecture of the system
- 4.2 Data flow Diagram and Levels of DFDs
- 4.3 Level 0 DFD
- 4.4 Level 1 DFD
- 4.5 Level 2 DFD

5. IMPLEMENTATION20-35

- 5.1 Proposed Methodology
- 5.2 Block Diagram
- 5.3 Algorithm Design
- 5.4 Source Code

6. TESTING.....36-37

Test Plan and Test Cases

7. SNAP SHOTS.....38-43

8. CONCLUSION AND FUTURE SCOPE.....44

REFERENCES.....45

APPENDIX.....46-47

CHAPTER - 1

INTRODUCTION

1.1 Overview

India has the world's second-largest road network. As a result, the road network is critical to India's economic development and social functioning. According to the report, the Road Transport sector's annual average GDP growth rate was close to 10% in the last ten years, compared to the general annual GDP growth rate of 6%. The Indian government is currently constructing roads at a breakneck speed. However, because of the poor drainage system and overloaded trucks, road upkeep is a difficult undertaking. Potholes arise on the road as a result of poor road maintenance, resulting in road accidents. According to data provided by the Indian government, potholes claimed the lives of 11,836 individuals and wounded 36,421 others between 2013 and 2016. Pothole issues are difficult to manage since practically every year, floods, disasters, excessive rainfall, and other natural calamities strike almost every part of the country. We may not be able to maintain the road, but we can help to limit the number of accidents that are increasing every year. Using deep learning, this research proposes a revolutionary method for detecting potholes up to 100 meters ahead. This is a problem that many authors are working on. M. Artis et al. have developed an accelerometer-based model that employs the Z-THRESH, Z-DIFF, STDEV(Z), and G-ZERO algorithms and can be used on Android OS devices with minimal hardware and software resources. They examine the model's performance using a 90 percent true positive value. Lin, J. et al. used a non-linear SVM model classification tool with a Gaussian radial basis function to detect a pothole in their research. Each of the 64 * 64 images in this model is converted to grayscale and used in the experiment. Before using the SVM model, each grayscale eigen value is determined, and an average range of eigenvalues between 60 and 100 is searched. After that, eigenvector is used to normalize each image in the range. Pereira, V. et al. employed Convolution Neural Networks and compared their model's performance to that of SVM, finding that their model outperformed SVM with 99.80% accuracy. They used CNN, pooling, ReLU activation function, Adam Optimizer, and Sigmoid function to deploy the model. Here, convolution and pooling have been used for feature extraction. Adam optimizer is used to reduce the cost function and sigmoid function for output prediction.

1.2 Motivations

Potholes have long been a problem, posing a threat to safe street movement. To address the problem, we propose this system, which incorporates machine learning and image processing. This will detect the potholes, allowing us to effectively deal with the problem. Because of these reasons it is very important to get the information of such bad road conditions, collect this information and distribute it to other vehicles, which in turn can warn the driver. But there are various challenges involved in this. First of all there are various methods to get the information about the road conditions. Then this information must be collected and distributed to all the vehicles that might need this information. Lastly the information must be conveyed in the manner which can be understood and used by driver. We in this project try to design and build such a system.

1.3 Objectives

To resolve the drawbacks and get good Accuracy. Here we have good dataset which will allow to find the Potholes. The main Objective of this project is To Detect the Potholes which are Present on the Road.

- We will collect the Pothole Images from Different Road from Cities.
- Based On the Data Collected we will split the Data as Train and Test.
- Apply the Machine Learning Model for the Collected Data.
- Model Will Calculate the Damage Percentage.
- Based On the Result we Received. Retrain the model to Improve the Accuracy.

1.4 Literature Survey:

Vision technologies offer cost-effective ways to automate jobs in a variety of engineering fields, including transportation, agriculture, and industry. This section depicts some of the research attempts that have been made to detect potholes in roadways automatically. The pothole detection techniques are classified into four approaches: sensor-based techniques, Sensor-based techniques, 3D reconstruction techniques (laser-based and stereo vision-based), image processing techniques, and model-based techniques are the four types of pothole detection techniques (machine-learning and deep learning).

1) A REVIEW PAPER ON EXISTING POTHOLE DETECTION METHODS:

Proposed an economical model to examine 3D pavement distress images. The computation cost is decreased by making utilization of a low-cost Kinetic sensor which gives the direct depth measurement. The sensor comprises of an IR camera and an RGB camera capturing depth images and RGB pictures which are examined under MATLAB environment by extracting the metrological and the characteristic features to establish the depth of the potholes. developed a model making use of LED direct light and 2 Charge Coupled Device cameras to identify the 3D cross-section of potholes in pavement. It uses different advanced image processing techniques including. image pre-processing, binarization, thinning, error analysis & compensation and 3D re-construction to get the depths of potholes. The constraint in this model is that the outcomes get influenced by LED light intensity and natural elements.

Author Name	Advantages	Disadvantages
Prof. A.P. Singh	For detecting potholes, the device employs a GPS sensor and a three-axis accelerometer. The GPS sensor and the 3-hub accelerometer's outputs.	Also does not work well at huge reflections.

2) Automatic and real-time Pothole detection and Traffic monitoring system using Smartphone Technology:

It is a distributed mobile sensor computing system called CarTel. This system includes a set of sensors installed in vehicles to collect and process data and send it to portal based upon the continuous queries which are processed by continuous query processor on remote nodes. It uses sensors like GPS for monitoring the movements of vehicles. CarTel includes, CafNet, a networking stack that uses opportunistic connection (e.g. WiFi, Bluetooth) to transfer information between portal and remote nodes. This information can be used for various applications such as time of travel, route planning. CarTel currently does not offer a way to aggregate information gathered across different users and it does not include machine learning; it just replies to the queries based upon the data stored in relational database.

RCM-TAGPS system is collects the sensor data using three-axis accelerometer and GPS. The sensor data has 4- tuples: current time, location, velocity and three direction accelerations. This system also does the data cleaning before processing or analyzing it to deal with technical challenges like GPS error, and transmission error. This system analyses the Power Spectral Density (PSD) to detect pavement roughness using Fourier transform. The International Roughness Index (IRI) is calculated based upon PSD. The pavement roughness is then classified in four levels (excellent, good, qualified and unqualified) according to, the Technical Code of Maintenance for Urban Road CJJ36-2006, one of the industry standards in the People's Republic of China. This standard evaluates the pavement roughness by Riding Quality Index (RQI). Based upon the value of RQI, the pavement roughness is classified. The system provides the evaluation of a section of road based upon its roughness. However, this system does not provide the proper location of pothole, bump or manhole.

Author Name	Advantages	Disadvantages
Prof. Adewole K.S Prof. Olayiwola W. Bello Prof. Abimbola Akintola	Determine the 3-axis acceleration, and magnetic vectors using the accelerometer sensors in device.	A larger and more representative test data must be collected to concretely evaluate such a system as this

3) A Real-Time Pothole Detection Approach for Intelligent Transportation System:

Proposed Approach This proposes a pothole detection approach which combines and improves the Z-THRESH and G-ZERO approaches to detect pothole. Furthermore, the Z-DIFF and STDEV(Z) approaches are limited in accordance with time differences and time periods, so these two approaches are not adopted. The pseudocode of the proposed pothole detection approach is presented in Algorithm 1. The input parameters of this proposed approach are three-axis accelerometer data, and the value of output is 1 when the proposed pothole detection approach supposes the car passed through a pothole. In the proposed approach, the parameter check method is used to record whether the value of $f1(ga,i,j)$ or $f4(ga,i,j)$ is 1. When one of Z-THRESH and G-ZERO approaches supposes that the car passed through a pothole, the timestamp ti,j is recorded and compared with the parameter check time. The value of output is 1 if $ti,j-check\ time$ is smaller than ε seconds, which means a pothole is detected. Furthermore, the parameter check time can be trained and learned by historical data from each practical run.

Author Name	Advantages	Disadvantages
Prof. Hsiu-Wen Wang Prof. Chi-Hua Chen	Send pothole, sensor data, and braking event data to a central web server.	It is needful to obtain traffic exposure data to accurately assess the current challenge of road traffic anomalies accidents

4) Sensor-Based Pothole Detection Approaches:

There are multiple research efforts to detect potholes using various vibration sensors (such as ICP accelerometer or PC-oscilloscope) mounted to motorcycles, vehicles, and buses to collect accelerated data to estimate pavement surface conditions. The vibration sensors could be built-in or external to a PC. Eriksson used GPS sensors and 3-axis accelerometers to collect data and used a machine-learning approach to identify severe road surface irregularities and potholes from accelerometer data (e.g., input x and z axis acceleration and vehicle speed). Five consecutive filters were studied: z-peak, x z-ratio, speed, high-pass, and speed vs. z ratio. These filters were used as well to exclude the generated data from events such as crossing railways and door slamming. To reduce the number of features, researchers used backward and forward selection, genetic algorithm, and support vector machine using principal component analysis. Sensor-based pothole detection methods are not efficient techniques because: (1) they are not suitable to be implemented on devices with limited hardware, (2) they may suffer from false positives as the joints of road could be detected as potholes and false negatives as the potholes in the center of a lane cannot be detected because they are not hit by any of the vehicle's wheels [10], (3) they cannot detect potholes until the vehicle pass over them, and (4) they lack information about the area and shape of potholes.

Author Name	Advantages	Disadvantages
Prof. Ramsha Suhail Prof. Faraz Ahmed Prof. Harleen Boparai	This active model of the presented system was assessed in artificial environment with unnatural potholes and humps.	A larger and more representative test data must be collected to concretely evaluate such a system as this.

