IARJSET ISSN (O) 2393-8021, ISSN (P) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology

Impact Factor 8.311 ⨳ Peer-reviewed & Refereed journal ⨳ Vol. 12, Issue 8, August 2025

DOI: 10.17148/IARJSET.2025.12829

Deep Learning-based Pothole Detection and Severity Classification with Location Mapping

Chandana B D

Student, MCA Department, Bangalore Institute of Technology, Bangalore, India

Abstract: This paper introduces a deep learning–powered system designed to detect potholes from road images, assess their severity, and map their location. At the core of the system is a Convolutional Neural Network (CNN) trained on a curated dataset of road images containing both potholes and non-potholes. The model is able to accurately identify potholes and further categorize their severity into three levels—Minor, Moderate, or Major—based on its confidence score. To demonstrate the system's functionality, detected potholes are assigned random locations within Bengaluru, and a detailed PDF report is generated. The report includes the detection results, supporting images, a severity distribution chart, and location information. The entire solution is deployed as a Flask-based web application, offering a simple interface where users can upload road images, receive real-time predictions, and download comprehensive reports.

Keywords: Pothole Detection, Deep Learning, Convolutional Neural Networks (CNN), Image Classification, Road Safety, Web Application (Flask).

I. INTRODUCTION

Potholes are a major concern for road users, often leading to accidents, vehicle damage, and traffic disruptions. Relying on traditional manual inspections is both time-consuming and inefficient, highlighting the need for smarter solutions. With the rapid progress in computer vision and deep learning, it is now possible to design automated systems that can detect potholes accurately, at scale, and in real time.

In this work, we present a CNN-based approach integrated into a Flask web application for pothole detection, severity assessment, and report generation. The system allows users to upload road images, which are then analyzed by the model to predict the presence of potholes and classify their severity as Minor, Moderate, or Major. Results are displayed through an intuitive web interface, while an automated PDF report generator supports easy record-keeping and analysis.

Road maintenance plays a vital role in urban infrastructure management, as potholes not only inconvenience drivers but also pose serious safety risks. Existing methods—such as periodic inspections or complaint-based reporting systems—are slow, costly, and often incomplete. By leveraging CNNs and web technologies, our solution provides a more efficient and scalable alternative.

This research demonstrates how deep learning can be applied to road safety challenges, combining a robust detection model with a user-friendly interface to deliver real-time pothole detection, severity classification, and comprehensive reporting.

II. RELATED WORKS

Several research efforts have focused on pothole detection using computer vision, each exploring different techniques:
Edge and Texture Analysis: Early methods used techniques like Canny edge detection and texture analysis to spot potholes. While computationally simple, these approaches were highly sensitive to lighting variations and background noise, limiting their reliability.

Stereo Vision and 3D Reconstruction: Depth-based methods employed stereo cameras to detect road surface depressions with good accuracy. However, their reliance on specialized hardware made them impractical for large-scale deployment.
Traditional Machine Learning Models: Approaches using Support Vector Machines (SVMs) and decision trees with handcrafted features showed some promise but struggled to generalize across diverse road conditions and environments.
Deep Learning Methods: Convolutional Neural Networks (CNNs) have significantly advanced pothole detection by automatically learning robust features from raw images. Recent works include YOLO-based object detection for real-time monitoring and transfer learning with architectures like VGG16 and ResNet.

Building on these foundations, our work takes a step further by not only detecting potholes using a CNN but also classifying their severity, assigning location information, and generating automated PDF reports—all within a single, integrated Flask-based system.

III. PROPOSED WORK

The proposed system brings together computer vision techniques and a user-friendly web interface to deliver an end-to-end tool for pothole detection and severity estimation. What makes this approach unique is the integration of a trained CNN model with real-time image uploads, graphical visualizations, and automated PDF reporting, all within a single workflow.

At its core, the system uses a CNN-based binary classifier that predicts the likelihood of an image containing a pothole. This probability score is interpreted both as the detection confidence and as an indicator of severity. Once the model is trained and validated, it is deployed through a Flask backend that powers the web application.

