iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79585

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

YOLOv8: An Integrated Framework for Automated Road Damage Quantification and Prioritization

Ram Krishna Verma[1], Archit Patle[2], Abhijit Routray[3], Aalaukik Dole[4], Ashwini Bajanghate[5], Anushka Gajbhiye[6]
[1]*Assistant Professor,* [2,3,4,5,6]*Student, Department of Computer Science and Engineering GH Raisoni Skill Tech University, Nagpur, Maharashtra, India*

Abstract: ***The bad state of roads is still a big problem in India, causing accidents, damaging vehicles, and creating delays that impact everyday life and the economy. One of the main problems that officials face is not having up-to-date and specific information about where repairs are needed, which makes it hard to decide which areas to fix first. Most current complaint systems use only text for input, don't have good ways to check if the information is correct, and don't give much helpful information to the public. To fix these issues, this project presents Street Info Hub, a web platform designed for citizens. It lets users report damaged roads by sharing pictures and providing location details through GPS. The main part of the system is a combined artificial intelligence process. A fine-tuned MobileNetV3 Small model first checks whether the uploaded image contains a road surface. Once validated, a YOLOv8n model trained to detect four types of road damage identifies and marks defects. The severity of each issue is calculated based on damage type, bounding-box area, and image dimensions. All verified reports are stored in a cloud-based MongoDB database along with detailed metadata. The platform also includes tools for analyzing damage patterns and supports route planning by displaying reported potholes on navigation paths using OpenRouteService and Leaflet maps. A dedicated dashboard enables users to monitor their submissions. After training, the YOLOv8n model achieved a mAP@50 score of 46.7% and a precision of 54.7%, showing reliable performance. Overall, the system demonstrates how citizen participation and computer vision can provide real-time data for faster and better road maintenance decisions in practical real-world deployment scenarios today.***
Keywords: Road Damage Detection, YOLOv8, MobileNetV3, Citizen Reporting, Route Planner.

I. INTRODUCTION

In India a lot of the roads are in poor condition because of problems like potholes, cracks, and grooves. These issues are usually not seen or fixed for a long period. Broken roads can damage cars, make them use more gas, and make trips take more time. The real issue isn't that people don't want to fix the roads, but that there's no timely, dependable, and specific information about where the roads are in need of repair. Traditional ways rely on engineers checking things often, but this method takes a lot of time and uses many resources. When people report problems, they often only write words and don't add images, specific places, or details about how bad the problem is. This makes it more difficult to use maintenance resources correctly. To fix this issue, Street Info Hub was created as a platform that puts citizens first. Users can report damaged roads by sending pictures and marking exact locations. The system is already working as a web app available at https://streetinfohub.cloud, allowing people to interact with it in real time, check the reports, and see how it works in real life. The platform uses an AI system that finds road surfaces, spots and sorts out different types of damage, and creates a rating showing how bad the damage is. This rating is then kept in a main database.

The project is centered around three key areas: reporting, verification, and raising awareness. It makes it easier for more people to report problems by providing simple tools, uses computer vision to do automatic checks, and gives helpful information to the public through analysis and maps that show dangerous areas along different routes. The system is built as a modular web platform that includes user login, reporting with location details, automatic analysis of damage, tools to display data visually, and planning routes with maps showing dangerous areas. At the moment, the idea is to develop a web version for people living in cities and towns. We might also create a mobile app in the future and link it to government services.

II. LITERATURE SURVEY

A. Overview

In the last few years, there has been a noticeable change in how road damage is detected and managed. This is mainly because of the use of deep learning along with web, mobile, and mapping technologies. Most of the work in this area is focused on three things: using models like YOLO and CNNs to detect road issues, building platforms where users can report problems themselves, and using geospatial tools to map and study these issues.Artificial intelligence has helped reduce the need for manual checking. Instead of relying completely on human inspection, systems can now identify damage automatically, which saves time and helps in faster response. At the same time, public reporting platforms have made it easier for people to share information about road conditions in their area. Because of these issues, recent studies from 2022 to 2024 are being considered to design a system that works better in terms of performance and integration.