5) Image Processing Pothole Detection Techniques:

The image processing object detectors are dependent on hand-crafted representations to extract low-level features. There were several previous image-processing research efforts to detect potholes in a single image/frame, and other video-based methods were proposed to detect potholes and count their number over a series of frames. The authors in collected different frames and converted the frames into blurring grayscale images and then applied morphological and edge detection methods to identify con-tours that are run through a Hough transform algorithm to extract features. Ouma et al. applied fuzzy c-means clustering algorithm and morphological reconstruction techniques to 2D color images to detect potholes on asphalt pavement. In addition, Nien aberet al. used image processing to identify the potholes on roads and reject unwanted objects such as vehicle and plants from the image. Frames are processed by simple image processing techniques such as Canny filters and contour detection to locate potholes. The experiments resulted in precision of 81.8% with recall of 74.4%. Although the accuracy values are satisfactory in the test images, it is not guaranteed that using the same techniques in all type of roads will result in the same accuracy. The authors in detect potholes in three stages:(1) pre-processing to extract the dark areas from a grayscale image, candidate extraction to find the vanishing point to create virtual lanes, and cascade detector to extract the pothole region using some threshold values. This technique achieved 88% accuracy with recall of 71%. Similarly, in, the authors detect potholes in three stages: (1) segmentation using histograms and morphology filters to extract dark regions, (2) candidate region extracted using various features, such as size and compactness, and (3) decision making as to whether candidate regions are potholes through comparing pothole and background features.

Author Name	Advantages	Disadvantages
Prof. S Nienbar Prof. M.J Booyesen	The proposed method for detecting a pothole starts by converting the extracted road section image to a grayscale image.	As the range in which it will detect a pothole in front of the vehicle.

6) Model-Based Approaches for Potholes Detection Techniques:

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. 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 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 pot-holes. Hoang 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. 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. The stated machine learning approach achieved significant accuracy, although they encountered the following challenges: (1) manual feature extraction must be performed by experts to improve the accuracy performance during the pothole detection process, and (2) they required high computational power, which are not feasible to be used by drivers in their devices. Deep learning (DL) approaches provide an alternative solution that automatically processes features extraction and classification simultaneously through convolutional neural network (CNN) operations.

Author Name	Advantages	Disadvantages
Prof. S Nienbar Prof. M.J Booysen	The proposed method for detecting a pothole starts by converting the extracted road section image to a grayscale image.	As the range in which it will detect a pothole in front of the vehicle.

1.5 Existing System Drawbacks:

- Low efficiency.
- It has limits when it comes to measuring information like pothole volume and depth.
- Lightning and shadow conditions have an impact on it.
- The sensor and vehicle used in the data collection procedure have an impact.

CHAPTER – 2

PROPOSED SYSTEM

The assembly of an operational group of computer programs that will perform, without modification, a significant portion of the functional requirements contained in this RFP. The Proposed System should include system interfaces and conversion tools as well as Contractor supplied or recommended third party software products required to properly design, develop, test, train, implement, interface, tune, and operate the Proposed Solution. The Proposed System should include document management, workflow, rules engine, claims management, risk management analytics engine, and a customer relationship management functionality.

2.1 Proposed System

Proposed Pothole-maintenance system with a pothole detector that Requires High-Definition Image/Video as input and Detect the Potholes. Using the Deep Learning, Image Processing Technique and Convolution Neural Network (CNN). The System Calculate the Damage Percentage of the Road. It helps to the User to Select the best Possible way to reach their destination Safely. Pothole detection is one of the emerging and trending method to find the potholes on roads. Prediction is always associated with inaccuracy. Therefore, efficiency of algorithm to predict the pothole density can be increased to predict accurate results.

2.2 Advantages

- This technique assists us in avoiding dangerous potholes and thereby terrible accidents.
- Pothole-related accidents can be avoided.
- Potholes will be communicated to the driver.
- If a pothole is identified, the automatic speed can be adjusted.
- Enhanced safety and security provided.

2.3 Disadvantages

- Surprisingly sensitive to the noise in Video Clip.
- The extracted component is heavily weighted in the categorization.
- High computational complexity.
- With the impact of light and shadow, extraction becomes difficult.
- The extracted component is heavily weighted in the categorization.

2.4 Scope

The project's ultimate goal is to create a generic system that can detect potholes and other abnormalities using a deep learning-based model that can detect potholes early using photos and videos, reducing the risk of an accident. Transfer Learning, Faster Region-based Convolutional Neural Network (F-RCNN), and Inception-V2 are the main components of this model. There are many potholes recognition models that combine the accelerometer with machine learning techniques (without requiring photos or videos), however there are fewer pothole detection models that use simply machine learning techniques.

CHAPTER - 3

SOFTWARE REQUIREMENT SPECIFICATION

The main work of providing software requirement specifications (SRS) is to build a bridge for communication between the people involved in project development. This part of the report describes the plays that are played by various users of the system, the functional overview of the project, characteristics regarding inputs and outputs, and also the non-functional overview of the project.

3.1 Overview of SRS

This section discusses various software and hardware aspects of the project which has an impact on the end result; they are classified into the following various sections, and each section lights on various parameters being encountered in the system implementation.

3.2 Requirement Specification

A software requirements specification (SRS) is a document that explains what the software will accomplish and how it will function. It also explains what features the product must have in order to meet the needs of all stakeholders (both business and users).

3.2.1 Functional requirements

This is specification is used to specify the requirements for the initial implementation of the system and updates the system in future. The Software requirements specifications bridges gap between client/user and the developer. This is the documentation that describes the user needs accurately.

i) Input- Image.

Processing- Soon After Giving the Image as Input It will Start Detecting the Potholes.

Output- Potholes Will be Detected.

ii) Input- Video Clip.

Processing- Soon After Giving the Video as Input It will Start Detecting the Potholes.

Output- Potholes Will be Detected.

3.2.2 Use Case Diagram

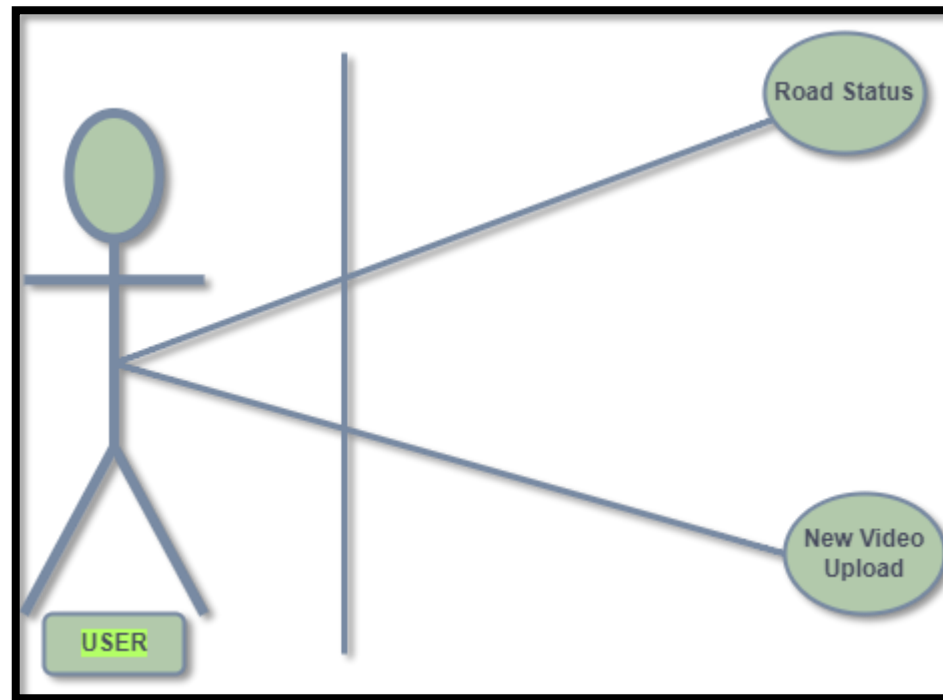


Fig 1. use case diagram

The above diagram Fig.1 shows various privileges being assigned to end-users, to use the application end-users need to launch the web app users can upload the Image/Video and view the result.

3.2.3 Non-Functional Requirements

Description:

System qualities such as security, reliability, performance, maintainability, scalability, and usability are defined by non-functional requirements (NFRs). Across the many backlogs, they serve as constraints or restrictions on the system's architecture.

3.2.4.1 Correctness: In this project, care is taken utilizing business rules to ensure only valid data is accepted using appropriate sensors

3.2.4.2 Reliability: The proposed project works well in all environments; it's being tested for various scenarios Users Launch web app Input the image Get it processed View result12 Plant Disease Identification

3.2.4.3 Robustness: The code takes care to deal with unexpected cases using alerts

3.2.4.4 Maintainability: The project works fine with the given requirements; new requirements could be done with the assistance of the developer.

3.2.4.5 Portability: This application works on all platforms irrespective of operating system and machine details.

3.3 The Hardware and Software Requirements

Hardware Specification

Processor	Intel Core i3 or above
RAM	8GB or Higher
Hard disk	500 GB

Software Specification

IDE	PyCharm
Front end	HTML and CSS
Back end	Python
Operating System	Windows 10 or Higher
Libraries	Open CV, Keras, TensorFlow

- **OpenCV**

It is the most widely used open-source computer vision and machine learning software library, with a focus on real-time computer vision. By providing hundreds of computer vision algorithms, OpenCV provides a standard framework for computer vision applications.

- **TensorFlow**

Tensor Flow is a deep learning framework for on-device machine learning inference that is open-source. It offers developers machine learning models for classification, detection, regression, and other machine learning procedures. It allows for minimal latency and tiny binary file sizes when executing models on mobile, embedded, and IoT devices.

- **Keras**

Keras is a Python interface for artificial neural networks created by an open-source software package. Keras serves as a user interface for TensorFlow. Keras supported a variety of backends up until version 2.3, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

- **Flask**

Flask is a Python-based microweb framework. It is referred to as a microframework because it does not necessitate the usage of any specific tools or libraries. It doesn't have a database abstraction layer, form validation, or any other components that rely on third-party libraries to do typical tasks.