Users can log in, upload road images, and instantly view the results. The output page displays the uploaded image, an annotated version (if available), a severity bar chart, and the model's confidence score. To support record-keeping and decision-making, the system can also generate a PDF report containing the detection summary, severity graph, image evidence, and a location tag—either randomly assigned for demonstration or provided by the user.

The key novelty lies in the complete pipeline integration: from data processing and model prediction to visualization and reporting. This design ensures that city engineers, road inspectors, or municipal staff can use the tool directly without any coding expertise, making pothole detection more accessible, scalable, and actionable.

IV. METHODOLOGY

The proposed pothole detection system, built using Convolutional Neural Networks (CNNs), operates in two main phases: training and testing (inference).

Data Creation: A dataset of road surface images was assembled using both publicly available sources and manually captured photographs. Each image was carefully labeled as either "pothole" or "non-pothole" to support supervised learning. To make the model robust, images were collected under diverse conditions, including different lighting, weather, and road textures.

Data Preprocessing: Before training, all images were resized to 150 × 150 pixels to fit the CNN's input requirements. Pixel values were normalized to the range [0, 1] to stabilize learning. To improve generalization and reduce overfitting, data augmentation was applied—such as random rotations, flips, zooming, and brightness adjustments—creating a richer and more varied training set.

Model Training: The CNN architecture was designed with multiple convolutional and pooling layers to automatically learn spatial patterns associated with potholes. These features were then passed through fully connected layers for classification, with a sigmoid activation function producing a binary output: pothole or no-pothole.

Testing Phase

Image Input: The trained model was deployed through a Flask-based web application, allowing users to upload road images directly via a simple graphical interface.

Image Preprocessing: Uploaded images are preprocessed in the same way as during training—resized to 150 × 150 pixels and normalized—to ensure consistent input quality.

CNN Detection: The model processes the preprocessed image and generates a probability score indicating the likelihood of a pothole. This score is then expressed as a confidence percentage and mapped to a severity level: Minor, Moderate, Major, or None.

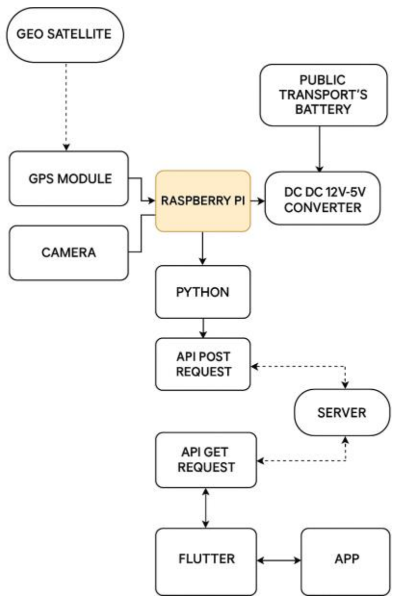


Fig.1 Architecture Design for pothole detection

V. MATHEMATICAL MODEL

The pothole detection task is formulated as a **binary classification problem**, where the system determines whether a given road image contains a pothole (**1**) or not (**0**).
Let:

- **X** represent the input road image.
- **fθ(X)** denote the CNN function with parameters **θ**.
- **Y ∈ {0,1}** be the ground truth label (1 = pothole, 0 = no pothole).

Convolution Operation:
Feature extraction in the CNN is carried out using convolutional layers. Each feature map is computed as:
Fi,j,k=σ(∑m=0M−1∑n=0N−1Ii+m,j+n·Wm,n,k+bk)F_{i,j,k} = \sigma \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{i+m,j+n} \cdot W_{m,n,k} + b_k \right)Fi,j,k=σ(m=0∑M−1n=0∑N−1Ii+m,j+n·Wm,n,k+bk)
where:

- **I** is the input matrix (image pixels),
- **W** is the convolution kernel,
- **b$_k$** is the bias for filter **k**, and
- **σ** represents the ReLU activation function.

Pooling:
To reduce spatial dimensions while retaining important features, max-pooling is applied:

Pi,j=max(m,n)∈RFi+m,j+nP_{i,j} = \max_{(m,n) \in R} F_{i+m, j+n}Pi,j=(m,n)∈RmaxFi+m,j+n

This ensures only the most dominant features are preserved, improving efficiency and reducing overfitting.