B. Detailed Description of Existing Systems

Different methods have been looked into for finding road damage with the help of deep learning. Shim et al. Suggested a way to make detection more accurate by improving image quality through super-resolution methods along with semi-supervised learning. Although this method is very accurate, it needs a lot of computing power, which makes it hard to use in real-time situations [4]. Pagar et al. Suggested a mobile app that uses a CNN model to detect simple potholes, allows reporting via GPS, and includes checks to make sure the input is correct. The system can only classify things into two categories and doesn't offer more detailed analysis or the ability to detect more than two types [1]. Adi et al. created iRodd, a system that uses YOLOv5 along with data collected from smartphones and LiDAR for 3D measurements. The model performs well in detection, but the system is mostly made for government use and doesn't have a user-friendly platform for regular people [2]. Dasari et al. Created a system using YOLOv8 that can detect damage in real time and includes a web-based interface for access. The system works well at finding defects, but it doesn't completely include advanced features like route-based hazard awareness [3]. However, most current research doesn't bring together AI-powered detection and real-time route visuals into one easy-to-use platform.

C. Gap Identification and Comparative Analysis

Even though current systems have experienced considerable improvement, some issues remain unresolved. While many methods focus on their ability to detect something or count on users to report issues, not all combine both ideas into a single system that works together. High-performance models typically require a significant amount of processing power, which can hinder their usefulness in real-time applications. Although there are studies that emphasize the use of routes to detect hazards, only a few people employ this method in real-life scenarios. Current systems are compared in Table 1:

Table 1Comparative Analysis of Existing Road Damage Detection Systems

Study	Machine LearningApproach	Citizen Reporting	Image Validation Gate	Public Analytics	Route-Aware Hazard Map
Shim et al. (2022)	SRGAN + Semi-supervised segmentation (mIoU 81.5%)	No	No	No	No
Dasari et al. (2024)	YOLOv8n, 7000+ images, 1280px resolution	Partial (web upload, webcam)	No	No	No (mentioned as future work)
Adi et al. (2024)	YOLOv5, 94% precision, 85% recall; LiDAR for 3D volume	Partial (field video, government data)	No	No	No
Pagar et al. (2024)	Fully Connected CNN, binary (pothole/normal)	Yes (Flutter app + GPS geo-tag)	Yes (non-related image rejected)	No	No

The table shows that current solutions do not fully cover efficient AI models, citizen participation, methods for verifying accuracy, and the use of geospatial data. The Street Info Hub deals with these problems by using a two-step machine learning method, along with real-time updates, awareness of dangers based on routes, and analysis that depends on location, providing a more complete and useful way to track infrastructure.

III. PROPOSED WORK

A. *Architecture Overview*

The system architecture of RoadDamage Detection is divided into two different stages:

1) *Stage One (MobileNetV3):* In stage one the MobileNetV3, scans the image to classify if there is road or not. If there is road then only the image is forwarded to Stage Two.
2) *Stage Two (YOLOv8):* If Stage One detects the road in image, then theimage is sent to YOLOv8. This modelcreates bounding boxes around damages in road and then classifies the damages into specific categories, such as longitudinal cracks and potholes.

