A Real Time Research Project/ Societal Related Project Report

On

Real-time Pothole Detection System: Mitigating Road Hazards through Image Processing

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

Bachelor of Technology

In

Department of Computer Science and Engineering

By

Sai Priya	22241A0567
Anushka Kumar	22241A0570
K. Sannitha	22241A0586
Surugu Akshaya	22241A05C1
Sneha Padhi	22241A05C2

Under the Esteemed guidance of

Dr. P. Chandra Sekhar Reddy
Professor

Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Bachupally, Kukatpally, Hyderabad, Telangana, India,500090

2023-2024

GOKARAJU RANGARAJU

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

CERTIFICATE

This is to certify that the Real Time Research Project/ Societal Related Project entitled "**Real-time Pothole Detection System: Mitigating Road Hazards through Image Processing**" is submitted by **Sai Priya (22241A0567), Anushka Kumar (22241A0570), K.Sannitha (22241A0586), Surugu Akshaya (22241A05C1), Sneha Padhi (22241A05C2),** in partial fulfillment of the award of degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2023-2024.**

INTERNAL GUIDE	HEAD OF THE DEPARTMENT
Dr. P. Chandra Sekhar Reddy	**Dr. B. SANKARA BABU**
Professor	**Professor and HOD**

ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **Dr. P. Chandra Sekhar Reddy**, **Professor**, Department of CSE for his/her support in the completion of our project report. We wish to express our honest and sincere thanks to **Dr. Y. Krishna Bhargavi and Ms. V. Jyothi** for coordinating in conducting the project reviews, **Dr. B. Sankara Babu, HOD,** Department of CSE for providing resources, and to the principal **Dr. J. Praveen** for providing the facilities to complete our Real Time Research Project/ Societal Related Project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Sai Priya	**(22241A0567)**
Anushka Kumar	**(22241A0570)**
K. Sannitha	**(22241A0586)**
Surugu Akshaya	**(22241A05C1)**
Sneha Padhi	**(22241A05C2)**

DECLARATION

We hereby declare that the Real Time Research Project/ Societal Related Project entitled "**Real-time Pothole Detection System: Mitigating Road Hazards through Image Processing**" is the work done during the period from **2023-2024** and is submitted in the fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad).** The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Sai Priya	**(22241A0567)**
Anushka Kumar	**(22241A0570)**
K. Sannitha	**(22241A0586)**
Surugu Akshaya	**(22241A05C1)**
Sneha Padhi	**(22241A05C2)**

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This project presents the development of a pothole detection system that uses dashcam footage and employs Convolutional Neural Network (CNN) models along with computer vision techniques to alert drivers when there is an upcoming pothole. Potholes are still a problem on roads around the world. These road defects are a serious threat to road safety, leading to vehicle breakdowns and accidents. Current pothole detection methods are manual and time-consuming, often hampering remediation strategies. The proposed system aims to enhance road safety by providing real-time warnings to drivers, enabling them to take appropriate actions on time.

The system utilizes a dashcam mounted on the vehicle to continuously capture video footage of the road ahead. This footage is processed using a CNN model trained to accurately detect and classify potholes in various road conditions and lighting scenarios. The detection is done using computer vision algorithms which employs the trained CNN model. The driver is then alerted through auditory signals to allow immediate response.

The developed pothole detection application offers a practical solution for reducing vehicle damage and accidents caused by potholes, contributing to overall road safety and maintenance efficiency. This system can be integrated into modern vehicles, forming a crucial component of future transportation infrastructure.

1. INTRODUCTION

1.1 Image processing:

Image processing involves the manipulation and analysis of images to enhance their quality or extract useful information. It is a crucial step in preparing raw dashcam footage for analysis by the CNN model. In this project, image processing techniques are applied to improve the clarity and visibility of the footage, which includes noise reduction, contrast enhancement, and edge detection. These techniques help in highlighting features relevant to pothole detection, such as the shape and texture of road surfaces. By preprocessing the images, the system ensures that the CNN model receives high-quality inputs, which improves the accuracy and reliability of pothole detection.

1.2 Deep Learning:

Deep learning is a subset of machine learning that focuses on training artificial neural networks with multiple layers, known as deep neural networks, to perform complex tasks. These networks are capable of learning and extracting intricate patterns and representations from large amounts of data. In this project, deep learning is employed to train models that can accurately detect and classify potholes from video footage. The ability of deep learning models to improve performance as more data becomes available makes them particularly suitable for this application.

1.3 CNN:

Convolutional neural networks (CNNs) are a specific type of deep learning model designed for processing structured grid data, such as images. CNNs consist of layers that apply convolution operations, allowing the model to automatically and adaptively learn spatial hierarchies of features from the input images. These features include edges, textures, and shapes, which are crucial for identifying objects within the images. In the context of this project, CNNs are used to analyze dashcam footage, effectively detecting and classifying potholes with high accuracy. Their ability to capture spatial dependencies and reduce the complexity of image processing makes CNNs ideal for this task.

1.4 Computer Vision:

Computer vision is a field of artificial intelligence that enables computers to interpret and understand visual information from the world. It encompasses a variety of techniques and algorithms to process, analyze, and extract meaningful information from images and videos. In this pothole detection system, computer vision techniques are employed to preprocess the dashcam footage, enhance image quality, and identify relevant features.

1.5 System Architecture:

A pothole detecting system's system architecture is shown in the image.
When a car drives over a pothole, the detecting procedure is triggered, which is how the system is started.

- Pothole Dataset: The photos and annotations utilized to train the machine learning model for pothole identification are likely included in this collection of pre-existing data.
- Annotating the dataset is the first stage in identifying attributes that the machine learning model can use to locate potholes in new photos.
- Convolutional neural networks, or CNNs, are a particular class of machine learning model well-suited to image recognition applications. The CNN model in this system probably learned how to recognize potholes in photos by being trained on the pothole dataset.
- Vehicle Stationary Optics: This part probably refers to the LiDAR sensor or camera that is installed on the car and uses it to take pictures or scans of the road surface while it is moving.
- Pothole Image: This is the picture that the car's sensor took and that the system uses to look for potholes.
- Comparison of Detection Performance: In this step, the pothole detection model's performance is assessed. It is possible to use metrics such as Mean Average Precision (MAP) and F1-score to gauge the model's accuracy in identifying potholes.
- Level of Severity Assessment: The system evaluates the severity of a pothole upon detection, potentially taking its size or depth into account.
- Driver Alerted: This denotes that the driver has received a notification from the system, probably via the dashboard or an in-car alert system, concerning the pothole.

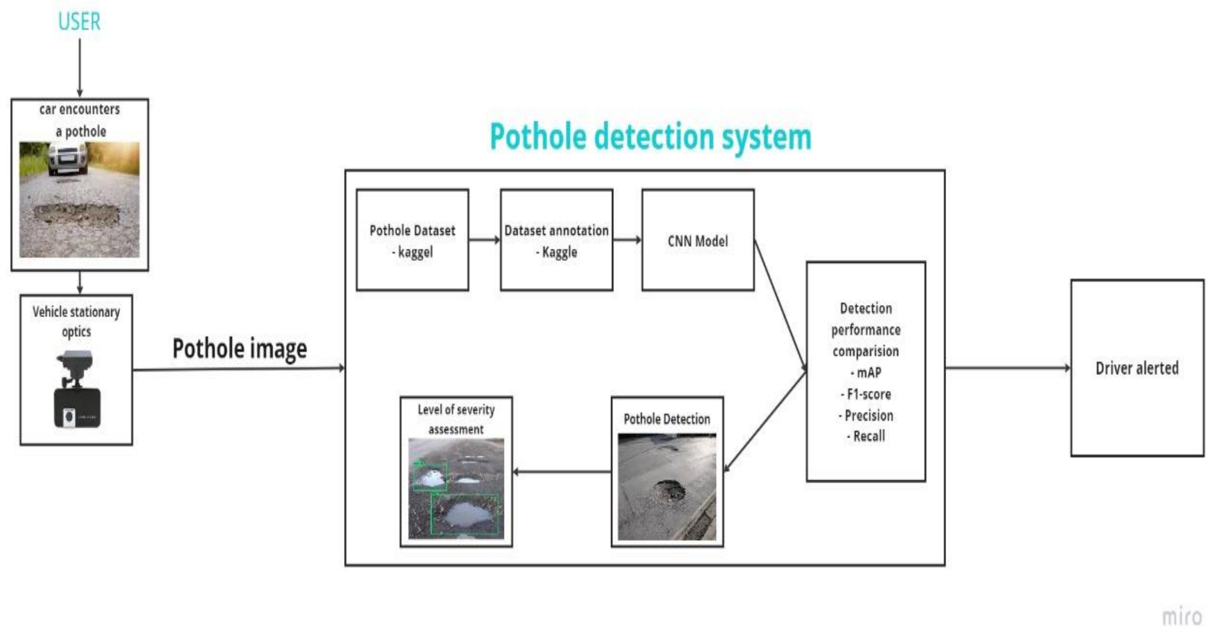

USER
car encounters a pothole
Vehicle stationary optics
Pothole image
Pothole detection system
Pothole Dataset - kaggel
Dataset annotation - Kaggle
CNN Model
Detection performance comparision - mAP - F1-score - Precision - Recall
Pothole Detection
Level of severity assessment
Driver alerted
miro

Fig 1.5 System Architecture

2. SYSTEM REQUIREMENTS

2.1 Software requirements:

- **Programming language:** Python

  Python provides robust libraries like OpenCV, Pillow, scikit-image, and SimpleITK for image processing. They perform as fundamental tools for a variety of computer vision applications, providing a range of functionalities such as filtering, segmentation, and feature extraction. This project requires Python 3.12.

  In the OpenCV community, Python is especially well-liked because of its readability and simplicity, which complement the intricacy of computer vision jobs. With comparatively few lines of code, developers can construct complex image processing features because to Python's succinct syntax. Additionally, Python makes prototyping faster, which is advantageous in settings for both academic and commercial research.

- **Integrated Development Environment (IDE):** The following Integrated Development Environments (IDEs) are recommended for working with this project:
  - **Visual Studio Code (VSCode):** A lightweight yet powerful IDE known for its versatility and extensive plugin ecosystem. It supports Python development with features like IntelliSense, debugging capabilities, and integrated version control.
  - **Jupyter Notebook:** An interactive computing environment that enables creation and sharing of documents containing live code, equations, visualizations, and narrative text. It is well-suited for data exploration, prototyping, and collaboration.
  - **PyCharm:** A comprehensive IDE specifically designed for Python development. PyCharm offers advanced features such as intelligent code assistance, debugging, and integration with various Python frameworks and libraries.

- **Python and Machine Learning libraries:**
  - **NumPy:** Essential for efficient array operations and mathematical computations, crucial for image data processing.
  - **OpenCV:** Provides extensive computer vision functionalities such as image

processing, feature detection, and object recognition.

  - **TensorFlow:** Deep learning framework for building and training neural networks, used here for developing the CNN model.
  - **Matplotlib:** Plotting library used for visualizing dashcam footage, training metrics, and detected potholes.
  - **SciPy:** Scientific computing library offering tools for optimization, interpolation, and statistical analysis.
  - **Keras:** High-level neural networks API simplifying the construction and training of deep learning models, built on TensorFlow.
  - **Pygame:** Multimedia library utilized for creating interactive audiovisual alerts to notify drivers of detected potholes.
  - **Scikit-learn:** Used for evaluating and reporting performance metrics such as precision, recall, F1-score, and confusion matrix in machine learning models.
  - **Pillow:** A library used for opening, manipulating, and saving various image file formats, offering extensive image processing capabilities.
  - **Tkinter:** A library for creating graphical user interfaces (GUIs) in Python, providing a range of widgets for building desktop applications.

- **Development OS:** Compatible with Windows, macOS, or Linux distributions.

2.2 Hardware Requirements:

- **Processor**: A multi-core processor (e.g., Intel Core i5 or higher) capable of handling intensive computations during training and real-time inference of the CNN model.
- **RAM**: At least 8 GB of RAM to efficiently handle data processing and model training. More RAM (e.g., 16 GB or higher) is beneficial for handling larger datasets and complex computations.
- **Hard Disk**: Adequate storage space (SSD recommended) for storing datasets, trained models, and intermediate data during experimentation and deployment.
- **Dashcam**: A compatible dashcam with high-resolution video capture capability (e.g., 1080p or higher) and reliable performance in various lighting and weather conditions. The dashcam provides real-time footage for pothole detection.
- **Audio Output**: Speakers or audio output device for alerting the driver to detected

potholes using auditory signals, enhancing user interaction and safety.

- **Connectivity**: A stable Internet Connection is required to download all the dependencies to run the project.

2.3 Datasets

The dataset used in this project was sourced from Kaggle and comprises over 700 images, featuring both potholes and normal road conditions. This diverse collection includes a variety of road environments, lighting conditions, and weather scenarios, ensuring comprehensive coverage of real-world driving conditions. The images were meticulously labeled to distinguish between pothole-containing and normal road scenes, facilitating the effective training of the CNN model. By using this dataset, the model was able to learn and recognize the distinguishing features of potholes, such as irregular shapes and textures, while also understanding the characteristics of smooth, intact road surfaces. This robust dataset was essential in achieving high accuracy and reliability in the pothole detection system.