CHAPTER - 4

SYSTEM DESIGN

4.1 Architecture of the system

A **system architecture** is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

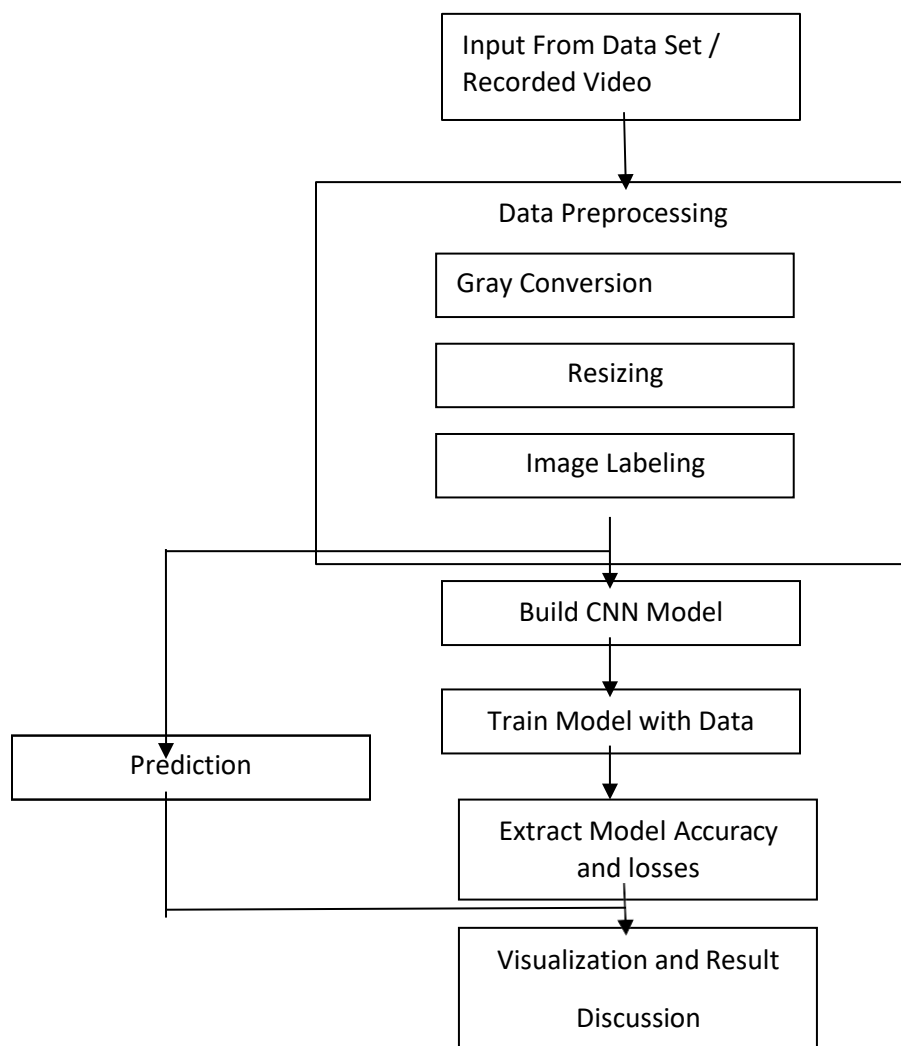


Fig 2. Architecture of the system

- **Image Pre-processing**

Picture scaling and image segmentation are the first two steps in the process. The quality of pothole detection is unaffected by image size. As a result, we reduce the size of an image by times to reduce the number of computations.

- **Data Processing**

Data Processing is the task of converting data from a given form to a much more usable and desired form i.e., making it more meaningful and informative. Using Machine Learning algorithms, mathematical modeling, and statistical knowledge, this entire process can be automated. The output of this complete process can be in any desired form like graphs, videos, charts, tables, images, and many more.

- **Gray Scale Conversion**

Will take the image that we have loaded with our preceding function, convert the image from color to grayscale, and resize the image's width and height. These steps are not strictly necessary, but both steps are done in many cases for efficiency purposes. Using RGB color images instead of grayscale increases the amount of data to be processed.

- **Resizing the Image**

Resizing images is a critical preprocessing step in computer vision. Principally, our machine learning models train faster on smaller images. An input image that is twice as large requires our network to learn from four times as many pixels — and that time adds up. Moreover, many deep learning models architectures require that our images are the same size and our raw collected images may vary in size.

- **Image Labeling**

Image labeling is the process of identifying and marking various details in an image. Image labeling is useful when automating the process of generating meta data or making recommendations to users based on details in their images.

- **Prediction**

Prediction in machine learning refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome.

- **Building CNN Model**

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on an underdone image and do not need any preprocessing.

- **Training the model with Data**

ML models can be trained to benefit manufacturing processes in several ways. The ability of ML models to process large volumes of data can help manufacturers identify anomalies and test correlations while searching for patterns across the data feed. It can equip manufacturers with predictive maintenance capabilities and minimize planned and unplanned downtime.

- **Model Accuracy**

Model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

- **Region of Interest Extraction**

The region of interest is the area on which potholes can be found, i.e., asphalt pavement. The B component of the RGB color space from the previous phase, which is a black-and-white image, is used as the input for this stage (0,255). To begin, we create a new matrix containing the pixel coordinates of all white pixels in a picture. Second, we choose a dynamic number of pixels for the first level seed points based on the image's standard deviation ($\sqrt{2}$). The method for determining the first level seed points. We introduce second-level seed locations to improve the accuracy of asphalt pavement detection. The values of adjacent pixels from the first level are used to create the second level. The region of interest is determined by the coordinates of these sites.

- **Pothole Detection**

It detects the pothole in current frame using Pre-Trained Model. If Pothole is in frame, it gives 1 otherwise its value is 0.

- **Calculation**

To calculate the damage area, it takes the total number of damaged frames in video to total number of frames.

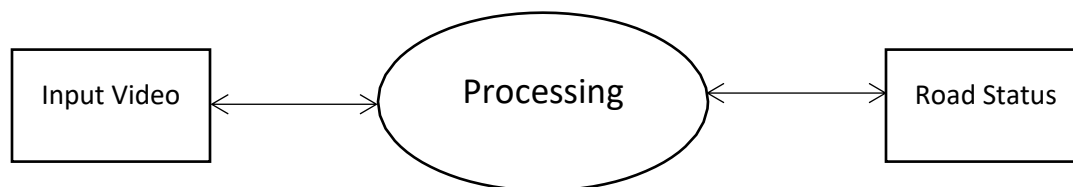
$$\text{Damage (\%)} = \frac{\text{Total No. of Damaged Frames}}{\text{Total No. of Frames}} \times 100$$

4.2 Detailed DFD for the Proposed System

Data Flow Diagram: A data flow diagram shows the way information flows through a process or system. It includes data input and output. Data stores and the various sub processes the data moves through. DFD are built using standardized symbol and notation to describe various entities and their relationships.

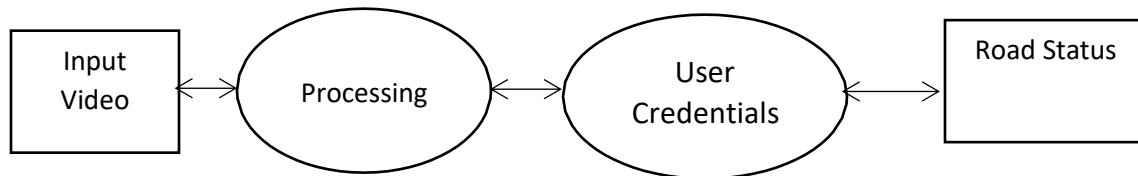
Data flow diagram visually represents systems and processes that would be hard to describe in a chunk of text. You can use these diagrams to map out an existing system and make it better or to plan out a new system for implementation. Visualizing each element makes it easy to identify inefficiencies and produce the best possible system.

4.2.1 Zero Level DFD

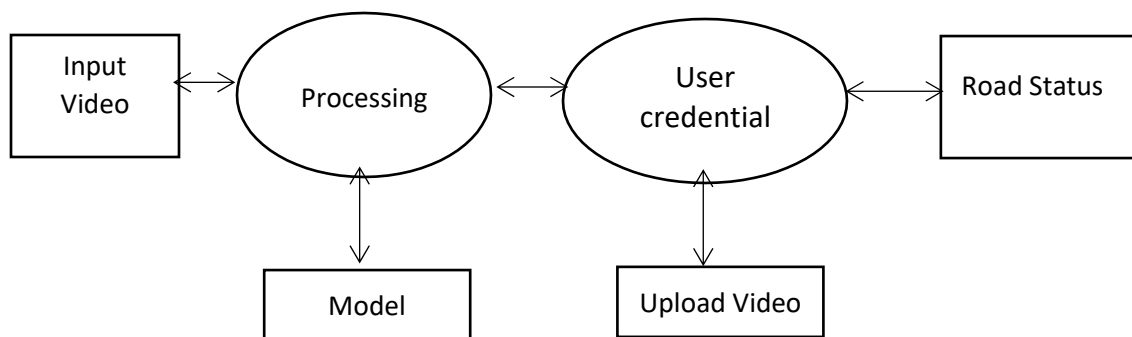


The above diagram shows the LEVEL 0 DFD of the proposed system, in the proposed system various modules associated with the system are displayed. The image dataset collected is fed to the classification system, it gets processed using RESNET model using ImageNet weights, and the end result is saved.

- **First Level DFD**



- **Second Level DFD**



CHAPTER - 5

IMPLEMENTATION

5.1 Proposed Methodology

The goal of this research was to distinguish between pothole and non-pothole incidents. As a result, it was a binary categorization issue. The data collecting and labelling of pothole and non-pothole events were the first stages of the research. As a result, two custom Android applications were created (apps). The first app records data from the accelerometer, gyroscope, and GPS, while the second app assists with data labelling. Before extracting key features ready to train with the machine learning model, the raw sensor data from both apps was preprocessed, combined, cleansed, and separated (Training/Validation and Test). The research methodology.