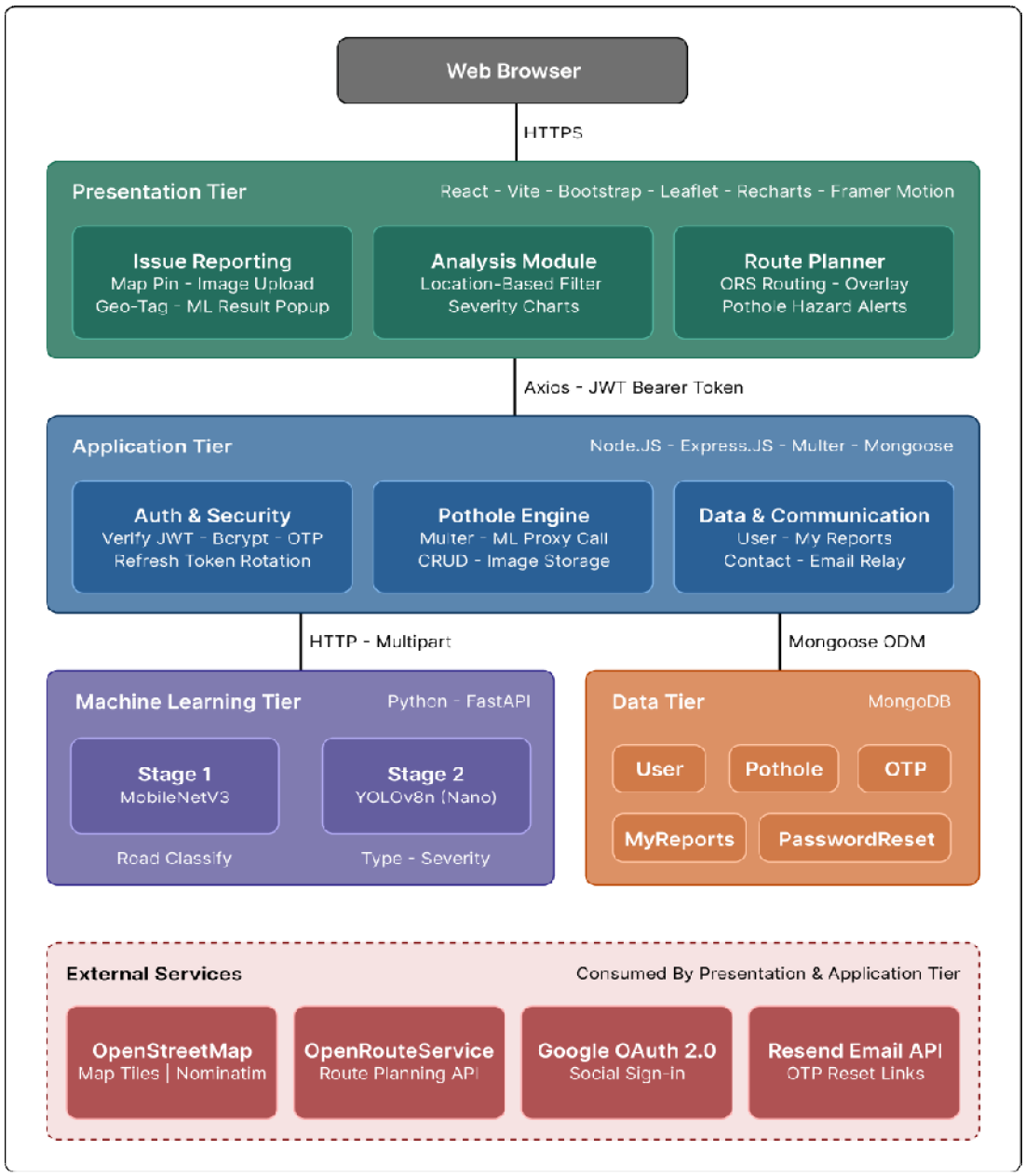


Figure 1System Architecture of Street Info Hub

B. Web Microservice Integration

The Street Info Hub system uses a microservice-based design, which makes it easy to manage, grow, and keep the different parts working well together. It has three main parts: a React frontend that lets users interact with the app, a Node.js backend that takes care of the app's logic and connects to APIs, and a FastAPI Python microservice that handles machine learning tasks.

These parts talk to each other using HTTP, which lets each one be built and put into use on its own. The process starts when someone sends in a report about road damage by sharing a photo and picking a place. The frontend sends this data as a multipart/form-data request to the backend, along with a JSON Web Token (JWT) to verify the user's identity. The system checks the token, keeps the image for a short time, and sends it to the machine learning service. The microservice uses a two-step process to handle the image. First, MobileNetV3 checks if there's a road in the image. Then, YOLOv8 is used to find and identify any damage in more detail. If a road isn't found, the request is turned down to prevent extra work from being done. Otherwise, the system gives back a clear answer that lists the types of damage found, how sure it is about each type, and how serious the damage is. The system saves the checked data into the database along with information about the user and their location, and the results are sent back to the front end at the same time so they can be shown on the screen in real time. This design lets different parts of the system be updated separately, and secure login and organized API communication help keep data sharing dependable and safe.

IV. METHODS AND IMPLEMENTATION

A. Dataset Preparation and Preprocessing

The models were trained using a large dataset consisting of Forty-Seven Thousand annotated images of road damages were used under different lighting and weather conditions. Data preprocessing involved adjusting the pixel values to a common range to help the neural networks train more effectively and efficiently. The areas marked by the bounding boxes were converted into the YAML format needed for the YOLOv8 model.

B. Stage One: MobileNetV3 Implementation

MobileNetV3 was chosen for the initial classification phase of roads and non-roads images, due to its efficient use of depth wise separable convolutions and squeeze-and-excitation modules. This feature allows it to do fast binary classification, like telling if the road is damaged or not. It stays in good condition with a small amount of memory used, which makes it perfect for starting a web API that needs to handle a lot of traffic.