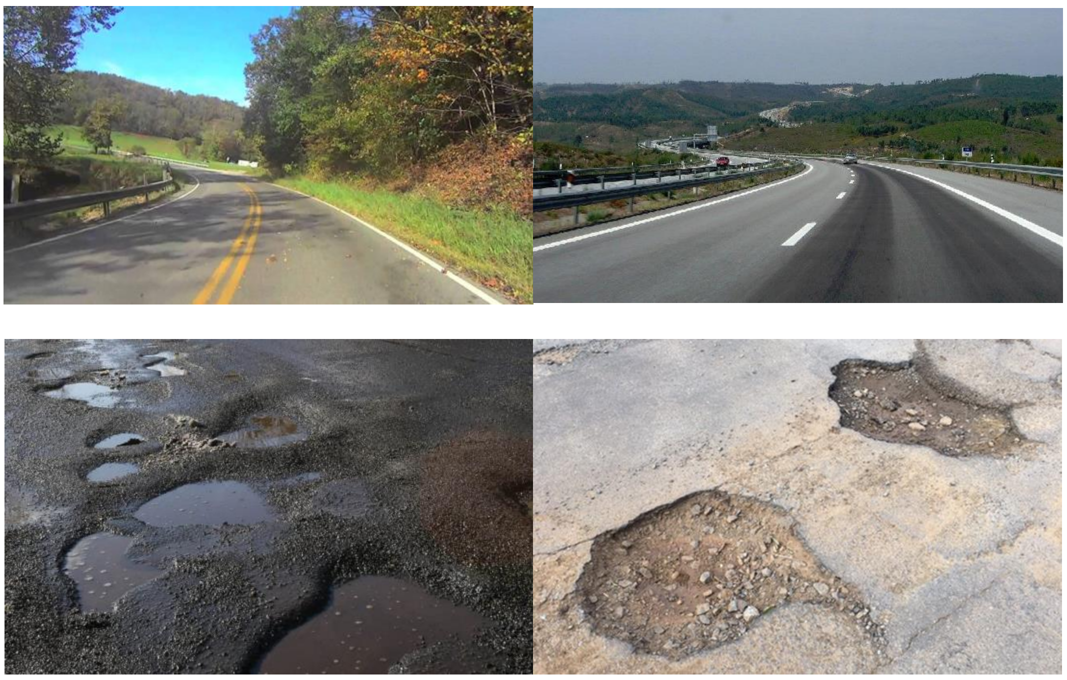


Figure 2.3 Dataset

3. LITERATURE SURVEY

1) **Title:** Pothole Detection System using deep learning

   Publishing year: 2021

   Authors: Samir Berjaoui, Anas Al-Shaghouri, Rami Alkhatib

   Observations: Potholes were successfully identified by all three methods (YOLOv3, YOLOv4, and SDD TensorFlow). Whereas YOLOv4 attained the fastest processing speed at a comparable accuracy, YOLOv3 provided a high recall and precision accuracy while offering a speedy 20 frames per second. According to the paper, processing speed and model complexity—which could have an impact on user-friendliness—can be traded off.

   Shortcomings: This study's scope is restricted by its focus on 2D images and the absence of training data details. To enhance its applicability, the analysis should encompass miss rates, weather effects, and a comprehensive evaluation of the trade-offs between accuracy and speed across various models.

2) **Title:** Real-time machine learning based approach for Pothole detection

   Publishing year: 2021

   Authors: Oche Alexander Egaji, Gareth Evans, Mark Graham Griffiths, Gregory Islas

   Observations: This research leverages machine learning for real-time pothole detection, a promising approach with practical benefits. The study explores a range of techniques, including Naive Bayes and Support Vector Machines, and evaluates their effectiveness using reliable metrics like accuracy and F-score. By combining real-time focus, diverse algorithmic analysis, and robust assessment methods, the study establishes a solid foundation for understanding the potential of machine learning in pothole identification, paving the way for practical applications.

   Shortcomings: While the study explores various machine learning methods for real-time pothole detection, it would be beneficial to explore more innovative approaches. Deep learning techniques, particularly in image recognition tasks, may yield improved outcomes. Moreover, although the study recognizes challenges like data imbalance and real-world variations, it fails to offer solutions, which may limit the system's practical applicability and versatility in real-world scenarios.

3) **Title:** Detection and segmentation of concrete pothole based on Image Processing

Publishing year: 2020

Authors: Mingxing Gao, Xu Wang, Shoulin Zhu, Peng Guan

Observations: The study investigates image processing methods for segmenting and detecting potholes in concrete pavements. This method shows potential for automated pothole detection. The study analyzes pothole characteristics using a variety of techniques, including texture filters and machine learning (SVM), and assesses performance using measures like recall and F1-score. This offers insightful information on how well image processing works to detect potholes.

Shortcomings: Despite the high accuracy of the research, the procedure is dependent on uncontrollable elements. Real-world applicability may be limited by the reliance on the image quality of a particular camera and the susceptibility to outside influences like illumination. Moreover, compared to more automated methods, the dependence on human feature selection and processing time may be disadvantages. Finally, the report makes no mention of how the approach performs in difficult situations such as sandy soil-filled potholes.

4) **Title:** Pothole detection system design with proximity sensor to provide motorcycle with warning system and increase road safety.

Publishing year: 2020

Authors: Hadistian Muhammad Hanif, Zener Sukra Lie, Winda Astuti, Sofyan Tan

Observations: This project suggests use proximity sensors and microcontrollers to create a motorbike pothole detecting system. Although the system has the benefit of real-time warning, its inability to identify deep potholes could get due to its dependence on short-range proximity sensors. Furthermore, the requirement for regulated testing settings and the possibility of speed measurement mistakes point to the need for additional development for wider use. The scalability and cost-effectiveness of the system are called into question due to its complexity, which includes several sensors.

Shortcomings: First off, deeper potholes may be unnoticed by the sensors due to their restricted detecting range. Second, its applicability in real-world scenarios with diverse road conditions is called into question due to its dependence on controlled circumstances during testing. Thirdly, the precision of pothole alerts may be impacted by possible mistakes in speed measurement. Lastly, the system's complexity with its many sensors points to problems with scalability and affordability for broad use.

5) **Title:** Application of various YOLO Models for Computer Vision-Based Real-Time Pothole Detection

   Publishing year: 2021

   Authors: Sung-Sik Park, Van-Than Tran, Dong-Eun Lee

   Observations: This paper investigates YOLO models (YOLOv4, Y4-tiny, YOLOv5) for computer vision-based real-time pothole identification. Although the technique shows promise in terms of metrics when applied to a well-labeled dataset, it may not be as effective at spotting potholes in certain lighting or weather scenarios. Practical applications should also take into account the dependence on human image labeling for training and the possible requirement for powerful hardware for real-time implementation.

   Shortcomings: While this YOLO-based pothole detection system is promising, there are still several issues. Under different situations, the system may have trouble with potholes while maintaining decent accuracy. Furthermore, the need for manually labeled training data and technology that can need a lot of resources could prevent real-world adoption. In conclusion, more research is necessary to see whether the model can be applied to different types of roads or ecosystems.

6) **Title:** PotSpot: Participatory sensing-based monitoring system for pothole detection using deep learning

   Publishing year: 2021

   Authors: Susmita Patra, Asif Iqbal Middya, Sarbani Roy

   Observations: This study suggests a pothole detection system that makes use of deep learning and human input. The algorithm performs admirably on a real-world dataset, but its efficacy depends on user input and high-quality data. Aspects that need more research include the possibility of unbalanced data sets and long-term user involvement. It would also need further research to scale this method to cover more geographic areas.

   Shortcomings: Deep learning and user-contributed data are used by this pothole detection system to achieve excellent accuracy. However, a few drawbacks need to be taken into account. First off, the system depends on regular user participation, which over time may prove to be unreliable. Second, factors like intermittent image capture or human mistake may give rise to problems with data quality. Finally, there are issues that must be resolved when expanding this strategy to include large geographic areas.

7) **Title:** PotNet: Pothole detection for autonomous vehicle system using Convolutional Neural Network.

   Publishing year: 2020

   Authors: Deepak Kumar Dewangan, Satya Prakash Sahu

   Observations: This study investigates the use of a convolutional neural network (CNN) with a single camera for pothole identification in autonomous vehicles. The excellent accuracy attained is encouraging. Nevertheless, more thought must be given to the system's reliance on camera calibration and its limitations with regard to depth perception from a single camera. Potential advancements in pothole detecting capabilities could come from adding more sensors or developing depth estimate techniques in the future.

   Shortcomings: CNN and one camera are used in this pothole detecting system for autonomous vehicles (PotNet). Despite its excellent accuracy, the system is not without flaws. First of all, it depends on accurate camera calibration, which can be a difficult and continuous procedure. Second, measuring pothole depth accurately is difficult with a single camera—a vital piece of information for autonomous vehicles. These drawbacks imply that more advancements in depth estimation or the addition of more sensors might be required for a more reliable solution.

8) **Title:** DeepBus: Machine Learning based Real Time Pothole Detection system for smart transportation using IoT

   Publishing year: 2020

   Authors: Kashish Bansal, Kashish Mittal, Gautam Ahuja, Ashima Singh, Sukhpal Singh Gill

   Observations: This study investigates the use of smartphone sensors and machine learning for real-time pothole identification. The method appears to be promising in locating potholes, yet there are certain drawbacks. First, because of sensor limitations, it may incorrectly identify bumps or uneven roads. Second, the technology lacks the ability to distinguish between potholes that are deep and shallow, which could be crucial for road upkeep. Lastly, using cellphones constantly in the real world raises questions about their dependability and data security.

Shortcomings: Initially, the precision of the system may be compromised, resulting in incorrect classifications over uneven terrain or as a result of speed changes. Second, its ability to prioritize road repairs is hampered by its inability to differentiate between the severity of potholes. Finally, in long-term, real-world deployments, the reliance on cellphones poses questions regarding data security and durability.

9) **Title:** A Real-time Pothole Detection Based on Deep Learning Approach

Publishing year: 2021

Authors: Yeoh Keng Yik, Nurul Ezaila Alias, Yusmeeraz Yusof and Suhaila Isaak

Observations: This study suggests utilizing a webcam and a deep learning algorithm to create a real-time pothole detecting system. Even while the method is very accurate at identifying potholes, it may overlook a sizable percentage (poor recall), necessitating additional research. Its usefulness in varying road conditions is called into question due to its dependence on a small dataset and possible calibration requirement. Lastly, the usage of a webcam raises the possibility of processing power constraints, which may affect real-time functionality.

Shortcomings: For real-time application, this pothole detecting system makes use of a webcam and a deep learning algorithm. Although the system is accurate in detecting potholes, it may overlook a significant number of them (poor recall). There are certain restrictions that must be met. Initially, the system may not be able to generalize to various road conditions because it was trained on a limited dataset. It gets more complicated because calibration is required. Finally, the usage of a webcam raises the possibility of processing power constraints, which could have an impact on real-time functionality.

10) **Title of research paper:** Intelligent Real Time Pothole Detection and Warning System for Automobile Applications Based on IoT Technology

Publishing year: 2020

Authors: M.S. Kamalesh, Bharatiraja Chokkalingam, Jeevanantham Arumugam, Gomathy Sengottaiyan, Shanmugavadivel Subramani, Mansoor Ahmad Shah

Observations: This project investigates a pothole detection and warning system for cars based on the Internet of Things. There are clear limits to the system, even if it provides data storage and real-time capabilities. First off, expensive component costs may result from the requirement for a strong CPU for image processing. Second, the system's application in rural locations may be limited due to its reliance on GSM/GPRS network connectivity. Further research is also necessary due to possible gadget malfunctions and a documented restriction in accurate detection at speeds higher than 30 km/h.

Shortcomings: Although there are many drawbacks, this study on an Internet of Things pothole detecting system for cars provides real-time warnings and data storage. First, the expense of image processing could increase due to the requirement for a powerful CPU. Second, its application in places with weak cellular reception may be restricted due to its reliance on GSM/GPRS network connectivity. Lastly, more research is needed to rule out device malfunctions and limits with regard to precise detection over 30 km/h.

11) **Title of research paper:** Pothole Detection and Dimension Estimation System using Deep Learning (YOLO) and Image Processing

Publishing year: 2020

Authors: Pranjal A. Chitale, Kaustubh Y. Kekre, Hrishikesh R. Shenai, Ruhina Karani, Jay P. Gala

Observations: This work investigates the use of deep learning and image processing in a pothole detection and dimension estimation system. The size of the custom dataset, which includes both domestic and international highways, may have an impact on the system's performance. Its dependence on a particular camera angle and the possibility of accuracy fluctuations brought on by external circumstances also call for additional research to ensure real-world robustness.

Shortcomings: This study uses deep learning and image processing to develop a pothole detection system with dimension estimation. The technology has limitations despite integrating a proprietary dataset with a variety of road conditions. First, its inability to be generalized to new contexts may be due to the small size of the dataset. Second, the system's reliance on a particular camera angle can limit how useful it is in practical scenarios. Ultimately, more research is necessary to ensure reliable pothole identification because there is a chance that environmental variables like weather or lighting can affect performance.

12) **Title of research paper:** YOLOv8-Based Visual Detection of Road Hazards: Potholes, Sewer Covers, and Manholes

Publishing year: 2023

Authors: Om Khare, Shubham Gandhi, Aditya Rahalkar, Sunil Mane

Observations: This project addresses using YOLOv8 for visually detecting road dangers like potholes, sewage covers, and manholes. While achieving strong results on a specific dataset, the system's generalizability requires further examination. To assure performance in real-world circumstances, including a greater variety of training data and testing the system under various variables like illumination, weather, and road surfaces would be advantageous.

Shortcomings: This research analyzes YOLOv8 for detecting road dangers visually. While producing positive outcomes, the system's reliance on a single dataset offers limits. For real-world robustness, fixing these limitations is vital. First, including a wider set of training data would improve generalizability. Secondly, testing the system under varied lighting, weather, and road conditions is important to confirm its performance in different contexts.

13) **App Name:** Pothole Radar

Details: Pothole Radar is an app that uses GPS location to find potholes marked by you and others nearby. It then plots them on a radar screen for you to see and gives an audible warning as they approach.

Reviews: no reviews, 4.2 rating by 32 users, 1000+downloads

Disadvantage: not user friendly, bad UI, only uses GPS to find potholes marked by users priorly not real-time detection.

14) **App Name:** Pothole Patrol

Details: allows road users to report potholes in their area by taking a picture of the pothole, record the location and notify authorities

Reviews: 2.3 rating by 64 users, 10k+ downloads

Users find the app's map tricky and unsafe to use while driving, reporting potholes does not result in fixes, and registration is problematic.

Disadvantage: Slow loading times, lack of response to issues, and excessive data collection.