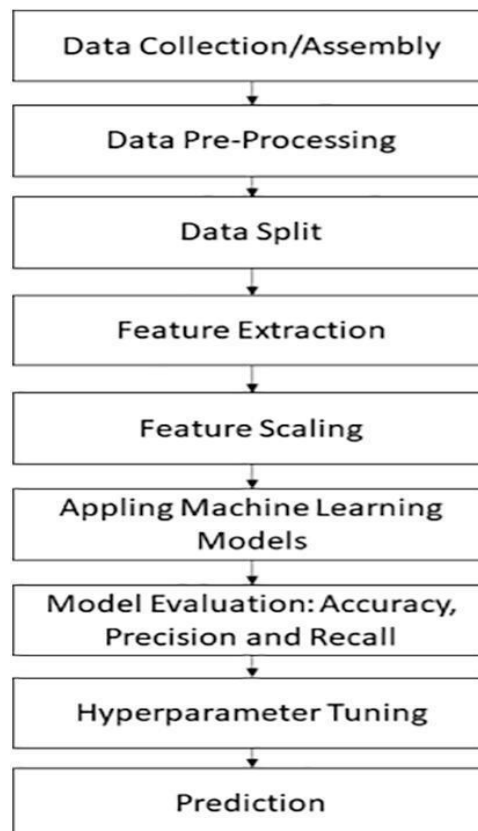


Fig 3. Implementation of the system

5.1.1: Data Collection

We will collect the Data from Different Roads from the Cities. Data It may be in the form of Image or Video.

5.1.2: Data Pre-Processing

Data Processing is the task of converting data from a given form to a much more usable and desired form i.e., making it more meaningful and informative. It checks the collected Data for Better output.

5.1.3: Data Split

Data can be Divided into Two forms train and Test split. with a two-part split, one part is used to evaluate or test the data and the other to train the model.

5.1.4: Feature Extraction

Feature type of dimensionality reduction Here a large number of pixels of the image are efficiently represented in such a way that interesting parts of the image are captured effectively. so it helps detect the potholes.

5.1.5: Applying the Machine Learning Model

A Machine Learning Model can be Applied for the data collected. The type of the Data decides which model should be applied. To get Better Accuracy and Desired Result.

5.1.6: Model Evaluation

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

5.1.7: Hyperparameter Tuning

In machine learning, hyperparameter optimization is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters are learned.

Step 8: Prediction

Prediction in machine learning refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome.

5.2 Block Diagram

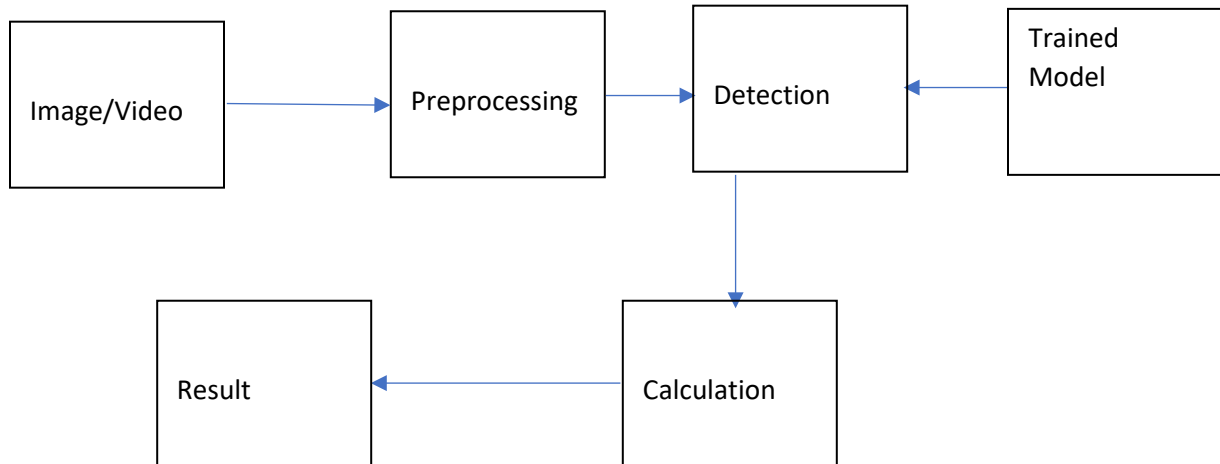


Fig 4. Block Diagram

The figure Fig.4 depicts the project's overall flow. In the beginning, the user will supply an image or video as input, and the model will preprocess the data before detecting potholes using the data set's Training model. It will calculate damage percentage and provide the results to the user based on the video/image clip given.

- **Image Pre-processing**

Picture scaling and image segmentation are the first two steps in the process. Pothole detecting quality is unaffected by image size. As a result, we first reduce the size of an image by times to reduce the number of computations.

- **Region of Interest Extraction**

The area where potholes can be located, i.e., asphalt pavement, is the area of interest. The B component of the RGB color space from the previous phase, which is a black-and-white image, is used as the input for this stage (0,255). To begin, we create a new matrix containing the pixel coordinates of all white pixels in a picture. Second, we choose a dynamic number of pixels for the first level seed points based on the image's standard deviation ($\sigma/2$). The method for determining the first level seed points. We introduce second-level seed locations to improve the accuracy of asphalt pavement detection. The values

of nearby pixels from the first level are used to create the second level. Based on the positions of these locations, the region of interest is calculated.

- **Pothole Detection**

It detects the pothole in current frame using Pre-Trained Model. If Pothole is in frame, it gives 1 otherwise its value is 0.

5.3 Algorithm Design

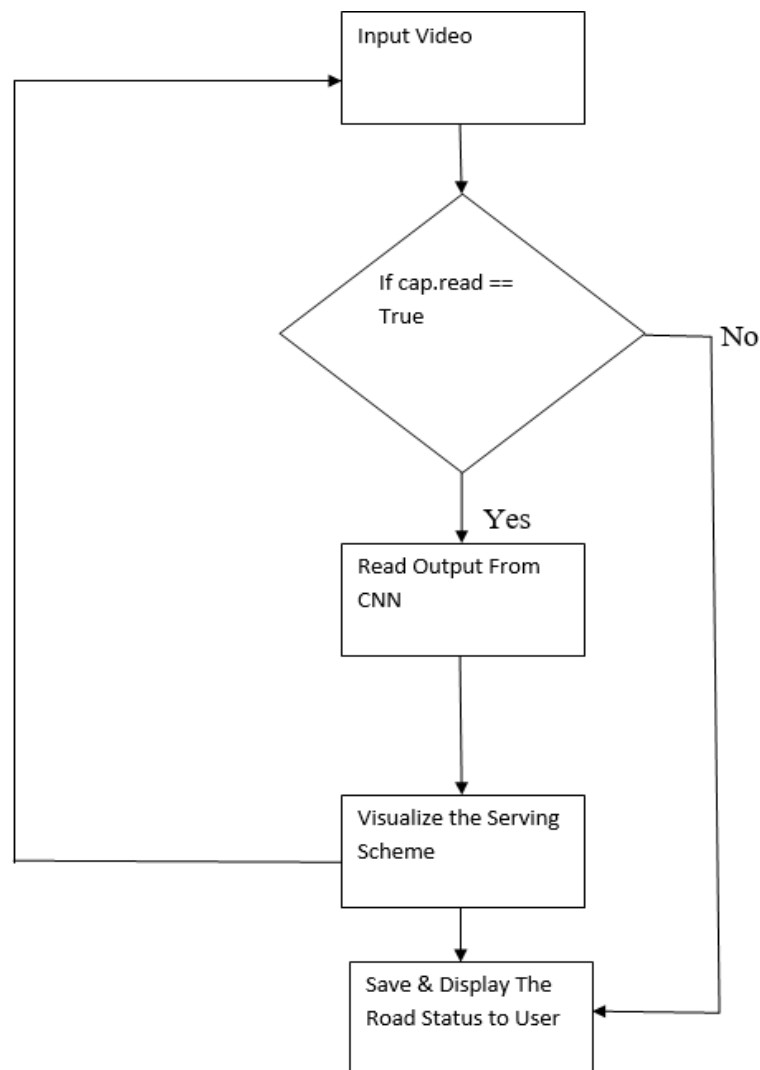


Fig 5. Flow chart

5.4 Algorithm: An algorithm is a set of well-defined instructions to solve a particular problem. It takes a set of input and produces a desired output.

Step 1: Read the input video

Step 2: If cap.read is true go to step 3 else step 5

Step 3: Read the output from CNN

Step 4: View the serving scheme

Step 5: Display the Road Status to user

5.4 Source code:

Code plays a very important role in software development; Without coding code, we can't create a web application. Even if you're going to assemble, glue it all together, and transmit findings from one module to the next. Great code enables us to have both a reliable system that doesn't break and a high-performance system that responds promptly to our demands. We can produce greater products by writing outstanding code, giving us a competitive advantage in the marketplace.