C. Stage Two: YOLOv8 Implementation

Once the images are confirmed to have damage, YOLOv8 is used to find the type and severity of the damage.The backbone of the modelefficiently extracts features and minimizes computational costs.The neck of the modelcombines features from different sizes, so the system can spot both big potholes and small alligator cracks.The main head estimates the precise location of objects with bounding boxes, along with confidence levels and probabilities for eachclass.

D. Deployment Strategy

The system suggests a three-layer setup that spreads out across different parts. It uses React for the front part where users interact, Node.js for the back part that handles requests, and a Python service built with FastAPI that focuses on machine learning tasks. These parts talk to each other using HTTP, which lets them work on their own, grow bigger as needed, and be simpler to fix. During development, services operate on the same machine, but in production, they can be packaged into containers and placed on a shared network, with MongoDB used as the cloud-based database. The frontend manages how users interact with the app, using React along with other related tools for moving between pages, showing data visually, and displaying maps. It uses React Context to handle login and authentication, keeps tokens safe, and remembers your login using cookies that can't be accessed through regular web scripts. Users can report road damage by filling out an interactive form that allows them to upload images and select a location.

The backend, which uses Express.js, handles routing, user login, and connecting to the machine learning service. It also ensures security by using CORS rules, bcrypt to protect passwords, rotating tokens, and managing cookies securely. Uploaded images are handled by Multer and checked for correctness. The backend sends images to the FastAPI microservice, where a two-step process is carried out. MobileNetV3 checks if there is a road, and YOLOv8n identifies and categorizes any damage. Invalid inputs are checked first and are not accepted, but when the input is correct, the results like bounding boxes, confidence level, and severity are given back in a JSON format. If the ML service isn't working, the system handles it smoothly by saving reports without checking them.

The data that has been processed is kept in MongoDB, connecting both the global reports and each user's own history, and the results are shown on the front part of the website.

The system works on its own without waiting for other parts to finish, which makes it faster and more efficient. It also has features like detecting dangers based on routes using analysis of geographical data. The design makes sure communication is safe, resources are used well, and the system can handle more users as needed for monitoring road damage in real time.