15) **App Name:** RoadBounce Pothole Guard

Details: aims to improve your driving experience by detecting potholes using your phone's sensors and alerting you about them. the phone is mounted on the car.

Reviews: 100+ downloads, No reviews yet.

Disadvantages: relying solely on phone sensors might not detect all potholes, especially smaller ones.

4. PROPOSED MODEL, MODULES DESCRIPTION, AND UML DIAGRAMS

4.1 Proposed Model

The proposed model for the pothole detection system is a convolutional neural network (CNN) designed to accurately identify potholes in real-time using dashcam footage. The model architecture balances complexity and efficiency to ensure detection accuracy while being capable of running on typical onboard hardware.

4.1.1 Model Architecture

1) **Input Layer:**
    - Accepts input images resized to a fixed dimension, such as 64x64 pixels, to standardize processing.
    - Preprocessing includes normalization to scale pixel values and enhance model performance.
2) **Convolutional Layers:**
    - Multiple convolutional layers are used to extract features from the input images.
    - Each convolutional layer applies filters to detect edges, textures, and shapes relevant to identifying potholes.
    - Layers use ReLU (Rectified Linear Unit) activation functions to introduce non-linearity.
3) **Pooling Layers:**
    - Max-pooling layers follow convolutional layers to downsample the feature maps, reducing the spatial dimensions and computational load.
    - Pooling helps in capturing dominant features while making the model more invariant to translations and distortions.
4) **Dropout Layers:**
    - Dropout layers are used to prevent overfitting by randomly setting a fraction of input units to zero during training.
    - This encourages the model to learn more robust features.

5) **Fully Connected Layers:**
    - One or more fully connected (dense) layers aggregate the features extracted by convolutional layers.
    - These layers perform the final classification based on the learned features.
6) **Output Layer:**
    - The output layer uses a softmax activation function to provide probabilities for each class (pothole or normal road).
    - The class with the highest probability is selected as the model's prediction.

4.1.2 Training

- **Dataset:** The model is trained on the Kaggle dataset, which includes over 700 labeled images of potholes and normal road conditions.
- **Loss Function:** Categorical cross-entropy is used as the loss function to measure the performance of the classification model.
- **Optimizer:** The Adam optimizer is employed for efficient gradient descent during training.
- **Validation:** A portion of the dataset is reserved for validation to monitor the model's performance and prevent overfitting.

4.1.3 Real-Time Processing

- **Inference:** During real-time operation, the trained model processes frames from the dashcam footage, detecting and classifying potholes.
- **Alert System**: Upon detecting a pothole, the system uses computer vision techniques and triggers an alert to warn the driver.

4.1.4 Performance Metrics

The performance of the model was evaluated using a test set consisting of 32 images. The following metrics were obtained:

- **Accuracy:** 0.9375 (93.75%)

  This metric indicates the percentage of correctly predicted instances out of the total number of instances evaluated. In this case, the model achieved an accuracy of 93.75%, suggesting that it correctly predicted the class labels for a large portion of the test dataset.

- **Loss:** 1.2515

  Loss represents the error of the model during training and evaluation. A lower loss value indicates a better fit of the model to the data.
- **Classification Report:** The classification report provides insights into how well the model performs for each class:

  Precision: This metric measures the accuracy of positive predictions.

  - **Precision for Class 0:** 0.38

    Indicates that when the model predicts Class 0, it is accurate 38% of the time.
  - **Precision for Class 1:** 0.00

    Reflects that the model does not correctly predict any instances of Class 1.

  Recall: This metric gauges the model's ability to correctly identify positive instances.

  - **Recall for Class 0:** 1.00

    Demonstrates that the model successfully identifies all instances of Class 0.
  - **Recall for Class 1:** 0.00

    Indicates that the model fails to identify any instances of Class 1.

  F1-Score: The harmonic mean of precision and recall, providing a balanced assessment of a model's performance.

  - **F1-Score for Class 0:** 0.55

    Reflects a moderate balance between precision and recall for Class 0.
  - **F1-Score for Class 1:** 0.00

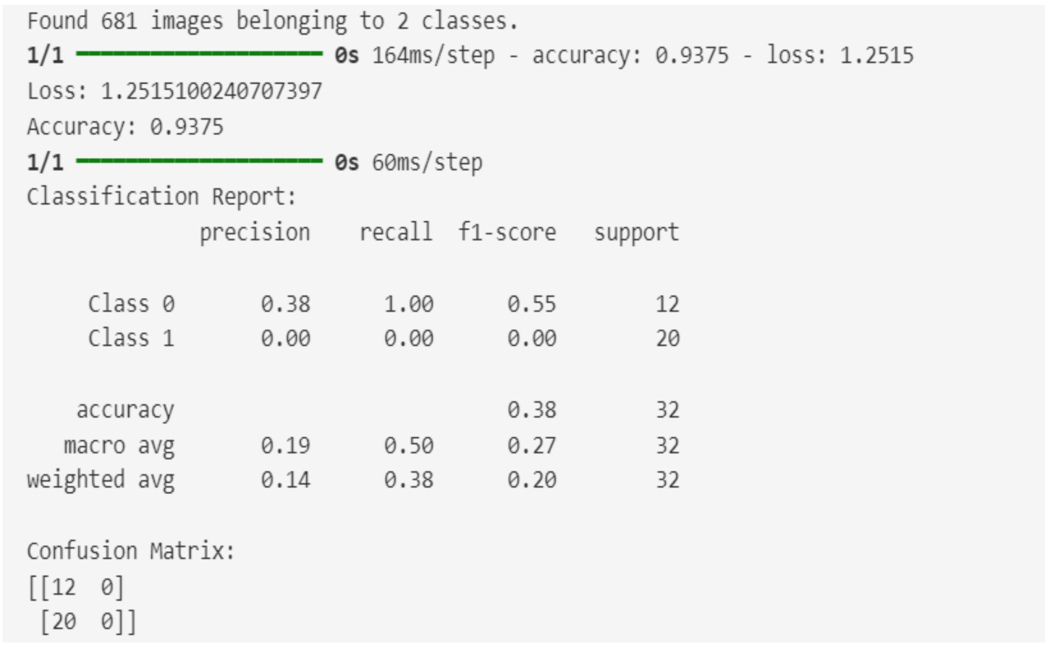
    Indicates the model's inability to correctly classify any instances of Class 1.

- **Confusion Matrix:**

  The confusion matrix summarizes the performance of the model by showing the number of correct and incorrect predictions for each class.

  - True Positives (TP): The model correctly predicted 12 instances of Class 0.
  - False Positives (FP): The model incorrectly predicted 0 instances of Class 1 as Class 0.

- False Negatives (FN): The model incorrectly predicted 20 instances of Class 1 as Class 0.
- True Negatives (TN): The model correctly predicted 0 instances of Class 1.



```
Found 681 images belonging to 2 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 164ms/step - accuracy: 0.9375 - loss: 1.2515
Loss: 1.2515100240707397
Accuracy: 0.9375
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.38      1.00      0.55        12
     Class 1       0.00      0.00      0.00        20

    accuracy                           0.38        32
   macro avg       0.19      0.50      0.27        32
weighted avg       0.14      0.38      0.20        32

Confusion Matrix:
[[12  0]
 [20  0]]
```

Fig 4.1.4 Metrics Output

4.2 Modules

The pothole detection system is composed of several key modules, each responsible for a specific aspect of the functionality.

1) **Data Collection and Preprocessing Module**

- **Purpose**: Captures and preprocesses dashcam footage for training and real-time detection.
- **Functions and Classes:**
  - **os.walk():** Traverses the directory to collect image paths.
  - **cv2.imread():** Reads images from the specified paths.
  - **plt.imshow():** Displays images for verification.
  - **ImageDataGenerator:** Preprocesses images by rescaling, shearing, zooming, and flipping.
  - **Inputs:** Raw video footage and images from the dashcam.
  - **Outputs:** Preprocessed images ready for model training and inference.

2) **Training Module**

- **Purpose:** Builds, trains, and saves the CNN model for pothole detection.
- **Functions and Classes:**
  - **tf.keras.models.Sequential()**: Constructs the CNN model.
  - **tf.keras.layers:** Adds various layers (Convolutional, Pooling, Dense, Dropout) to the model.
  - **model.compile():** Configures the model for training.
  - **model.fit():** Trains the model on the dataset.
  - **Inputs:** Preprocessed training and validation data.
  - **Outputs:** Trained CNN model saved to disk.

3) **Prediction Module**

- **Purpose:** Loads the trained model and makes predictions on new images.
- **Functions and Classes:**
  - **image.load_img():** Loads and resizes test images.
  - **image.img_to_array():** Converts images to arrays for model input.
  - **model.predict():** Predicts the presence of potholes in the test images.
  - **Inputs:** New test images.
  - **Outputs:** Predictions indicating the presence or absence of potholes.
  - **model.fit():** Trains the model on the dataset.
  - **Inputs:** Preprocessed training and validation data.
  - **Outputs:** Trained CNN model saved to disk.

4) **Real-Time Detection Module**

- **Purpose:** Processes video frames in real-time to detect potholes and alert the driver.
- **Functions and Classes:**
  - **cv2.VideoCapture():** Captures video from the dashcam.
  - **cv2.resize():** Resizes video frames for the model.

    - **np.expand_dims():** Prepares frames for prediction.
    - **model.predict():** Makes real-time predictions on video frames.
    - **pygame.mixer.music.play():** Plays an alert sound when a pothole is detected.
    - **Inputs:** Real-time video frames from the dashcam.
    - **Outputs:** Real-time alerts for the driver and video frames with overlaid detection results.

5) **Visualization and Logging Module**

- **Purpose:** Displays frames with detection results and logs performance metrics.
- **Functions and Classes:**
    - **cv2.putText():** Overlays detection results on video frames.
    - **cv2.imshow():** Displays video frames with detection results.
    - **Inputs:** Video frames with detection results.
    - **Outputs:** Visual display of detection results and logged performance data.

6) **Audio Alert Module**

- **Purpose:** Generates audio alerts when a pothole is detected in real-time.
- **Functions and Classes:**
    - **pygame.mixer.music.load():** Loads the alert sound file.
    - **pygame.mixer.music.play():** Plays the alert sound when a pothole is detected.
    - **Inputs:** Detection results indicating the presence of potholes.
    - **Outputs:** Audio alerts to the driver.

7) **GUI Module**

- **Purpose:** Provides visual feedback on a Tkinter canvas and triggers audio alerts using Pygame when potholes are detected.
- **Functions and Classes:**

- **mainloop():** Runs the Tkinter main event loop to handle GUI events.
- **cv2.VideoCapture():** Opens a video file or device for capturing frames.
- **cap.read():** Reads a frame from the video capture.
- **cv2.resize():** Resizes the frame to a specified size.
- **cv2.cvtColor():** Converts the color space of the frame from BGR to RGB.
- **Image.fromarray():** Creates a PIL Image object from a numpy array
- **load_model():** Loads a pre-trained Keras model
- **pygame.init():** Initializes Pygame modules for audio playback.
- **pygame.mixer.music.load():** Loads an audio file for playback.
- **pygame.mixer.music.play():** Plays the loaded audio file.
- **tk.Tk():** Initializes the main Tkinter window.
- **tk.Canvas():** Creates a canvas widget for displaying video frames.
- **tk.Label():** Creates a label widget for displaying detection results.
- **label.config():** Configures the text displayed in the label widget.
- **np.expand_dims():** Expands the dimensions of a numpy array.
- **root.after():** Schedules a function to be called after a specified amount of time.
- **cap.release():** Releases the video capture object.
- **cv2.destroyAllWindows():** Destroys all OpenCV windows.
- **Inputs:** Video frames from the dashcam feed.
- **Outputs:** Visual feedback on the Tkinter canvas and audio alerts to the driver.

8) Model Evaluation Module

- **Purpose:** assesses the performance of a trained Keras model using test images from a specified directory. It provides key metrics such as loss, accuracy, classification report, and confusion matrix to evaluate the model's effectiveness.
- **Functions and Classes:**
    - **load_model():** Loads a pre-trained model stored in 'model2.h5'.

- **ImageDataGenerator ():** Generates batches of augmented data for testing.
- **flow_from_directory():** Loads images from the specified directory for testing.
- **model.evaluate():** Computes the loss and accuracy of the model on the test data.
- **model.predict():** Predicts the classes for the test data.
- **classification_report ():** Generates a classification report.
- **confusion_matrix():** Computes a confusion matrix.
- **Inputs:** Directory (test_dir) containing test images, Pre-trained model (model2.h5).
- **Outputs:** Evaluation metrics: Loss and accuracy of the model; Classification report showing precision, recall, and F1-score for each class; Confusion matrix displaying true positive, false positive, true negative, and false negative counts.

4.3.UML Diagrams

4.3.1 Use Case Diagram: -

The "Pothole Detection System" is shown by this use case diagram, which also shows how a user interacts with the system.

- **Actors:** The main player in the Pothole Detection System interaction loop is the User. The User initiates the system's detection process by coming across potholes.
- **Utilization Example:** The user encounters a pothole in the first use case, Encounters Pothole. The system's Detection Module locates the pothole in the Detects Pothole uses case. Upon detection, the Notification Services initiate the Send Notification use case, which entails preparing and sending a notification. Lastly, the use case that Notifies User makes sure the user gets the notification.
- **Components of the System:** The potholes that the user encounters are found by the Detection Module. Notification Services makes sure the user is notified by sending notifications as soon as a pothole is found.

- **Flow:** When a pothole is encountered by the user, the Detection Module is triggered to detect it, initiating the flow. The system then advances to the Send Notification step, where the user receives a notification from the Notification Services, concluding the process and alerting them to the found pothole.