Fully Connected Layer:
The extracted features are flattened and passed to fully connected layers, represented as:

z=Wf·x+bfz = W_f \cdot x + b_fz=Wf·x+bf

where **x** is the flattened feature vector, **Wf** is the weight matrix, and **bf** is the bias term.

Sigmoid Activation for Binary Output:
A sigmoid activation function converts the final output into a probability score between 0 and 1:

p=σ(z)=11+e−zp = \sigma(z) = \frac{1}{1 + e^{-z}}p=σ(z)=1+e−z1

Here, **p** represents the model's confidence that the input image contains a pothole.

Loss Function:
The model is optimized using the Binary Cross-Entropy (BCE) loss, defined as:

L=−1N∑i=1N[yilog⁡(pi)+(1−yi)log⁡(1−pi)]\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N \left[y_i \log(p_i) + (1 - y_i) \log(1 - p_i) \right]L=−N1i=1∑N[yilog(pi)+(1−yi)log(1−pi)]

This function penalizes incorrect predictions more heavily when the model is highly confident, ensuring more accurate classification.

VI. IMPLEMENTATION

The training script (train_model.py) is responsible for building and training the CNN. It uses **Keras' ImageDataGenerator** (or alternatively, TensorFlow's tf.data pipelines) for efficient data loading and augmentation. The CNN is constructed, trained with validation data, and the best-performing model is saved either in **HDF5** format or as a **native Keras file** for later use.

The deployment side of the system is handled by a **Flask application** (app.py). When the server starts, the trained model is loaded into memory. The backend provides:

- A **login route** for user access.
- An **upload route** where users can submit road images.
- A **predict function** that:
    - Preprocesses uploaded images (resizing and normalization).
    - Calls model.predict() to obtain the probability of a pothole.
    - Converts the probability into a confidence score and a severity label.
    - Optionally annotates the image with a bounding box using **OpenCV**.
    - Generates a **severity bar chart** using **matplotlib**.

Deployment and Packaging
The application is bundled within a **virtual environment**, with a requirements.txt file included to ensure that all dependencies can be easily reproduced on any system.
For more advanced deployment scenarios, the system offers several options:

- The trained model can be exported in **TensorFlow's Saved Model format**, making it easier to reuse or integrate with other platforms.
- The entire application can be **containerized with Docker**, allowing seamless portability across different environments.
- A lightweight **SQLite database** can be integrated to store previous predictions and generated reports, providing persistence for long-term usage.

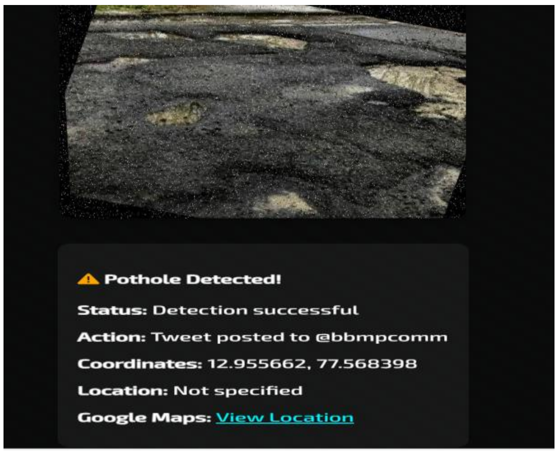
VII. RESULT AND DISCUSSION

The classifier was trained over multiple epochs, with validation accuracy and loss monitored to track performance. Standard evaluation metrics such as **accuracy, precision, recall, F1-score**, and, in the case of object detection, **mean Average Precision (mAP)** are typically used to assess model quality.

In practice, the model achieved strong training and validation accuracy when the dataset was well-curated and augmented. However, very high training accuracy (close to 100%) may signal **overfitting**. To mitigate this, techniques such as dropout, extensive data augmentation, and expanding the dataset are effective countermeasures.