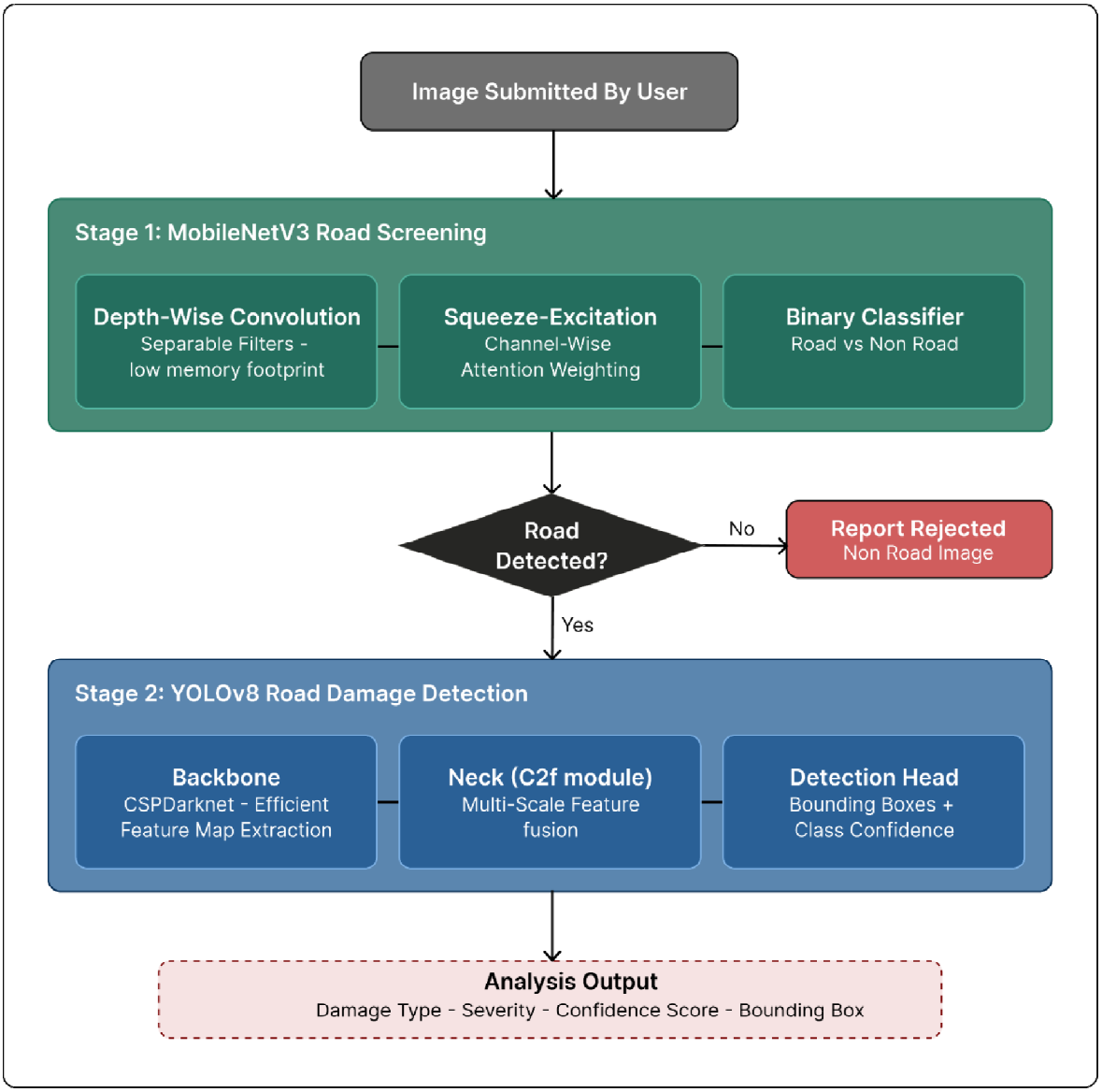


Figure2 ML Microservice Flowchart

To make the system more dependable, the different parts of the system were connected together by using correct APIs and JSON-based data sharing. One of the benefits is that you can update any part of the system, such as the frontend, backend or the machine learning part, without affecting the rest of the system. It is easy to figure out what is not working and fix it easily. The system also has simple ways to deal with mistakes and backup plans if anything goes wrong or stops working. Because of this, the system can still work well in most cases, which makes it more stable and effective.

V. RESULTS AND DISCUSSIONS

The implementation of the proposed system - Street Info Hub shows that efficient and scalable road damage detection can be achieved using a two-stage deep learning architecture combined with a web-based microservice system. Users can upload images along with geo-location coordinates.

The uploaded image then goes through the pipeline and is then the result is shown to the user in the form of pop-up. The report submission interface provides a form with image upload, location selectionand input fields for detailed issue reporting, as shown in Figure 3.

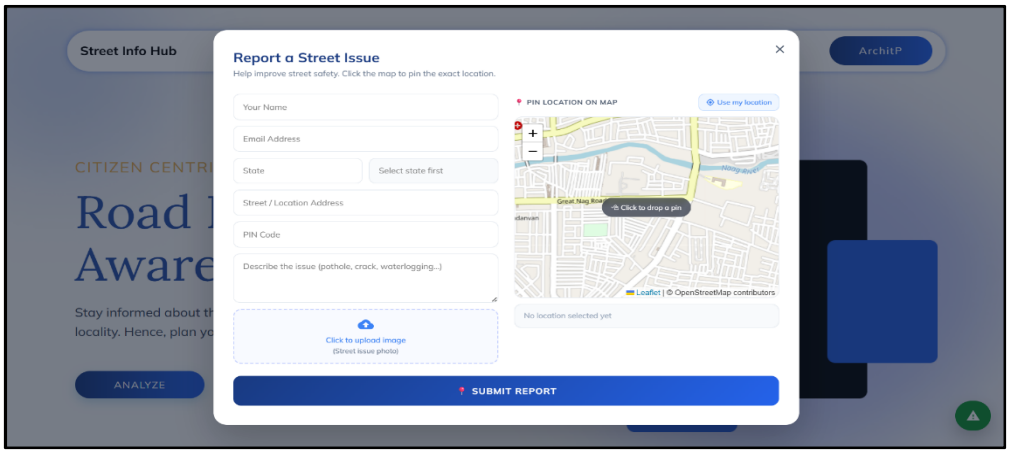


Figure 3 Report Issue Submission Interface

After submission, the system processes the input and gives the analyzed output to the user as shown in Figure 4.