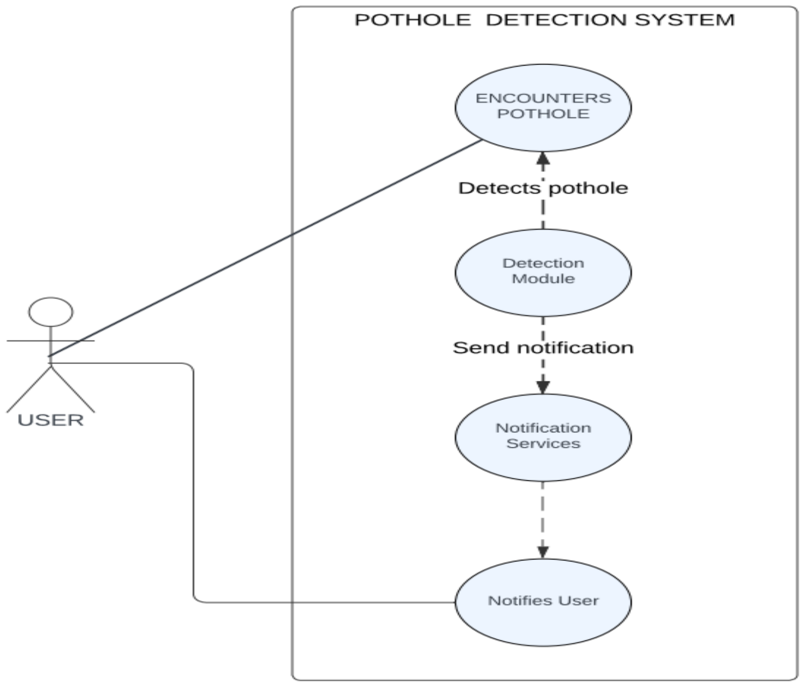


Fig 4.3.1 Use Case Diagram

4.3.2 Class Diagram: -

- User: The User class keeps track of encounters using encounterPothole(location) and has userId and location attributes.
- Pothole Identification Device: Using handlePotholeEncounter(location) and notify (userId, location), the Pothole Detection System synchronizes detection and alerts.
- Pothole Detector: Using analyzePotholeSeverity(image), the Pothole Analyzer assesses severity.
- Manager of Datasets: Pothole data is added and stored by the Dataset Manager using addPotholeData(image, location, severity).
- Picture Capture: Using capturePotholeImage(), the Image Capture class takes pictures.
- Assessor of Performance: Using evaluatePerformance(detectionResults), the performance evaluator rates performance.

- Notification Services: SendNotification(userID, location) is the notification format used by the Notification Services. Phone-specific notifications are implemented through Phone Notification Services.
- Stationary Sensors and Optics: GetSensorData() is used by Vehicle Stationary Optics to retrieve picture data. Sensor uses getSensorData() to retrieve sensor data.
- Module for Detection: Using detectPothole(sensorData), the Detection Module finds potholes.
- Interaction Flow: encounterPothole(location) sets off the system when a user comes across a pothole. In the Pothole Detection System, handlePotholeEncounter(location) processes it using detectPothole(sensorData). After taking a picture and determining its level of severity, Notification Services alerts the user. The Dataset Manager stores data and assesses system performance.

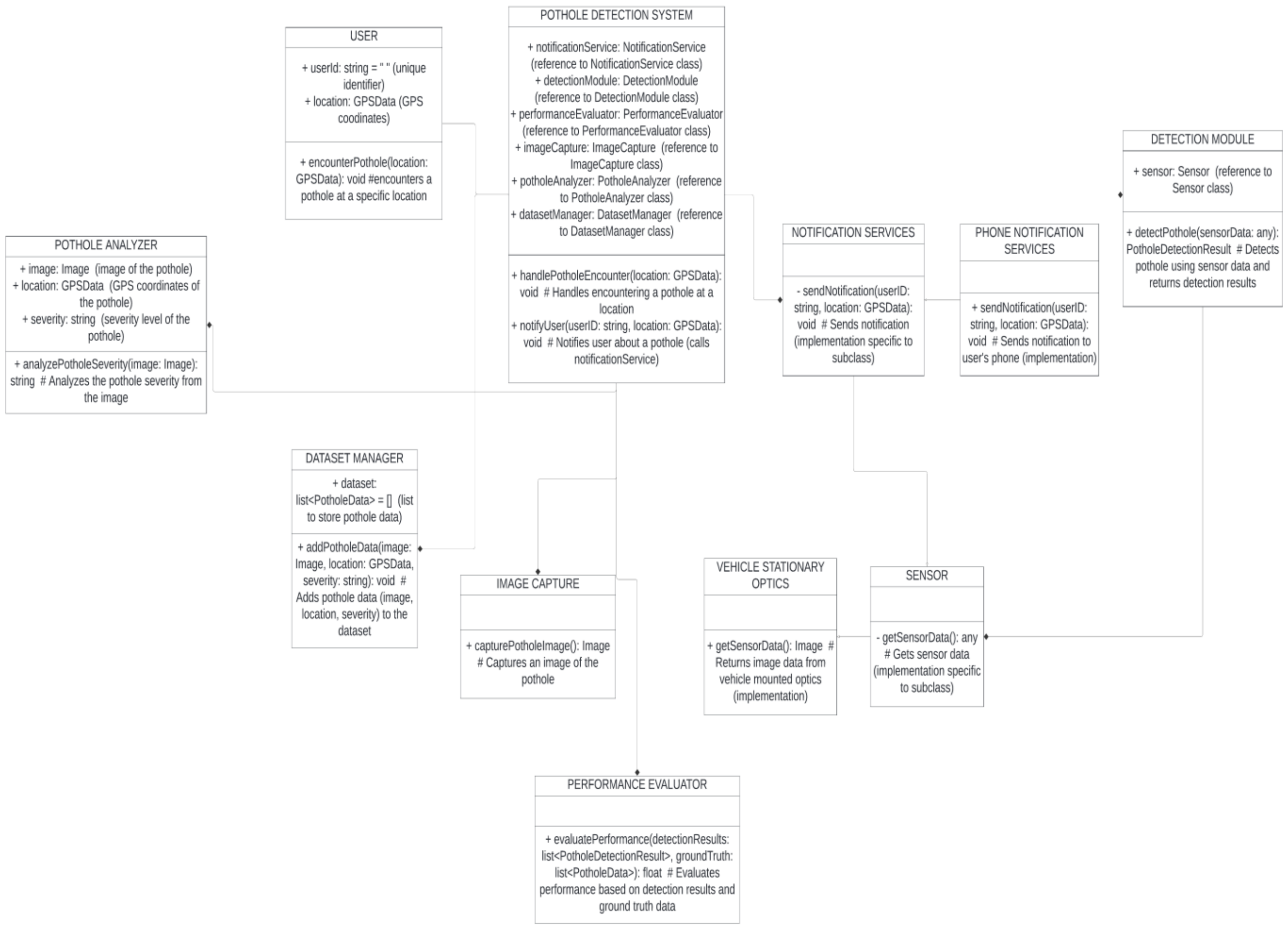

USER
+ userId: string = " " (unique identifier)
+ location: GPSData (GPS coodinates)
+ encounterPothole(location: GPSData): void #encounters a pothole at a specific location
POTHOLE DETECTION SYSTEM
+ notificationService: NotificationService (reference to NotificationService class)
+ detectionModule: DetectionModule (reference to DetectionModule class)
+ performanceEvaluator: PerformanceEvaluator (reference to PerformanceEvaluator class)
+ imageCapture: ImageCapture (reference to ImageCapture class)
+ potholeAnalyzer: PotholeAnalyzer (reference to PotholeAnalyzer class)
+ datasetManager: DatasetManager (reference to DatasetManager class)
+ handlePotholeEncounter(location: GPSData): void # Handles encountering a pothole at a location
+ notifyUser(userID: string, location: GPSData): void # Notifies user about a pothole (calls notificationService)
DETECTION MODULE
+ sensor: Sensor (reference to Sensor class)
+ detectPothole(sensorData: any): PotholeDetectionResult # Detects pothole using sensor data and returns detection results
POTHOLE ANALYZER
+ image: Image (image of the pothole)
+ location: GPSData (GPS coordinates of the pothole)
+ severity: string (severity level of the pothole)
+ analyzePotholeSeverity(image: Image): string # Analyzes the pothole severity from the image
NOTIFICATION SERVICES
- sendNotification(userID: string, location: GPSData): void # Sends notification (implementation specific to subclass)
PHONE NOTIFICATION SERVICES
+ sendNotification(userID: string, location: GPSData): void # Sends notification to user's phone (implementation)
DATASET MANAGER
+ dataset: list<PotholeData> = [] (list to store pothole data)
+ addPotholeData(image: Image, location: GPSData, severity: string): void # Adds pothole data (image, location, severity) to the dataset
IMAGE CAPTURE
+ capturePotholeImage(): Image # Captures an image of the pothole
VEHICLE STATIONARY OPTICS
+ getSensorData(): Image # Returns image data from vehicle mounted optics (implementation)
SENSOR
- getSensorData(): any # Gets sensor data (implementation specific to subclass)
PERFORMANCE EVALUATOR
+ evaluatePerformance(detectionResults: list<PotholeDetectionResult>, groundTruth: list<PotholeData>): float # Evaluates performance based on detection results and ground truth data

Fig 4.3.2 Class Diagram

4.3.3 Sequence Diagram: -

A pothole detection system is shown in the sequence diagram. Road data is constantly being captured by sensors like cameras. To find potholes, a processing unit analyzes this data. A pothole's position and severity are determined, and an optional alarm is delivered along with other information. Repeating this method, the algorithm never stops looking for fresh data.

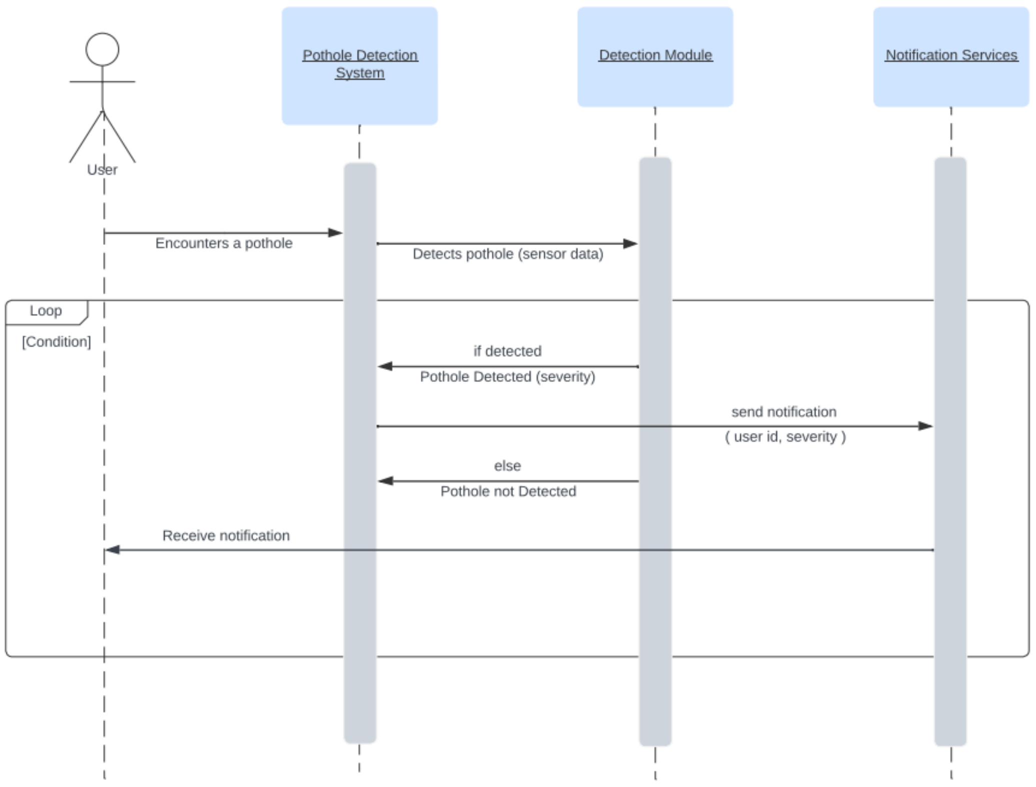


Fig 4.3.3 Sequence Diagram

4.3.4 Component Diagram: -

This component diagram illustrates a car's pothole detection system. The entire procedure is coordinated by the pothole detection system, which serves as the focal point. It uses sensor data to collect information about the road surface, which can be from LiDAR or cameras. The raw material of the system is this data. Then, to locate potholes, the pothole detection system analyzes this data, perhaps using a detection module (which isn't expressly mentioned). Lastly, the connection between the user-notified component and the pothole detection system in the diagram suggests that the system has the ability to notify the user of potholes that have been detected.