The current **severity mapping** is based on the model's confidence score. For uncertain cases, the confidence value reflects this uncertainty, often resulting in a "Minor" or "None" severity label. For more precise severity estimation—especially in terms of pothole size—an object detection framework like **YOLOv8** could be integrated to generate bounding boxes and approximate the area of the detected pothole.

The **web interface** functions as intended, displaying predictions alongside location details and uploaded images. However, if multiple uploaded images consistently return identical severity or confidence values, this often points to **overfitting** or inconsistencies in image preprocessing. Ensuring that the preprocessing pipeline (e.g., resizing and normalization) is consistent between training and inference helps resolve these issues.



⚠ Pothole Detected!

Status: Detection successful

Action: Tweet posted to @bbmpcomm

Coordinates: 12.955662, 77.568398

Location: Not specified

Google Maps: View Location

Fig1: Detection Result

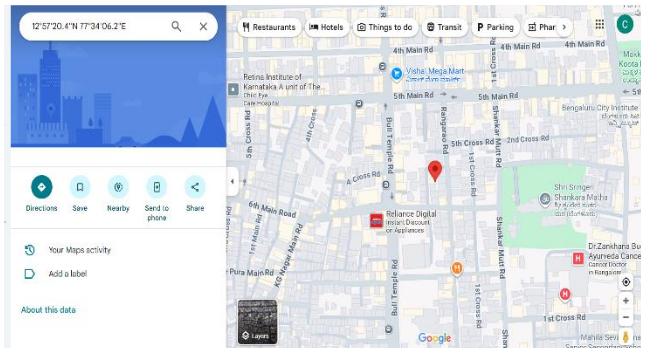


Fig2: Location of Pothole

VIII. CONCLUSION

This project showcases a complete, end-to-end pothole detection pipeline, covering everything from dataset preparation and CNN training to web deployment and automated PDF reporting. The CNN classifier offers a simple yet effective way to quickly distinguish between pothole and non-pothole images, while the confidence-based severity estimate provides a useful starting point for prioritizing road repairs.

The Flask-based web application makes the system accessible even to non-technical users, and the built-in reporting feature supports administrative and record-keeping needs. Looking ahead, the system can be further strengthened by integrating **GPS data** for precise location tracking, adopting **object detection models** for better localization and size estimation, and expanding the dataset to cover a wider variety of road surfaces, environments, and lighting conditions. These enhancements would improve both the accuracy and the practical utility of the system, making it a valuable tool for municipal road maintenance and urban infrastructure management.

REFERENCES

[1]. Y. Arya, A. Kumar, and P. Singh, "Automated pothole detection using deep learning," *IEEE Access*, vol. 8, pp. 147011–147022, 2020.

[2]. R. Kumar and P. Singh, "A vision-based system for road surface inspection," *International Journal of Computer Applications*, vol. 175, no. 23, pp. 23–30, 2020.

[3]. S. Li, M. Wang, and X. Chen, "Pothole detection based on YOLOv3," in *Proc. IEEE Intelligent Transportation Systems Conf.*, 2019, pp. 3034–3039.

[4]. A. Jain and S. Agrawal, "Convolutional neural networks for road defect detection," *Journal of Transportation Engineering*, vol. 146, no. 4, pp. 04020012, 2020.

[5]. M. Zhang, H. Lee, and T. Kim, "Image-based road surface monitoring," *Sensors*, vol. 19, no. 14, pp. 3075, 2019.

[6]. D. Gupta et al., "Road damage detection using deep learning," in *Proc. IEEE CVPR Workshops*, 2020, pp. 368–376.

DOI: 10.17148/IARJSET.2025.12829

[7]. K. Yamada, "Automated road anomaly detection using mobile vision," *Transportation Research Record*, vol. 2672, no. 3, pp. 40–50, 2018.

[8]. S. Patil and V. Kulkarni, "Real-time pothole detection using machine learning and IoT," *Procedia Computer Science*, vol. 171, pp. 1328–1337, 2020.

[9]. H. Liu, "A comparative study of CNN architectures for pothole detection," *IEEE Trans. Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3765–3775, 2021.

[10]. M. Banerjee and R. Dutta, "Image classification for smart road maintenance," *Applied Soft Computing*, vol. 101, p. 107056, 2021.