1) Home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <script src='https://kit.fontawesome.com/a076d05399.js' crossorigin='anonymous'></script>
</head>

<body style="background-image: url({{ url_for('static', filename='bg.jpg') }})">
<div class="jumbotron text-center">
  <h1>Home Page</h1>
</div>
<br>
<div class="container">
  <button <a href="/home">hooome</a> </button>
  <button <a href="/logout">Logout</a> </button>
</div>
<br>
<br>
<div class="jumbotron">
<form action="/predict" method="POST" enctype="multipart/form-data">

  <div class="text-center">
    <label for="sname">Street Name:</label>
    <input type="text" name="sname" id="sname"><br><br>
    <input type="file" name="file"><br><br>
    <input type="submit" value="Predict">
  </div>
</form>
</div>
</body>
</html>
```

2) Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <script src='https://kit.fontawesome.com/a076d05399.js' crossorigin='anonymous'></script>
</head>
<body>
<div class="jumbotron text-center">
  <h1>Login Page</h1>
</div>
<p class="text-center">{{msg}}</p>
<div class="jumbotron">
<form action="/login" method="POST">
  <div class="row">
    <div class="col-7 text-right">
      <label for="user">User ID:</label>
      <input type="text" id="user" name="user">
    </div>
  </div>
  <div class="row">
    <div class="col-7 text-right">
      <label for="psw">Password:</label>
      <input type="password" id="psw" name="psw">
    </div>
  </div>
  <div class="text-center">
    <input type="submit" value="Login"></div>
</form>
  <p class="text-center">click here to <a href="/register">Register</a> </p>
</div>
</body>
</html>
```


3) Register.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Enhanced</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  <script src='https://kit.fontawesome.com/a076d05399.js' crossorigin='anonymous'></script>
</head>
<body>
<div class="jumbotron text-center">
  <h1>Pothole Detection</h1>
</div>
<div class="container">
<button ><a href="/home">Home</a> </button>
  <button><a href="/log">Login</a> </button>
</div>
<div class="container">
  <table class="table table-bordered">
    <thead>
      <tr>
        <th>Sl No.</th>
        <th>Road Name</th>
        <th>Damage in %</th>
        <th>Sample Images</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>
          {% for row in rows %}
          <td>{{row[0]}}</td>
          <td>{{row[1]}}</td>
          <td>{{row[2]}}</td>
          <td><a href="/view/{{row[1]}}">View</a> </td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
</div>
</body>
</html>

```

4) Video_predictor.py

```
import cv2
import time
import imutils
import numpy as np
from sklearn.metrics import pairwise
import time
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import model_from_json
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
import glob

global loadedModel
size = 300

# resize the frame to required dimensions and predict
def predict_pothole(currentFrame):
    currentFrame = cv2.resize(currentFrame, (size, size))
    currentFrame = currentFrame.reshape(1, size, size, 1).astype('float')
    currentFrame = currentFrame / 255
    prob = loadedModel.predict(currentFrame)
    classes_x = np.argmax(prob, axis=1)
    print(classes_x[0])
    if classes_x[0] == 1:
        print("Pothole Detected")
    else:
        print("Road is Plane")
    max_prob = classes_x[0]
    if (max_prob > .90):
        return (loadedModel.predict(currentFrame) > 0.5).astype("int32"), max_prob
    #return loadedModel.predict_classes(currentFrame), max_prob
    return "none",

# main function
if __name__ == '__main__':

    loadedModel = load_model('full_model.h5')

    camera = cv2.VideoCapture("video_pot6.mp4")

    show_pred = False
    # loop until interrupted
    while (True):
```

```
(grabbed, frame) = camera.read()
if grabbed == True:
    frame = imutils.resize(frame, width=700)
    frame = cv2.flip(frame, 1)

    clone = frame.copy()

    (height, width) = frame.shape[:2]

    grayClone = cv2.cvtColor(clone, cv2.COLOR_BGR2GRAY)

    pothole, prob = predict_pothole(grayClone)

    keypress_toshow = cv2.waitKey(1)

    if (keypress_toshow == ord("e")):
        show_pred = not show_pred

    if True:
        cv2.putText(clone, str(pothole) + ' ' + str(prob * 100) + '%', (30, 30), cv2.FONT_HERSHEY_DUPLEX, 1,
                      (0, 255, 0), 1)

    cv2.imshow("GrayClone", grayClone)

    cv2.imshow("Video Feed", clone)

    keypress = cv2.waitKey(1) & 0xFF

    if (keypress == ord("q")):
        break

camera.release()

cv2.destroyAllWindows()
```

5) image_predictor.py

```
import cv2
import time
import imutils
import numpy as np
from sklearn.metrics import pairwise
import time
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import model_from_json
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
import glob

global loadedModel
size = 300

# resize the frame to required dimensions and predict
def predict_pothole(currentFrame):
    currentFrame = cv2.resize(currentFrame, (size, size))
    currentFrame = currentFrame.reshape(1, size, size, 1).astype('float')
    currentFrame = currentFrame / 255
    prob = loadedModel.predict(currentFrame)
    classes_x = np.argmax(prob, axis=1)
    print(classes_x[0])
    if classes_x[0] == 1:
        print("Pothole Detected")
    else:
        print("Road is Plane")
    max_prob = classes_x[0]
    if (max_prob > .90):
        return (loadedModel.predict(currentFrame) > 0.5).astype("int32"), max_prob
    #return loadedModel.predict_classes(currentFrame), max_prob
    return "none", 0

# main function
if __name__ == '__main__':
    loadedModel = load_model('full_model.h5')
    #camera = cv2.VideoCapture(0)
    show_pred = False
    # loop until interrupted
    #(grabbed, frame) = camera.read()
    frame = cv2.imread('4.jpg', cv2.IMREAD_COLOR)
    frame = imutils.resize(frame, width=700)
    frame = cv2.flip(frame, 1)

    clone = frame.copy()
```

```
(height, width) = frame.shape[:2]

grayClone = cv2.cvtColor(clone, cv2.COLOR_BGR2GRAY)

pothole, prob = predict_pothole(grayClone)

keypress_toshow = cv2.waitKey(1)

if (keypress_toshow == ord("e")):
    show_pred = not show_pred

if (show_pred):
    cv2.putText(clone, str(pothole) + ' ' + str(prob * 100) + '%', (30, 30), cv2.FONT_HERSHEY_DUPLEX, 1,
                (0, 255, 0), 1)

cv2.imshow("GrayClone", grayClone)

cv2.imshow("Video Feed", clone)

keypress = cv2.waitKey(1) & 0xFF

cv2.waitKey(0)

cv2.destroyAllWindows()
```

6) app.py

```
import sqlite3
import cv2
import os
import time
import imutils
import numpy as np
from sklearn.metrics import pairwise
import time
import tensorflow as tf
from keras.datasets import mnist
from keras.models import Sequential
from keras.models import model_from_json
from keras.models import load_model
from keras.layers import Dense
from keras.layers import Dropout
from keras.utils import np_utils
import glob

from flask import *

global loadedModel
loadedModel = load_model('full_model.h5')
path = "/static/Plain"
size = 300

app = Flask(__name__)
# resize the frame to required dimensions and predict
def predict_pothole(currentFrame):
    currentFrame = cv2.resize(currentFrame, (size, size))
    currentFrame = currentFrame.reshape(1, size, size, 1).astype('float')
    currentFrame = currentFrame / 255
    prob = loadedModel.predict(currentFrame)
    classes_x = np.argmax(prob, axis=1)
    print(classes_x[0])
    if classes_x[0] == 1:
        print("Pothole Detected")

    else:
        print("Road is Plane")
    max_prob = classes_x[0]
    if (max_prob > .90):
        return (loadedModel.predict(currentFrame) > 0.5).astype("int32"), max_prob
        #return loadedModel.predict_classes(currentFrame), max_prob
    return "none", 0

@app.route('/')
def index():
    return redirect(url_for('home'))
```

```
        return redirect(url_for('home'))
@app.route('/log')
def log():
    return render_template('login.html')

@app.route('/register')
def register():
    return render_template('register.html')

@app.route('/sign_up',methods=['POST'])
def sign_up():
    if request.method == 'POST':
        name = request.form['name']
        mob = request.form['mob']
        mail = request.form['mail']
        psw = request.form['psw']
        conn = sqlite3.connect('data.db')
        conn.execute("INSERT INTO accounts(name,mob,mail,psw) VALUES(?,?,?,?)",(name,mob,mail,psw))
        conn.commit()
        msg = "Registered Successfully"
        return render_template("register.html",msg=msg)
@app.route('/login',methods=['POST'])
def login():
    if request.method == 'POST':
        mail = request.form['user']
        psw = request.form['psw']
        conn = sqlite3.connect('data.db')
        cur = conn.execute("SELECT * FROM accounts WHERE mail=? AND psw=?", (mail,psw))
        rows = cur.fetchone()
        if rows == None:
            msg = "Invalid Id & Password"
            return render_template('login.html',msg=msg)
        else:
            return render_template('home.html')
@app.route('/logout')
def logout():
    return render_template('login.html')
@app.route('/predict',methods=['POST'])
def predict():
    if request.method == "POST":
        sname=request.form['sname']
        file= request.files['file']
        f=file.filename
        if not os.path.isdir('static/'+str(sname)):
            os.mkdir('static/'+sname)
```

```
camera = cv2.VideoCapture(f)

show_pred = False
n1=n2=n3=0

# loop until interrupted
while (True):

    (grabbed, frame) = camera.read()
    if grabbed == True:
        frame = imutils.resize(frame, width=700)
        # frame = cv2.flip(frame, 0)

        clone = frame.copy()

        (height, width) = frame.shape[:2]

        grayClone = cv2.cvtColor(clone, cv2.COLOR_BGR2GRAY)

        pothole, prob = predict_pothole(grayClone)
        n1 = n1 + 1
        print(pothole, prob, n1)

        if prob > 0:
            n2 = n2 + 1
            if n2>n3+40:
                n3=n2
                cv2.imwrite(os.path.join('static/'+str(sname),str(n2)+'.jpg'),frame)
            if(n1==100 or n1==200 or n1==300 or n1==400):
                cv2.imwrite(os.path.join('static/' + str(sname), str(n1) + '.jpg'), frame)

        keypress_toshow = cv2.waitKey(1)

        if (keypress_toshow == ord("e")):
            show_pred = not show_pred

        if True:
            cv2.putText(clone, "Pothole" + ' ' + str(prob * 100) + '%', (30, 30), cv2.FONT_HERSHEY_DUPLEX, 1,
                          (0, 255, 0), 2)

            cv2.imshow("GrayClone", grayClone)

            cv2.imshow("Video Feed", clone)

        else:
            damage = (n2 / n1) * 100
            damage=round(damage, 2)
            break
```