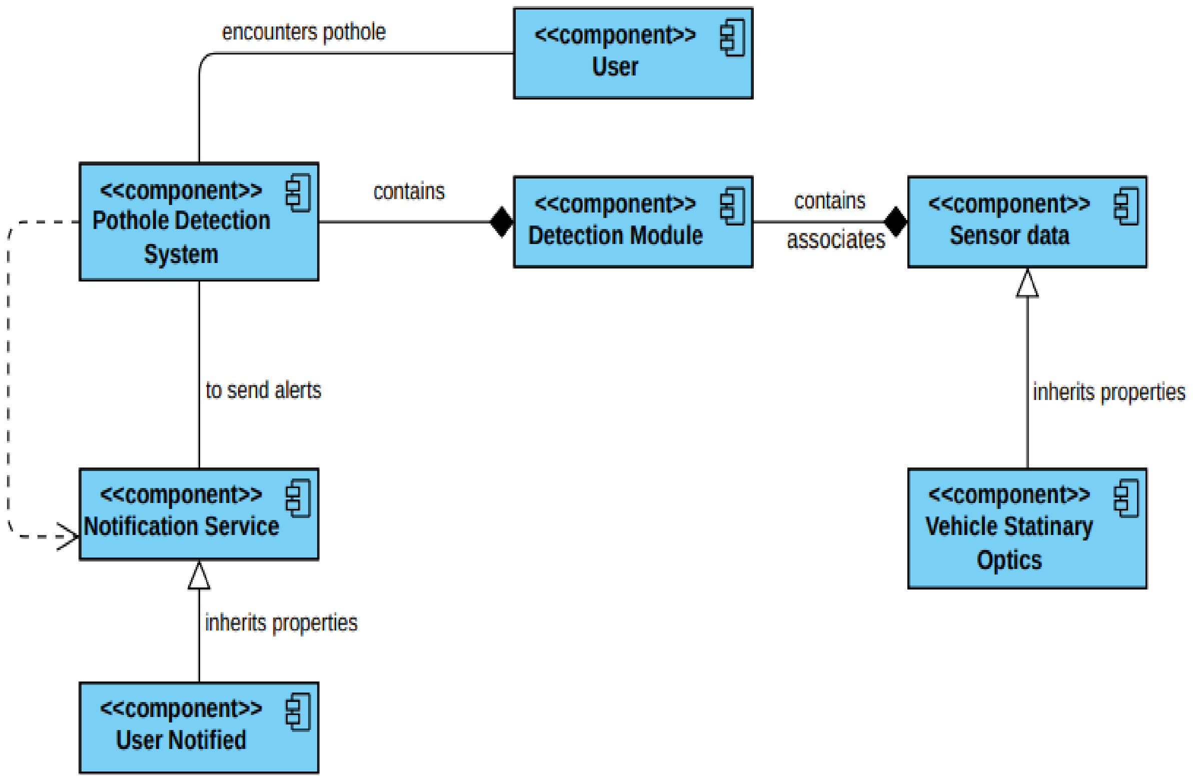
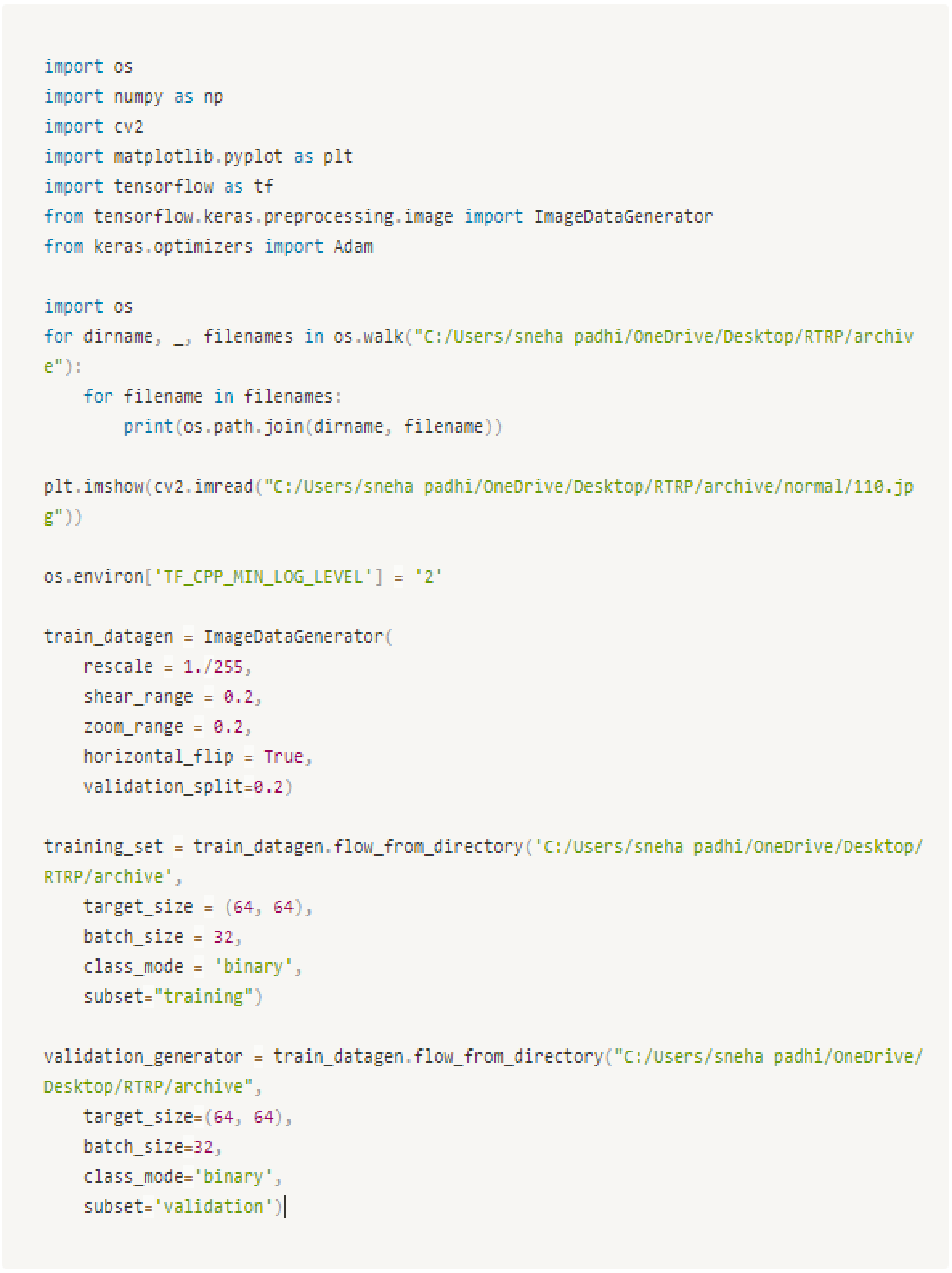

encounters pothole
<<component>> User
<<component>> Pothole Detection System
contains
<<component>> Detection Module
contains
associates
<<component>> Sensor data
to send alerts
inherits properties
<<component>> Notification Service
<<component>> Vehicle Statinary Optics
inherits properties
<<component>> User Notified

Fig 4.3.4 Component Diagram

5. IMPLEMENTATION

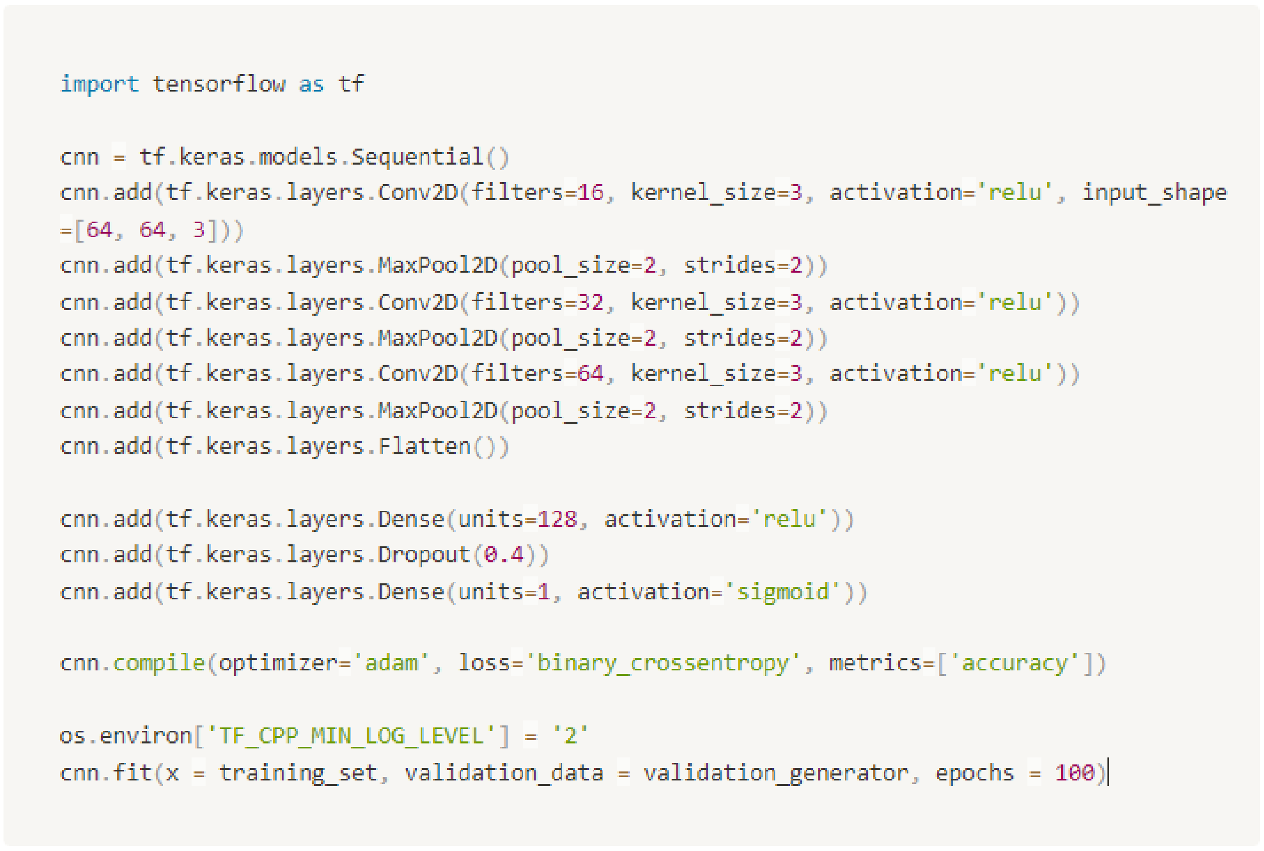
5.1 Code Snippets

5.1.1 Data Collection and Preprocessing



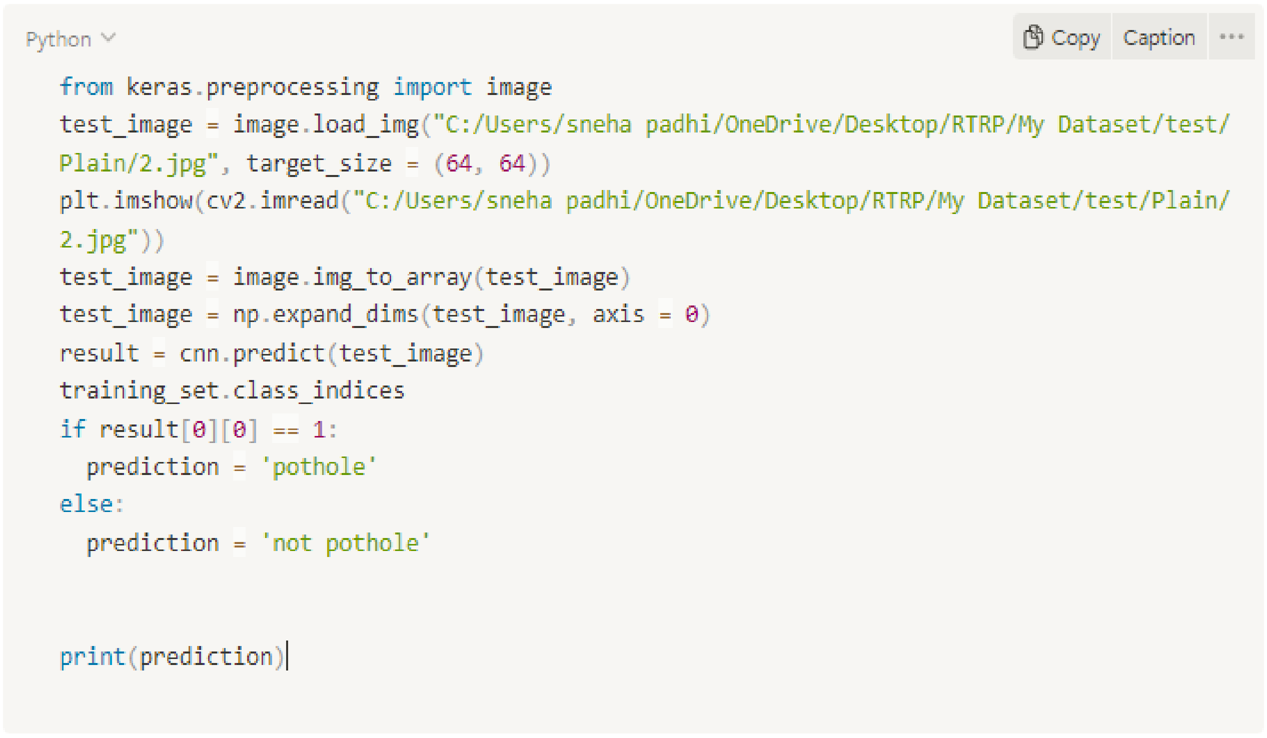
```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam

import os
for dirname, _, filenames in os.walk("C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive"):
    for filename in filenames:
        print(os.path.join(dirname, filename))

plt.imshow(cv2.imread("C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive/normal/110.jpg"))

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

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

training_set = train_datagen.flow_from_directory('C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive',
    target_size = (64, 64),
    batch_size = 32,
    class_mode = 'binary',
    subset="training")

validation_generator = train_datagen.flow_from_directory("C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive",
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary',
    subset='validation')
```

5.1.2 CNN Model Training



```
import tensorflow as tf

cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(filters=16, kernel_size=3, activation='relu', input_shape
=[64, 64, 3]))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=32, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Conv2D(filters=64, kernel_size=3, activation='relu'))
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))
cnn.add(tf.keras.layers.Flatten())

cnn.add(tf.keras.layers.Dense(units=128, activation='relu'))
cnn.add(tf.keras.layers.Dropout(0.4))
cnn.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

cnn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
cnn.fit(x = training_set, validation_data = validation_generator, epochs = 100)
```

5.1.3 Model Prediction



```
from keras.preprocessing import image
test_image = image.load_img("C:/Users/sneha padhi/OneDrive/Desktop/RTRP/My Dataset/test/
Plain/2.jpg", target_size = (64, 64))
plt.imshow(cv2.imread("C:/Users/sneha padhi/OneDrive/Desktop/RTRP/My Dataset/test/Plain/
2.jpg"))
test_image = image.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image)
training_set.class_indices
if result[0][0] == 1:
  prediction = 'pothole'
else:
  prediction = 'not pothole'


print(prediction)
```

5.1.4 Real-Time Detection and Alert

```python
import cv2
import numpy as np
from keras.models import load_model
import pygame

pygame.init()
pygame.mixer.music.load('beep.mp3')

model = load_model('model2.h5')

video_path = 'DASH CAM 2016 01 29 (42 Miles of Potholes).mp4'

cap = cv2.VideoCapture(video_path)

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    img_resized = cv2.resize(frame, (64, 64))

    img_resized = img_resized.astype('float32') / 255.0
    img_resized = np.expand_dims(img_resized, axis=0)

    prediction_raw = model.predict(img_resized, verbose=0)
    maxArg = np.argmax(prediction_raw[0])
    pred = maxArg
    conf = prediction_raw[0][maxArg]

    if pred == 0 and conf >= 0.7:
        cv2.putText(frame, 'pothole', (0, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0),
2, cv2.LINE_AA)
        pygame.mixer.music.play()

    cv2.imshow('Frame', frame)
    if cv2.waitKey(2) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

5.1.5 Graphical User Interface

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
import tkinter as tk
from PIL import Image, ImageTk
import pygame

model = load_model('C:\\Users\\sneha padhi\\OneDrive\\Desktop\\RTRP\\model2.h5')

video_path = 'C:\\Users\\sneha padhi\\OneDrive\\Desktop\\RTRP\\DASH CAM 2016 01 29 (42 M
iles of Potholes).mp4'

root = tk.Tk()
root.title("Pothole Detection System")

canvas = tk.Canvas(root, width=1240, height=600)
canvas.pack()

label = tk.Label(root, text="", font=("Helvetica", 24))
label.pack()

pygame.init()
pygame.mixer.music.load('C:\\Users\\sneha padhi\\OneDrive\\Desktop\\RTRP\\beep.mp3')

cap = cv2.VideoCapture(video_path)

def process_frame():
    ret, frame = cap.read()
    if not ret:
        return

    img_resized = cv2.resize(frame, (64, 64))

    img_resized = img_resized.astype('float32') / 255.0
    img_resized = np.expand_dims(img_resized, axis=0)

    prediction_raw = model.predict(img_resized, verbose=0)
    maxArg = np.argmax(prediction_raw[0])
    pred = maxArg
    conf = prediction_raw[0][maxArg]
    img_resized = img_resized.astype('float32') / 255.0
    img_resized = np.expand_dims(img_resized, axis=0)

    prediction_raw = model.predict(img_resized, verbose=0)
    maxArg = np.argmax(prediction_raw[0])
    pred = maxArg
    conf = prediction_raw[0][maxArg]

    if pred == 0 and conf >= 0.7:
        label.config(text="Pothole Detected!")
        pygame.mixer.music.play()
    else:
        label.config(text="")

    img = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    img = Image.fromarray(img)
    img = ImageTk.PhotoImage(img)

    canvas.create_image(0, 0, image=img, anchor='nw')
    canvas.image = img

    root.after(1, process_frame)

process_frame()

root.mainloop()

cap.release()
cv2.destroyAllWindows()
```

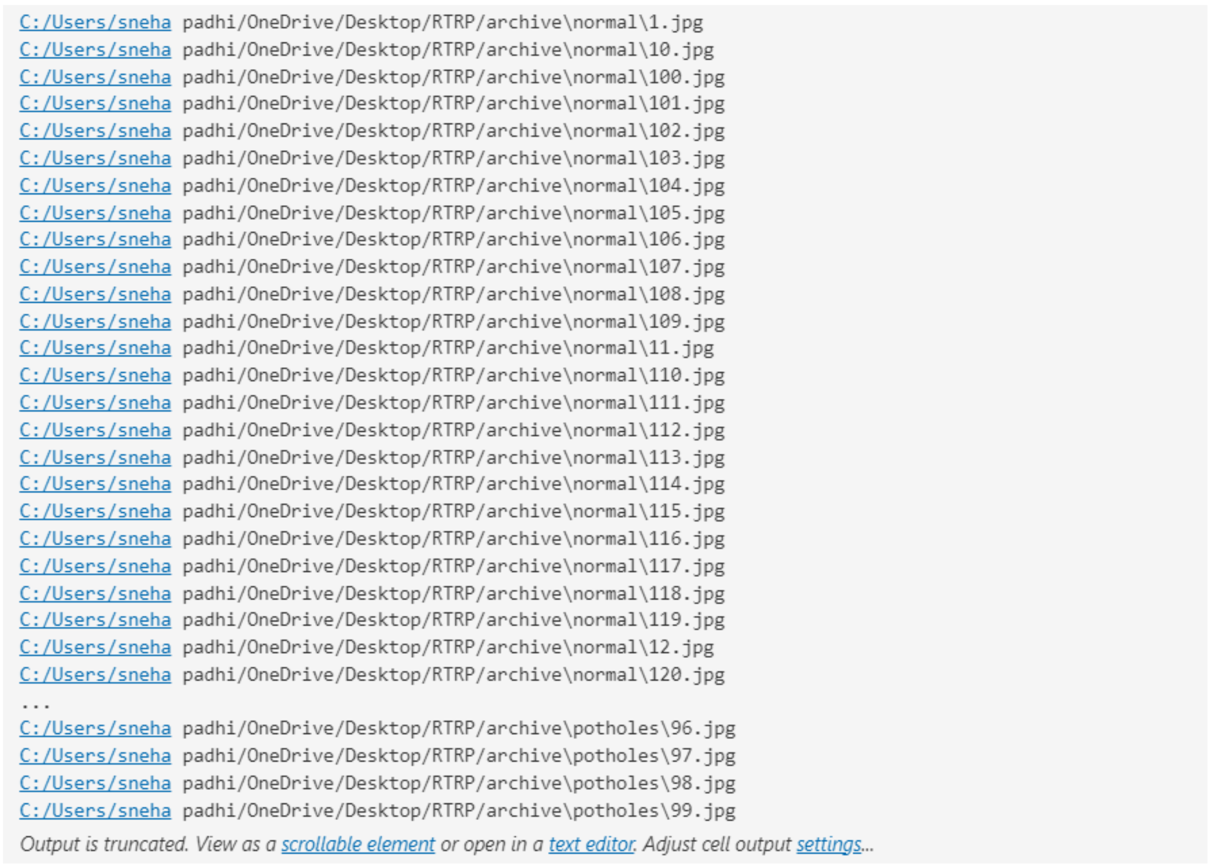
5.1.6 Evaluation Metrics

```
import numpy as np
from tensorflow.keras.models import load_model
from sklearn.metrics import classification_report, confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator

model = load_model('model2.h5')

test_dir = 'C:\\Users\\sneha padhi\\OneDrive\\Desktop\\RTRP\\archive'

test_datagen = ImageDataGenerator(rescale=0.255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary'

X_test, y_test = next(test_generator)

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Loss: {loss}")
print(f"Accuracy: {accuracy}")

y_pred = model.predict(X_test)

if len(y_test.shape) > 1 and y_test.shape[1] > 1:
    y_test_classes = np.argmax(y_test, axis=1)
else:
    y_test_classes = y_test

y_pred_classes = np.argmax(y_pred, axis=1)

print("Classification Report:")
print(classification_report(y_test_classes, y_pred_classes, target_names=['Class 0', 'Cl
ass 1']))

print("Confusion Matrix:")
print(confusion_matrix(y_test_classes, y_pred_classes))
```

5.2 Output

5.2.1 Output 1

- The paths of images in the dataset are printed.



```
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\1.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\10.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\100.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\101.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\102.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\103.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\104.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\105.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\106.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\107.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\108.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\109.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\11.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\110.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\111.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\112.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\113.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\114.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\115.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\116.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\117.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\118.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\119.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\12.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\normal\120.jpg
...
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\potholes\96.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\potholes\97.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\potholes\98.jpg
C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive\potholes\99.jpg
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

- A sample image from the dataset is displayed.

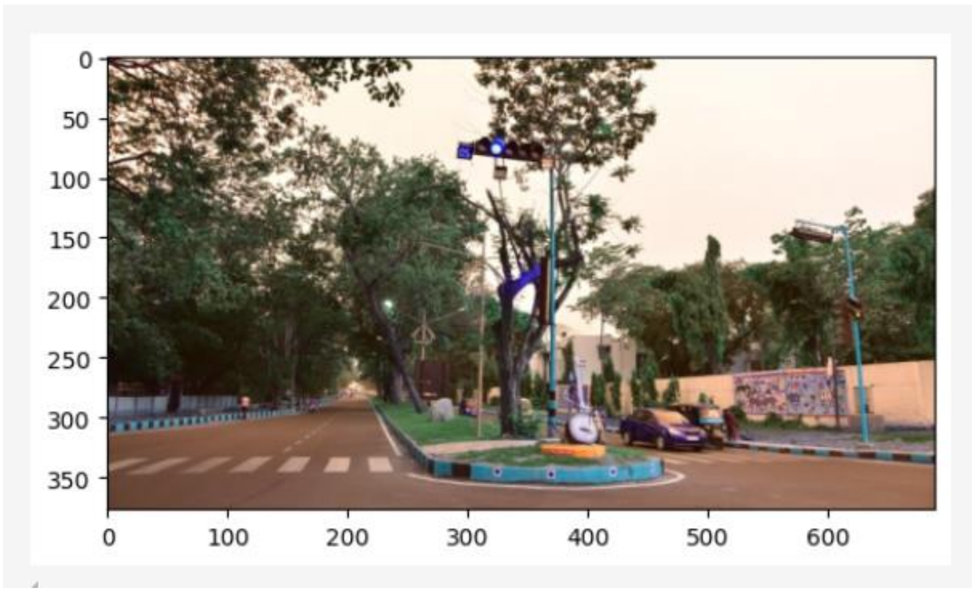
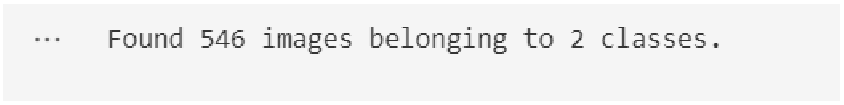
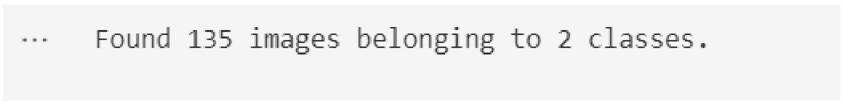


Fig 5.2.1 Output 1

- The training and validation datasets are prepared, with the generator indicating the number of images found for each.



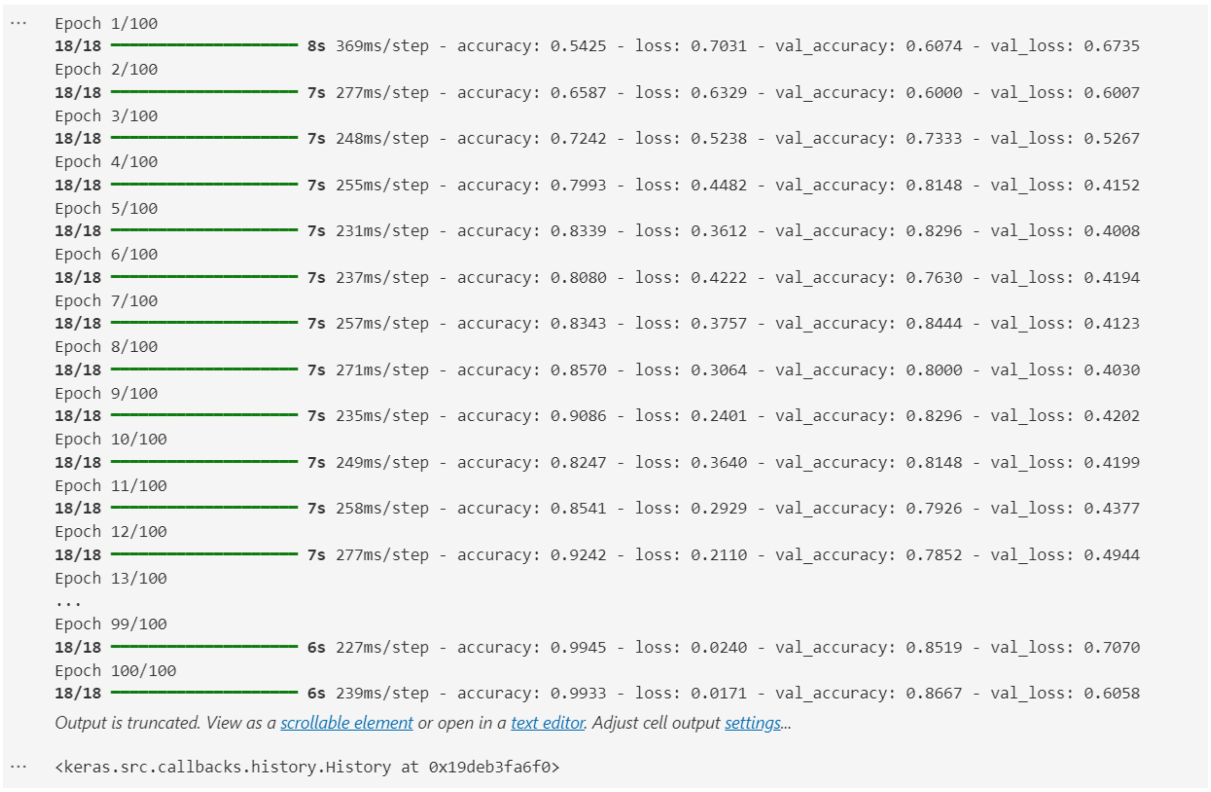
```
...   Found 546 images belonging to 2 classes.
```