```
cv2.destroyAllWindows()
conn = sqlite3.connect("data.db")
cur = conn.execute("SELECT * FROM s_road WHERE s_name=?", (sname,))
row=cur.fetchone()
if row==None:
    conn.execute("INSERT INTO s_road(s_name,accuracy)VALUES(?,?)", (sname,damage))
    conn.commit()
    conn = sqlite3.connect("data.db")
    rows = conn.execute("SELECT * FROM s_road")
    return render_template("result.html",rows=rows)
else:
    conn.execute("UPDATE s_road SET accuracy=? WHERE s_name=?", (damage,sname))
    conn.commit()
    conn = sqlite3.connect("data.db")
    rows = conn.execute("SELECT * FROM s_road")
    return render_template("result.html",rows=rows)

@app.route('/view/<a>')
def view(a):
    folder='static/'+str(a)
    rows=[]
    for count, filename in enumerate(os.listdir(folder)):
        rows.append(filename)
    return render_template("view.html",rows=rows,a=a)
@app.route('/home')
def home():
    conn = sqlite3.connect("data.db")
    rows = conn.execute("SELECT * FROM s_road")
    return render_template("result.html",rows=rows)
# main function
if __name__ == '__main__':
    app.run(host="0.0.0.0")
```

CHAPTER 6

Software Testing

The system execution is to find the errors that can be defined as testing. It can also be defined as the process that defines, isolates, and subjects to rectification of defects, and so that the customer satisfaction is reached at last with the assurance of the system is free from defects. The important component of software testing is quality assurance and it represents the SRS, designing, coding, and implementation of the system proposed.

6.1 Levels of testing

Test Planning:

The test plan is the document that gives the information regarding the procedure that is to be followed in performing various tastings on the whole application. This document involves the scope and objectives of the testing, areas that are to be tested and areas that should not be tested, scheduling of resources available, the area that needs to be automated, and various tools that are used for testing.

Test Development:

Test development involves the development of test cases and their procedural preparation i.e., description of the developed test cases.

6.2 Types of testing

Unit testing:

As the name itself says, the testing is made on small units of the system. A part of the system is considered a unit and its testing are done. If as an example, the login page is considered; the user or the administrator can enter into their respective home pages only after giving the valid username and password. This part of validating a system, by considering Login as a unit can be said to a unit testing.

Integration testing:

This part of testing deals with the testing procedure. It involves, the testing of various integrations of several units. It checks whether the system is functioning correctly when two or more units are integrated

together. This part of testing gives information about the order of arrangements of various units, integrating modules, systems, sub-systems, and the entire system as a whole.

System testing:

This testing technique deals with the process of testing the system as a whole. At the end of each project, all defects are removed and the interface errors are uncovered to achieve the good functioning of the whole system. This testing technique can be called the final part of the whole testing process.

6.3 Test cases:

Test Cases	Input	Expected output	Actual Output	Conclusion
#Tc001	Video clip with 84.32 % Damage	84.32 % Damage	84.32 % Damage	PASS
#Tc002	Video clip with 00.00 % Damage	00.00 % Damage	00.00 % Damage	PASS
#Tc003	Video clip with 5.91 % Damage	5.91 % Damage	5.91 % Damage	PASS
#Tc004	Video clip with 19.37 % Damage	19.37 % Damage	19.37 % Damage	PASS

CHAPTER - 7

SNAPSHOTS

1) Pothole Detection:



Fig. 6 Pothole Detection

Fig.6 Describes Potholes on the road are seen in the above image. The user's input is what we're looking for That pothole can be absorbed here. Here, 0 denotes no potholes and 1 denotes 100% pothole detection.

2) Non-Pothole Detection:

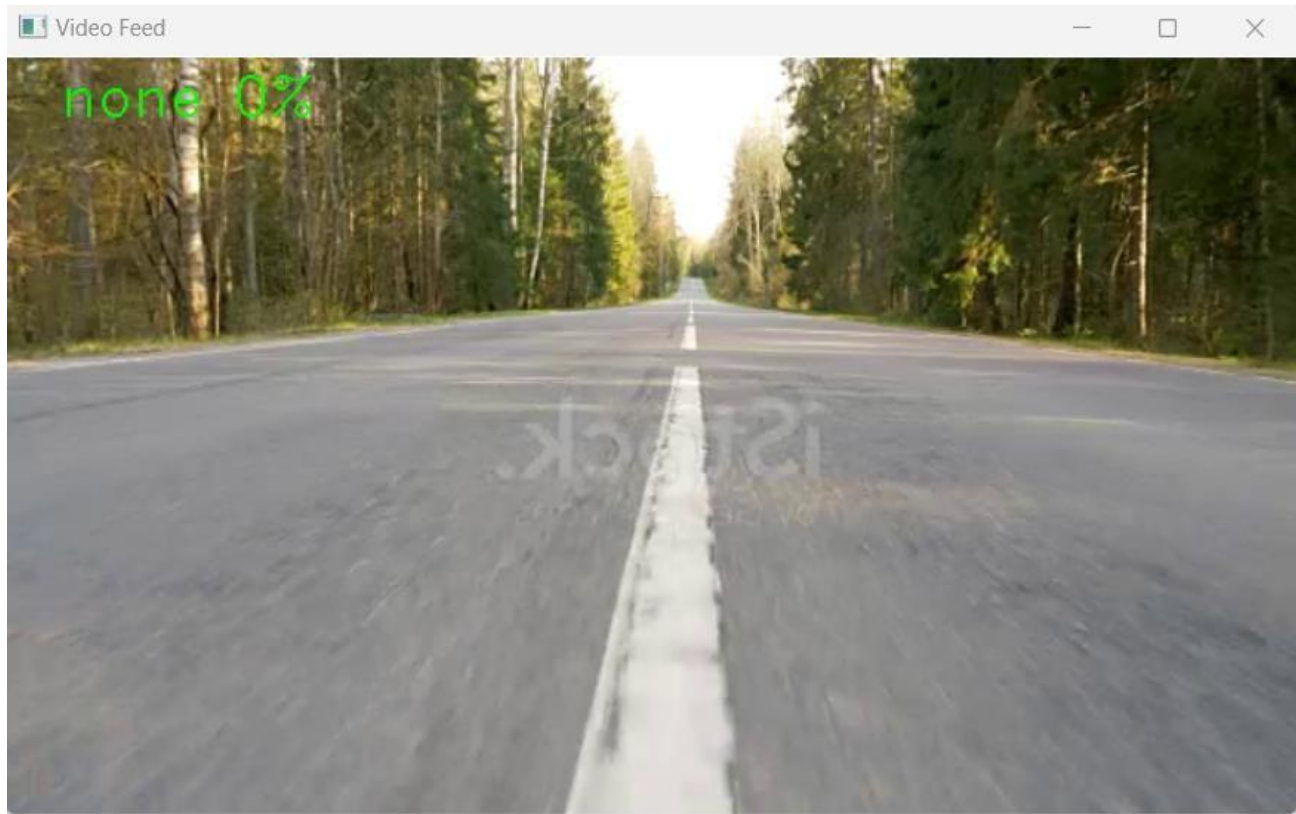


Fig 7. Non-Pothole Detection

Fig.7 Describes There is No Potholes on the road are seen in the above image. The user's input is what we're looking for That Non-pothole can be absorbed here. Here, 0 denotes no potholes.

3) Accuracy Graphs:

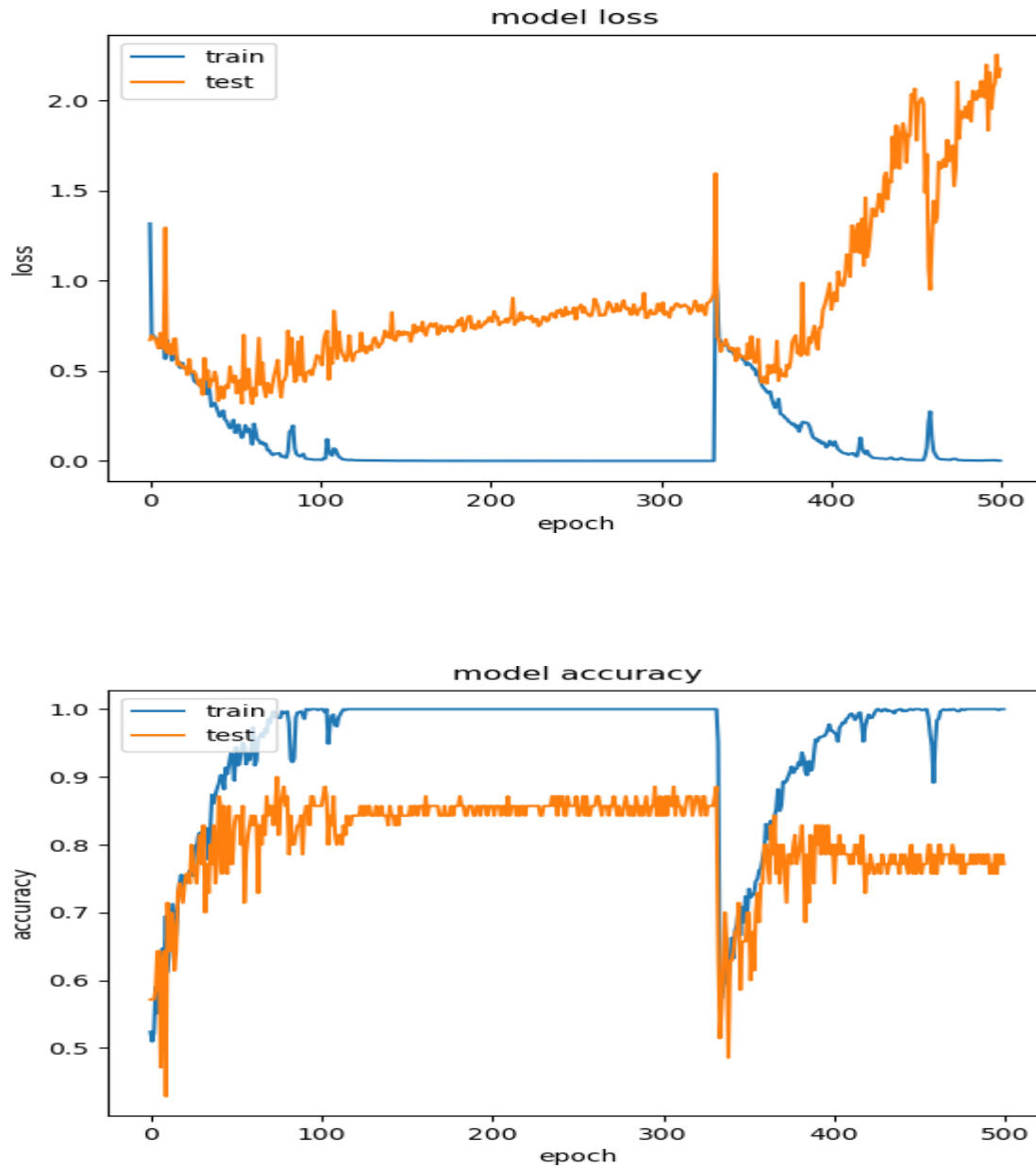


Fig 8. Model Accuracy