```
...   Found 135 images belonging to 2 classes.
```

5.2.2 Output 2

- The CNN model structure is printed.
- Training progress is displayed, showing the accuracy and loss for each epoch.



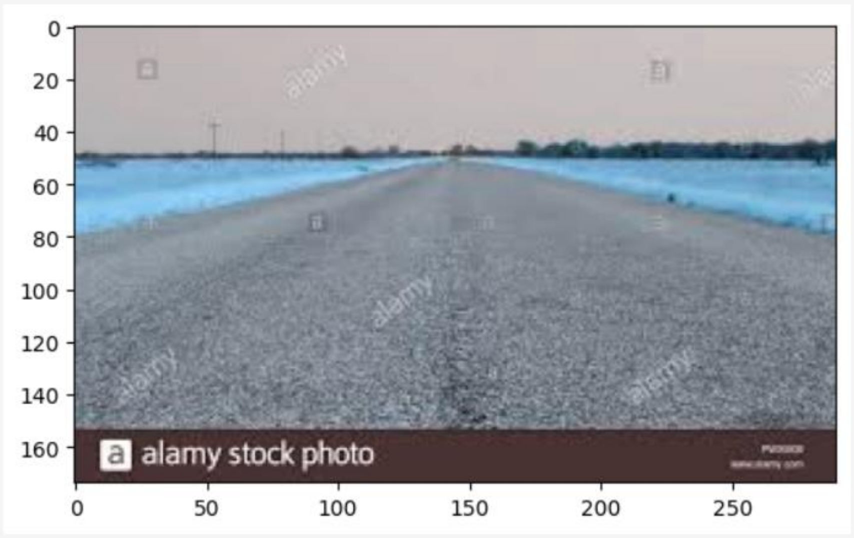
```
...   Epoch 1/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 8s 369ms/step - accuracy: 0.5425 - loss: 0.7031 - val_accuracy: 0.6074 - val_loss: 0.6735
      Epoch 2/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 277ms/step - accuracy: 0.6587 - loss: 0.6329 - val_accuracy: 0.6000 - val_loss: 0.6007
      Epoch 3/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 248ms/step - accuracy: 0.7242 - loss: 0.5238 - val_accuracy: 0.7333 - val_loss: 0.5267
      Epoch 4/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 255ms/step - accuracy: 0.7993 - loss: 0.4482 - val_accuracy: 0.8148 - val_loss: 0.4152
      Epoch 5/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 231ms/step - accuracy: 0.8339 - loss: 0.3612 - val_accuracy: 0.8296 - val_loss: 0.4008
      Epoch 6/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 237ms/step - accuracy: 0.8080 - loss: 0.4222 - val_accuracy: 0.7630 - val_loss: 0.4194
      Epoch 7/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 257ms/step - accuracy: 0.8343 - loss: 0.3757 - val_accuracy: 0.8444 - val_loss: 0.4123
      Epoch 8/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 271ms/step - accuracy: 0.8570 - loss: 0.3064 - val_accuracy: 0.8000 - val_loss: 0.4030
      Epoch 9/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 235ms/step - accuracy: 0.9086 - loss: 0.2401 - val_accuracy: 0.8296 - val_loss: 0.4202
      Epoch 10/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 249ms/step - accuracy: 0.8247 - loss: 0.3640 - val_accuracy: 0.8148 - val_loss: 0.4199
      Epoch 11/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 258ms/step - accuracy: 0.8541 - loss: 0.2929 - val_accuracy: 0.7926 - val_loss: 0.4377
      Epoch 12/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 7s 277ms/step - accuracy: 0.9242 - loss: 0.2110 - val_accuracy: 0.7852 - val_loss: 0.4944
      Epoch 13/100
      ...
      Epoch 99/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 6s 227ms/step - accuracy: 0.9945 - loss: 0.0240 - val_accuracy: 0.8519 - val_loss: 0.7070
      Epoch 100/100
      18/18 ━━━━━━━━━━━━━━━━━━━━ 6s 239ms/step - accuracy: 0.9933 - loss: 0.0171 - val_accuracy: 0.8667 - val_loss: 0.6058
      Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

...   <keras.src.callbacks.history.History at 0x19deb3fa6f0>
```

Fig 5.2.2 Output 2

5.2.3 Output 3

- The test image is displayed.
- The model's prediction is printed, indicating whether a pothole was detected

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
not pothole
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
pothole
```

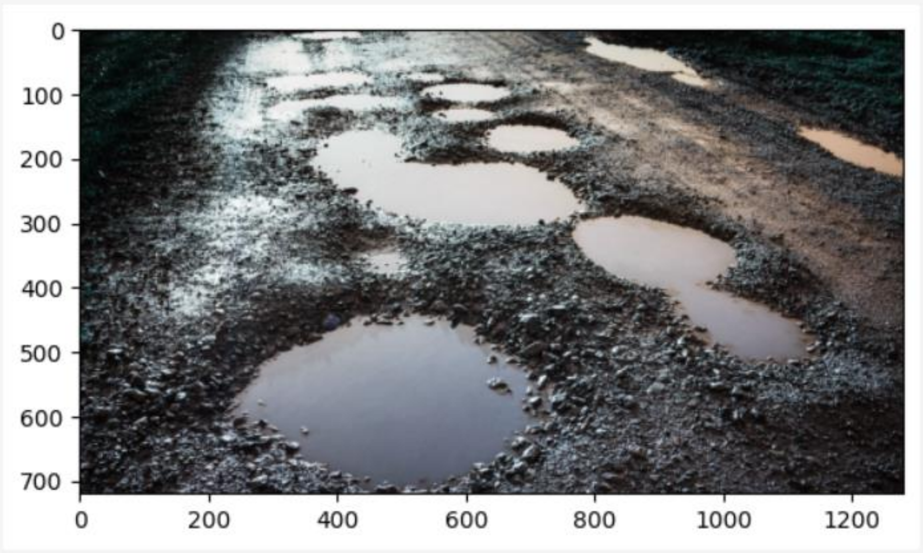


Fig 5.2.3 Output 3

5.2.4 Output 4

- Real-time video is displayed with overlaid detection results.
- An audio alert is triggered when a pothole is detected.

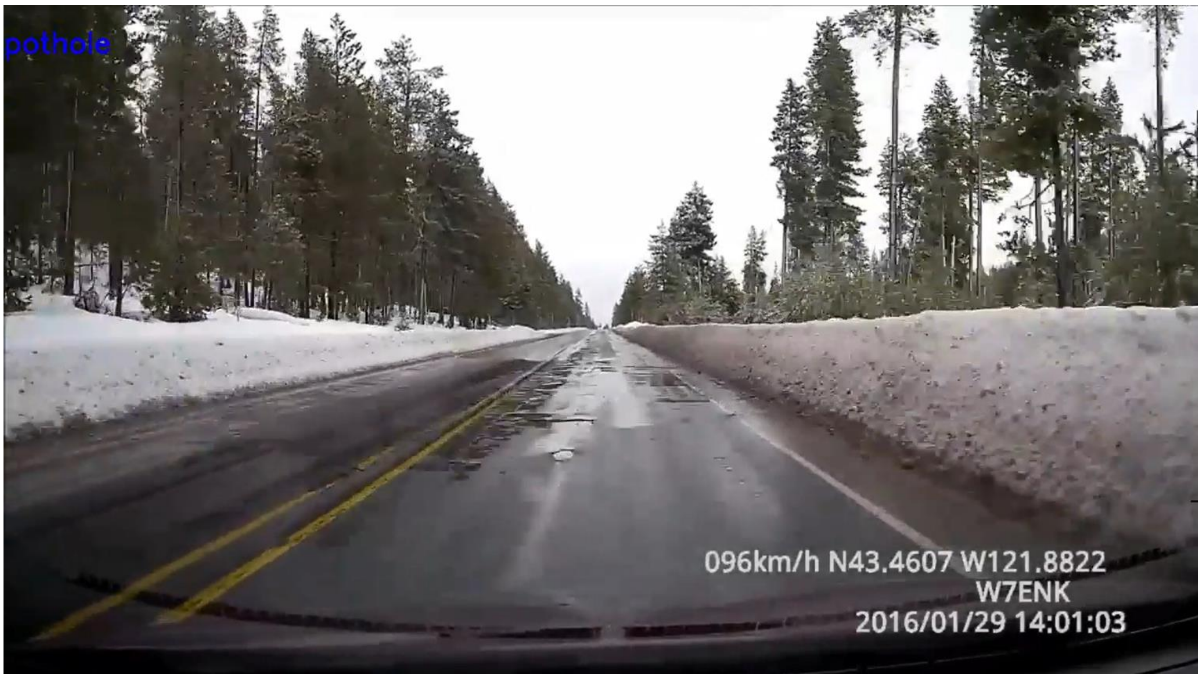


Fig 5.2.4 Output 4

5.2.5 Output 5

- Real-time video is displayed on Tkinter canvas with detection results displayed below.
- An audio alert is triggered when a pothole is detected.

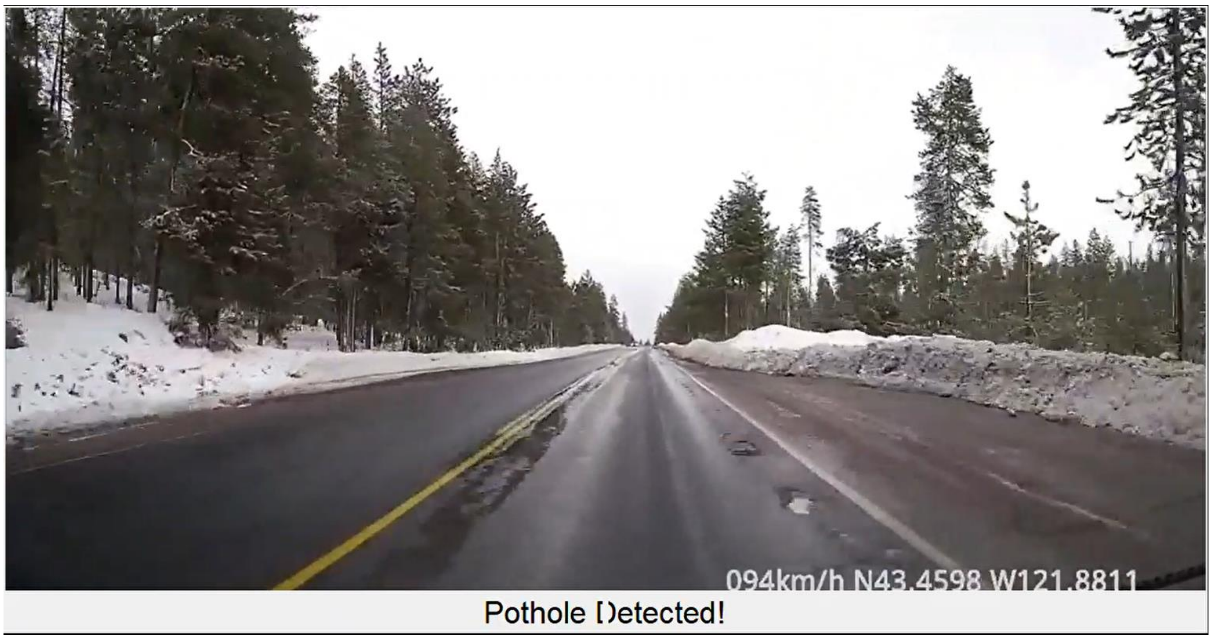
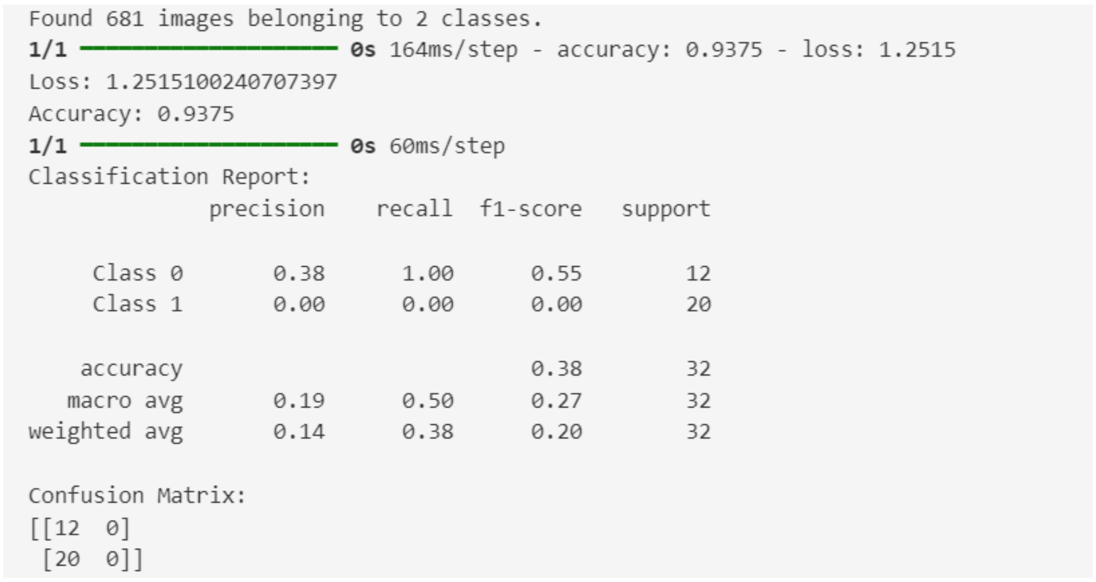


Fig 5.2.5 Output 5

5.2.6 Output 6

- Loss and Accuracy of the model is calculated on the provided test data.
- A detailed classification report is generated, showcasing metrics such as precision, recall and F1-score for each class.
- The confusion matrix is generated, offering insights into the model's performance with true positive, false positive, true negative, and false negative counts.



```
Found 681 images belonging to 2 classes.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 164ms/step - accuracy: 0.9375 - loss: 1.2515
Loss: 1.2515100240707397
Accuracy: 0.9375
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step
Classification Report:
              precision    recall  f1-score   support

     Class 0       0.38      1.00      0.55        12
     Class 1       0.00      0.00      0.00        20

    accuracy                           0.38        32
   macro avg       0.19      0.50      0.27        32
weighted avg       0.14      0.38      0.20        32

Confusion Matrix:
[[12  0]
 [20  0]]
```