Fig.8 Describes The train and test split of the data set are represented in the above graph. Also describe the model's accuracy.

4) Home page:

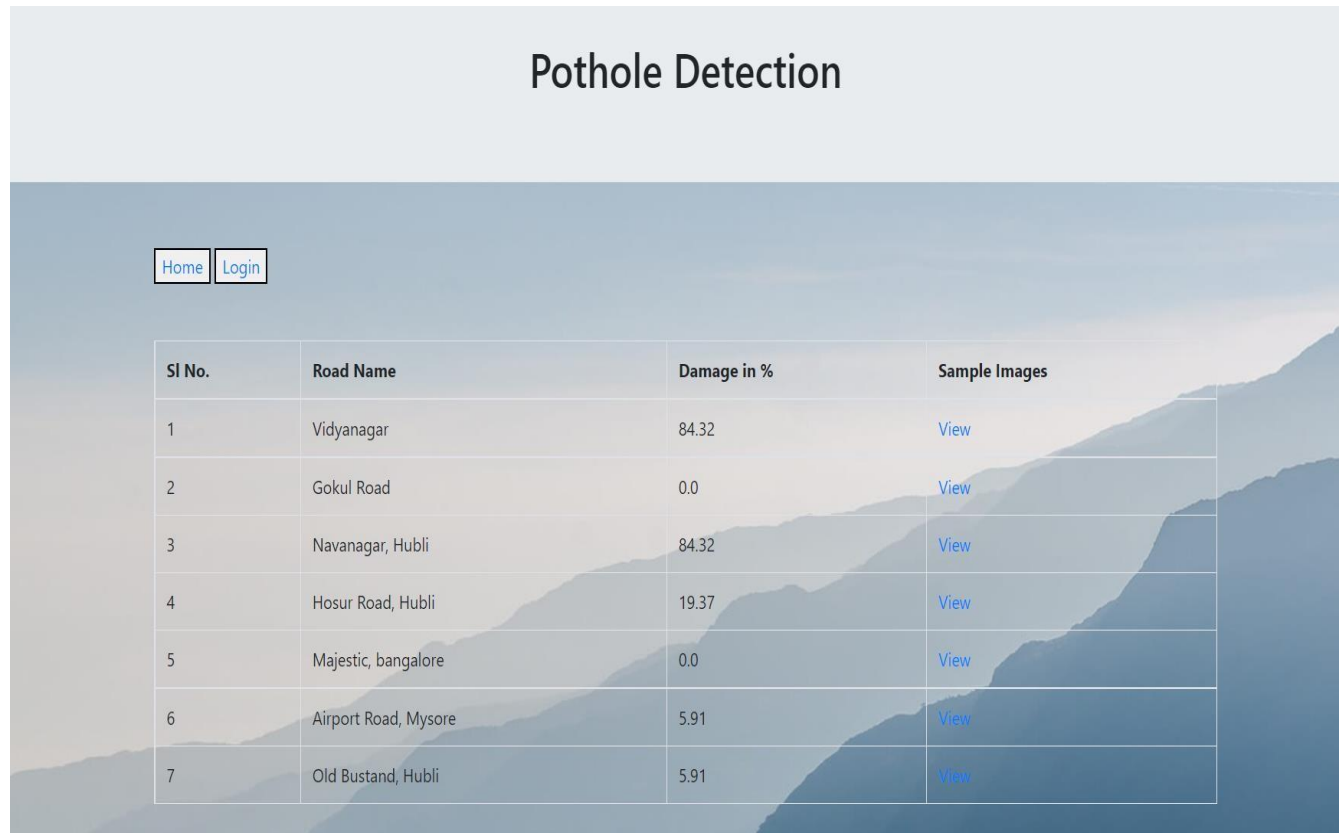
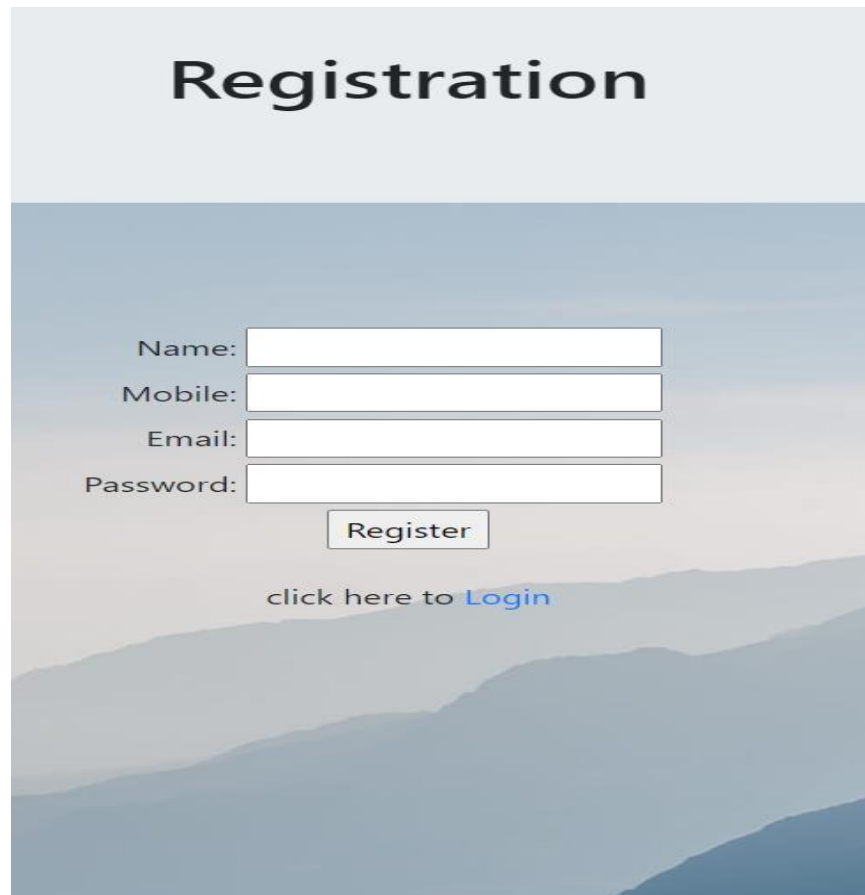


Fig 9. Pothole Detection Dashboard

Fig.9 Depicts that percentage of roads that have been damaged, as well as some images of those roads. That Columns Road Name, Damage in Percentage, and Sample Images can be absorbed. These will serve as the road's summary.

5) Registration Page:

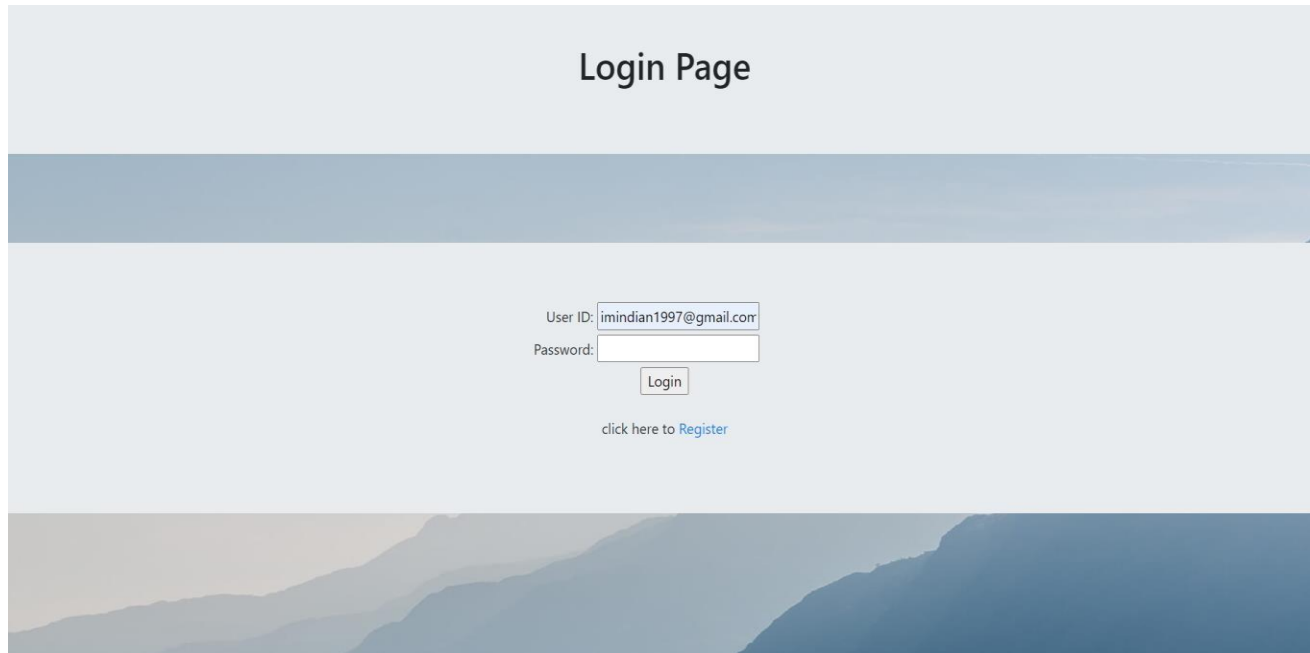


The image shows a web page titled "Registration" with a light blue header. Below the header, there is a registration form with four input fields: "Name:", "Mobile:", "Email:", and "Password:". Each field is a white rectangle with a thin border. Below the "Password:" field is a "Register" button with a light blue background and a thin border. Below the button, there is a link that says "click here to [Login](#)". The background of the page is a blurred image of mountains.

Fig 10. Registration Page

Fig.10 Depicts that users must first register before using the Portal. It requires the user's name, mobile number, email address, and a strong password.

6) Login Page:

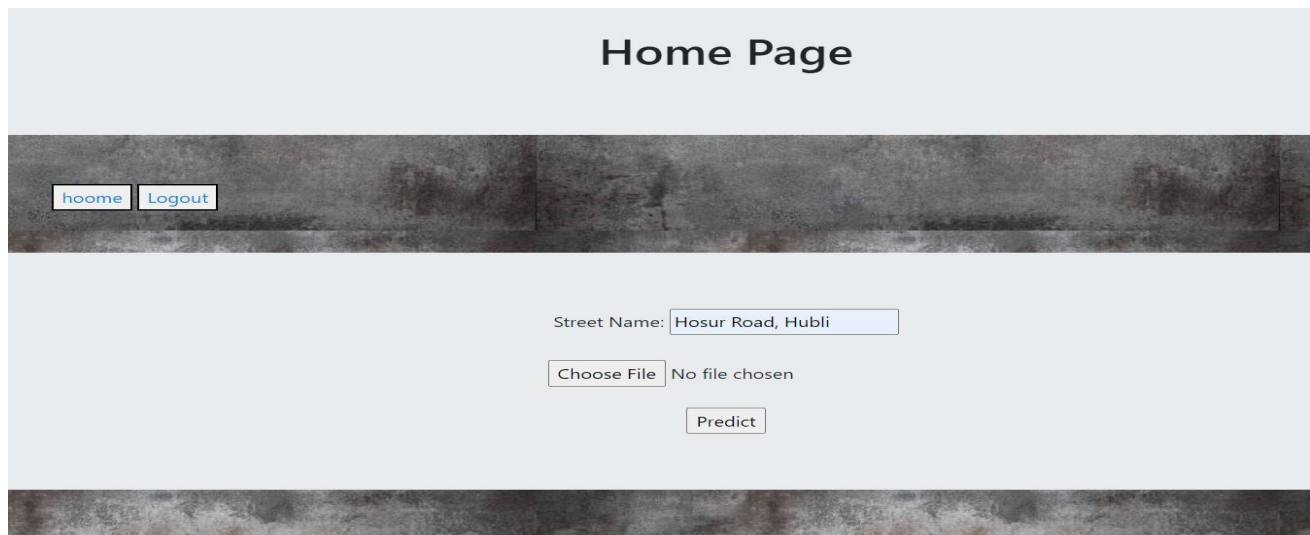


The screenshot shows a web page titled "Login Page" with a light blue header. Below the header is a dark blue horizontal bar. The main content area is light blue and contains a login form. The form has two input fields: "User ID:" with the value "imindian1997@gmail.com" and "Password:" with an empty field. Below the password field is a "Login" button. At the bottom of the form, there is a link that says "click here to [Register](#)". The background of the page features a faint image of mountains.

Fig 11. Login Page

Fig.11 Indicates that After completing the registration process, users may log in to the portal by entering their respective User-ID and Password.

7) Input Page:



The screenshot shows a web page titled "Home Page" with a light blue header. Below the header is a dark blue horizontal bar. The main content area is light blue and contains a home page form. At the top left, there are two buttons: "home" and "Logout". Below these buttons is a form with a "Street Name:" label and a text input field containing "Hosur Road, Hubli". Below the street name field is a "Choose File" button and a text label "No file chosen". At the bottom of the form is a "Predict" button. The background of the page features a faint image of mountains.

Fig 12. Home Page

Fig.12 Depicts that the user may enter the street name along with a video clip, and it will begin anticipating potholes based on the information provided by the user.

CHAPTER-8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

This project explains and implements a real-time pothole detection and traffic monitoring system, and it was able to use Smartphone sensors to tackle a worldwide challenge, apply Machine Learning to a real-world problem, and create a scalable, dependable system via crowdsourcing. The transportation difficulty, as described in the introduction that led to this study, has been proven to be considerable. As technology progresses and penetration increases, such difficulties can be addressed with readily available solutions, as demonstrated by the development of this system. The sample size, however, is a limitation of this study. More practical findings will be retrieved and studied in the future in order to adopt the proposed strategy everywhere. Furthermore, due to the mobile device's limited battery capacity, the topic of saving computation power can be investigated. To lower the frequency of accelerometer data detection while maintaining high pothole identification accuracy, a green pothole detection approach is required.

8.2 Future Scope

In future we propose to do more experiments with variety of the scenario. As of now we have 83% Accuracy of the model. In future we will Increase the Accuracy of the model by retraining the model And applying the hyperparameters. And we also Apply the Different Machine Learning Algorithm To Improve the Accuracy.

- Maps can be included in the android application for graphically showing the pothole location.
- Along with the existing voice notification “There is pothole nearby go slow”, Distance of the pothole can also be voice notified.
- More accurate and complex algorithm can be used to find the distance between two latitude and longitude.

REFERENCES