Fig 5.2.6 Output 6

5.3 Test Cases

Test Case ID	Test scenario	Pre-conditions	Test Steps	Test Data	Expected Results	Post-Conditions	Actual Results	Status
1	Verify Pothole Detection in Training Data	Dataset located at "C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive" Model trained with ImageDataGenerator and CNN architecture	Load the training dataset using ImageDataGenerator. Train the CNN model with the training dataset for 100 epochs.	Training images located in "C:/Users/sneha padhi/OneDrive/Desktop/RTRP/archive"	Model should achieve a reasonable accuracy on the training and validation sets.	Trained model (cnn) available for further evaluation	Accuracy metrics printed after training	PASS
2	Verify Pothole Prediction on Test Image	Trained model (cnn) loaded with satisfactory training results. Test image located at "C:/Users/sneha padhi/OneDrive/Desktop/RTRP/My Dataset/test/Plain/2.jpg"	Load the test image. Perform prediction using the loaded cnn model.	Test image "2.jpg" located in "C:/Users/sneha padhi/OneDrive/Desktop/RTRP/My Dataset/test/Plain/"	Prediction result should indicate either "pothole" or "not pothole".	Prediction outcome (prediction) available.	Printed prediction result.	PASS
3	Verify Pothole Detection in Real-time Video Stream	Model model2.h5 loaded with satisfactory training results. Video file "DASH CAM 2016 01 29 (42 Miles of Potholes).mp4" available at	Open the video file. Process each frame, resize it to 64x64, and normalize. Use the model to predict presence of potholes in each frame. Play sound (beep.mp3) if a pothole is	Video file "DASH CAM 2016 01 29 (42 Miles of Potholes).mp4"	Real-time display of video frames with "pothole" indicated where detected.	Video stream processed and potential potholes indicated audibly and visually.	Interactive display of video frames with detection outcomes.	PASS

		specified path.	detected with confidence >= 0.7.					
4	Evaluate Model Performance on Test Dataset	Model model2.h5 loaded with satisfactory training results. Test dataset located at "C:\Users\sneha padhi\OneDrive\Desktop\RTRP\archive".	Load the test dataset using ImageDataGenerator. Evaluate the model on the test dataset to obtain loss and accuracy metrics. Generate classification report and confusion matrix.	Test images located in "C:\Users\sneha padhi\OneDrive\Desktop\RTRP\archive".	Metrics such as loss, accuracy, classification report, and confusion matrix should indicate model performance.	Evaluation metrics and analysis available for model assessment.	Printed evaluation metrics, classification report, and confusion matrix.	PASS
5	Verify Pothole Detection in Real-time Video with GUI	Model model2.h5 loaded with satisfactory training results. Video file "DASH CAM 2016 01 29 (42 Miles of Potholes).mp4" available at specified path.	Launch GUI application. Open the video file. Display each frame in the GUI canvas. Predict presence of potholes in each frame. Display "Pothole Detected!" message and play sound (beep.mp3) if a pothole is detected with confidence >= 0.7.	Video file 1. "DASH CAM 2016 01 29 (42 Miles of Potholes).mp4" 2. "test_vid.mp4"	Real-time GUI display with "pothole" indication and audio feedback where detected.	Interactive GUI application running with real-time detection capabilities.	Functional GUI with detection feedback.	1. PASS 2. FAIL

Table 5.3 Test Cases

6. CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This project successfully developed a real-time pothole detection system by leveraging convolutional neural networks (CNNs) and computer vision techniques, significantly contributing to enhanced road safety. The primary objective was to create a system that could accurately detect potholes from dashcam footage and alert drivers in real-time, thus preventing potential vehicle damage and accidents.

6.1.1 Achievements

1. **Dataset utilization:**

   The dataset used in the project was very diverse and contained over 700 images on Kaggle of various road conditions, both with and without potholes. This kind of rich dataset exposed the model to most real-life situations, rendering the model very robust and accurate. Every image was labeled; thus, the model could learn to recognize features that make a pothole, such as irregular shape and texture.

2. **Model Development:**

   Implemented a Convolutional Neural Network through TensorFlow and Keras and trained it. Several convolutional layers were used to extract the features, followed by max-pooling layers for reducing dimensionality, and then dense layers for classification. This structure provided proper learning about characteristics of potholes by the model while keeping it computationally efficient.

3. **Image Preprocessing:**

   ImageDataGenerator was used, rescaling, shearing, zooming, and horizontal flipping having been incorporated. Due to this augmentation technique, the model will be exposed to many different transformations of the input data. Thus, it created a generalized model and improved performance on unseen data.

4. **Model Training and Validation:**

   The preprocessing dataset was trained into the model; its performance had to be validated against a different validation set. This process is done using categorical cross-entropy as the loss function and Adam optimizer for efficient training. During training, the accuracy and loss of the model were tracked to ensure it was learning fine without any overfitting.

5. **Real-Time Detection and Alert System:**

   This is where a real-time detection system came into play, which worked with video frames from the dashcam footage. Each frame was resized, normalized, and then fed into the trained model to make a prediction. If the model detected a pothole with high enough confidence, it would visibly alert in the video frame and provide an audio warning via Pygame. This real-time feedback mechanism lets drivers take appropriate corrective measures, enhancing road safety.

6. **Performance and Reliability:**

   Testing showed an extremely high system accuracy in pothole detection, thus validating the efficiency of using CNNs and computer vision in such applications. This robust design ensured that the functionality of the model was relatively high in real time so that it can, in turn, alert drivers reliably.

6.2 Future Scope:

The success of this project lays a solid foundation for further enhancements and future developments:

- **Dataset Expansion:** Anything that will increase the dataset size and diversity to some extent, further enhancing the accuracy and fitting of the model. More images covering different weather conditions, lighting scenarios, and types of roads may make the model more versatile.
- **Model Optimization:** Other CNN architectures, a variety of hyperparameters, and some of the advanced techniques like transfer learning can also be experimented with for better performance.

- **Vehicle Integration:** The integration into commercial vehicles itself as a part of ADAS—Advanced Driver-Assistance Systems—can be quite pervasive in its benefits. This would involve collaboration with the automotive manufacturers, testing under real-world drive conditions.
- **Extended Applications:** In this respect, the framework developed in the research can easily be extended to detect more road anomalies, such as cracks, debris, and construction zones, thereby enhancing its utility.

7. REFERENCES

[1] Samir Berjaoui, Anas Al-Shaghouri, Rami Alkhatib: Pothole detection system using deep learning, 2021

[2] Oche Alexander Egaji, Gareth Evans, Mark Graham Griffiths, Gregory Islas: Real-time machine learning based approach for Pothole detection, 2021

[3] Mingxing Gao, Xu Wang, Shoulin Zhu, Peng Guan: Detection and segmentation of concrete pothole based on Image Processing, 2020

[4] Hadistian Muhammad Hanif, Zener Sukra Lie, Winda Astuti, Sofyan Tan: Pothole detection system design with proximity sensor to provide motorcycle with warning system and increase road safety, 2020

[5] Sung-Sik Park, Van-Than Tran, Dong-Eun Lee: Application of various YOLO Models for Computer Vision-Based Real-Time Pothole Detection, 2021

[6] Susmita Patra, Asif Iqbal Middya, Sarbani Roy: PotSpot: Participatory sensing-based monitoring system for pothole detection using deep learning, 2021

[7] Deepak Kumar Dewangan, Satya Prakash Sahu: PotNet: Pothole detection for autonomous vehicle system using Convolutional Neural Network, 2020

[8] Kashish Bansal, Kashish Mittal, Gautam Ahuja, Ashima Singh, Sukhpal Singh Gill: DeepBus: Machine Learning based Real Time Pothole Detection system for smart transportation using IoT, 2020

[9] Yeoh Keng Yik, Nurul Ezaila Alias, Yusmeeraz Yusof and Suhaila Isaak: A Real-time Pothole Detection Based on Deep Learning Approach, 2021

[10] M.S. Kamalesh, Bharatiraja Chokkalingam, Jeevanantham Arumugam, Gomathy Sengottaiyan, Shanmugavadivel Subramani, Mansoor Ahmad Shah: Intelligent Real Time Pothole Detection and Warning System for Automobile Applications Based on IoT Technology, 2020

[11] Pranjal A. Chitale, Kaustubh Y. Kekre, Hrishikesh R. Shenai, Ruhina Karani, Jay P. Gala: Pothole Detection and Dimension Estimation System using Deep Learning (YOLO) and Image Processing, 2020

[12] Om Khare, Shubham Gandhi, Aditya Rahalkar, Sunil Mane: YOLOv8-Based Visual Detection of Road Hazards: Potholes, Sewer Covers, and Manholes, 2023

[13] Pothole Radar:

https://play.google.com/store/apps/details?id=com.jcl.pothole&hl=en_IN&gl=US

[14] Pothole Patrol:

https://play.google.com/store/apps/details?id=com.jcl.pothole&hl=en_IN&gl=US

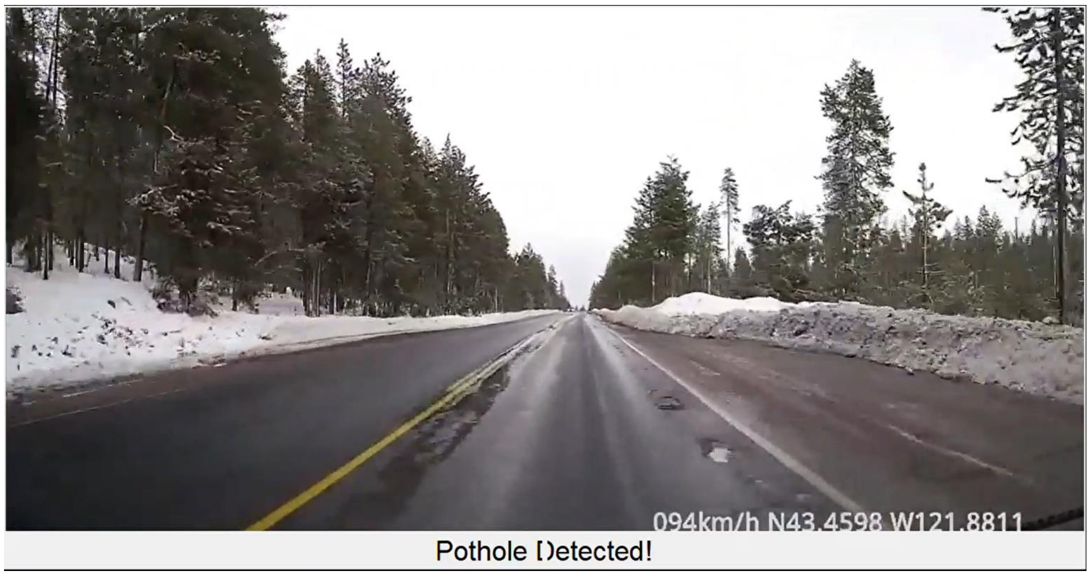
[15] RoadBounce Pothole Guard:

https://play.google.com/store/apps/details?id=com.definitics.rb.pothole.guard

APPENDIX

i. Snapshot of the Result

The final output of the Real-Time Pothole Detection Module is a dynamic display on a Tkinter GUI, where the live video feed from the dashcam is shown. When a pothole is detected, an alert message "Pothole Detected!" is displayed on the screen, and an audio alert is played to notify the driver immediately.

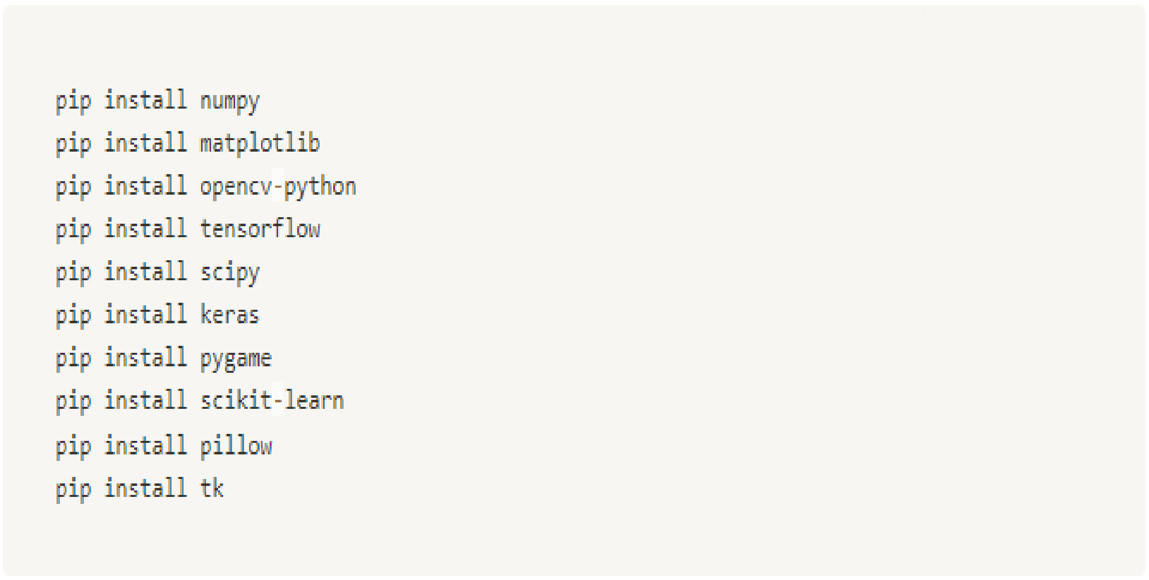


Fig(i) Final Output

ii. Software Installations

To set up and run the pothole detection system described in this project, the following software installations are required:

1. **Python:** Ensure that Python 3.12 is installed on your system. You can download it from the official Python website.
2. **Python Libraries:** Install the necessary Python libraries using pip. The required libraries include NumPy, Matplotlib, OpenCV, TensorFlow, SciPy, Keras, Pygame, Scikit-learn, Tkinter, and Pillow. You can install these libraries by running the following commands:



```
pip install numpy
pip install matplotlib
pip install opencv-python
pip install tensorflow
pip install scipy
pip install keras
pip install pygame
pip install scikit-learn
pip install pillow
pip install tk
```

3. **Integrated Development Environment (IDE):** It is recommended to use an IDE for developing and running the project. Suitable options include:
    - **Visual Studio Code (VSCode):** A lightweight yet powerful IDE that supports Python development.
    - **Jupyter Notebook:** An interactive environment suitable for prototyping and data exploration.
    - **Pycharm:** A comprehensive IDE specifically designed for Python development.