[1] Riga, LV 1006, Latvia, Artis Mednis, Girts strazdins, Reinholds zviedris, Georgijs Kanonirs. "Real time pothole detection using android smartphones with accelerometers".

Date of Publish: 29 June 2011, DOI:10.1109/DCOSS.2011.5982206

[Online] Available at: <https://ieeexplore.ieee.org/document/5982206>

[2] Jin Lin ,Yayu Liu, "Potholes detection based on SVM in the pavement distress image".

Date of Publish: 12 August 2010, DOI: 10.1109/DCABES.2010.115

[Online] Available at: <https://ieeexplore.ieee.org/document/5571563>

[3] Pereira, V., Tamura, S., Hayamizu, S., & Fukai, H. "A Deep Learning-Based Approach for Road Pothole Detection".

Date of Publish: 02 August 2018, DOI: 10.1109/SOLI.2018.8476795

[Online] Available at: <https://ieeexplore.ieee.org/document/8476795>

[4] Seung-Ki Ryu, Taehyeong kim, "Image-Based Pothole Detection System for ITS Service and Road Management System".

Date of Publish: 16 Sep 2015, DOI: 10.1155/2015/968361

[Online] Available at: <https://www.hindawi.com/journals/mpe/2015/968361/>

[5] S. Nienaber, M.J. Booysen, R.S. Kroon, "A comparison of Low-cost Monocular Vision Techniques for Pothole Distance Estimation",

Date of Publish: 07 December 2015, DOI: 10.1109/SSCI.2015.69

[Online] Available at: <https://ieeexplore.ieee.org/document/7376642>

APPENDIX

TECHNOLOGY USED

HTML: Web pages and web applications are created using Hypertext Markup Language, which is a standard markup language. It forms a triad with Cascading Style Sheets (CSS) and JavaScript as cornerstone technologies of the World Wide Web. Images and other objects, including interactive forms, may be embedded into the rendered page using HTML constructs.

CSS: Cascading Style Sheets portrays how HTML components are to be shown on screen, paper or in other media. CSS spares a parcel of work. It can control the format of different web pages all at once. CSS may be a foundation technology of the World Wide Web, nearby HTML and JavaScript.

JAVASCRIPT: JavaScript, commonly referred to as JS, is an interpreted, high-level programming language. The World Wide Web was built on JavaScript as one of its three core technologies. It empowers intuitively Web pages and in this way is an basic portion of web applications. The endless larger part of websites utilizes it and all the major web browsers have a committed JavaScript engine to execute it.

Bootstrap: An open-source CSS framework that targets responsive, mobile-first web development, Bootstrap is free and open-source. This includes CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface elements. Bootstrap is a framework that allows you to design websites faster and easier.

PYTHON: The Python language is an interpreted, objective, high-level language with dynamic semantics. The easy-to-learn syntax of Python emphasizes readability and therefore reduces the cost of program maintenance. Modularity and code reuse are encouraged with Python's modules and packages.

SQLite3: SQLite is a relational database management system (RDBMS) compiled into a C library. Contrary to most other database management systems, SQLite is not a client-server engine. SQLite is embedded inside the application. For local/client storage in application programs such as web browsers, SQLite is a popular choice.

PYCHARM: PyCharm is a computer programming integrated development environment (IDE) that focuses on the Python programming language. PyCharm is available in Windows, Mac OS X, and Linux versions. There is a Community Edition that is provided under the Apache License, a Professional Edition with more features that is released under a subscription-funded proprietary license, and an educational edition.

CHROME: Google Chrome is a cross-platform web browser that was created by the company Google. It was first published for Microsoft Windows in 2008, and it was made using free software components from Apple Web Kit and Mozilla Firefox. It was eventually ported to Linux, macOS, iOS, and Android, where it is now the operating system's default browser.

DIAGRAMS.NET: diagrams.net (formerly draw.io) is a cross-platform graph sketching application written in HTML5 and JavaScript that is free and open source. Flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams can all be created using its interface. diagrams.net is a cross-browser web app as well as an offline desktop application for Linux, macOS, and Windows. The Electron framework is used to create its offline application.