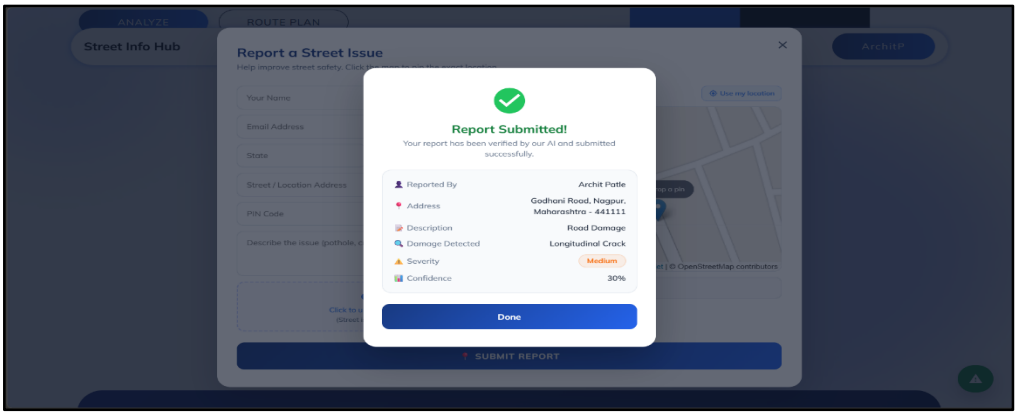


Figure 4 ML Analysis Result Display

The route planner visualizes routes along with highlighted road damages, enabling users to identify high-risk areas in real time, as shown in Figure 5.

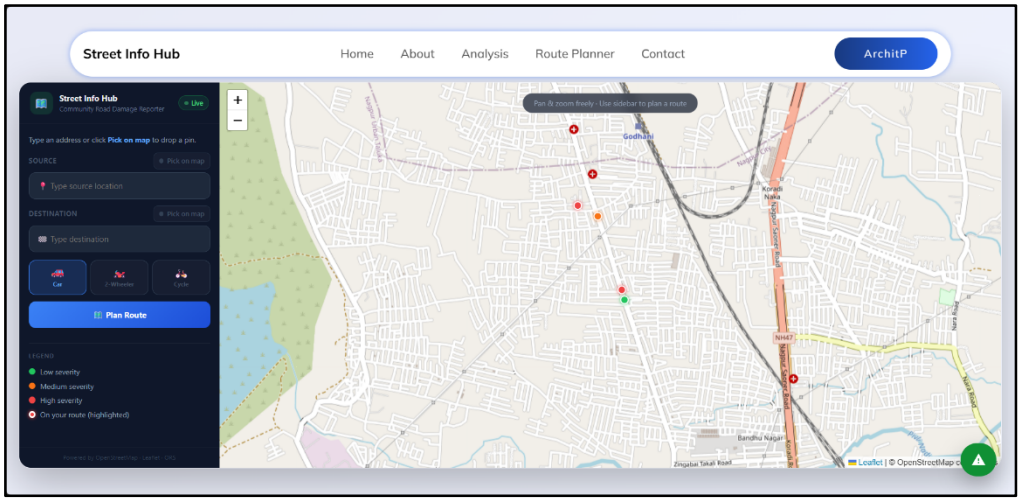


Figure 5 Route Planner with Hazard Overlay

To support data-driven insights, the platform includes graphical representations such as pie charts and bar graphs representing damage distribution and severity levels, as shown in Figure 6.

Figure 6 Analytical Dashboard Showing Damage Distribution

From a performance point of view, the two-stage pipeline greatly boosts how efficiently the computer can process tasks. MobileNetV3 is used first to process the inputs, which helps reduce the amount of data that YOLOv8 has to handle. This makes the system work more efficiently on the CPU and GPU, and it still keeps the detection results accurate. YOLOv8 reached a base mAP@50 score of 46.7%, with good balance between precision and recall, showing it can detect damage well while keeping false positives to a minimum. However, certain limitations were observed. The model's performance was affected because of things like not enough data, lower quality images, and some classes having more examples than others. These issues made it harder to detect small cracks. Even though this is the case, the accuracy we've achieved is enough for a system that focuses on helping citizens by tagging and prioritizing road problems.

The MobileNetV3 validation step helped make things more reliable by stopping non-road images from getting into the process. In terms of system performance, the time it took to submit the full report was on average between 8 to 12 seconds, mostly because of the machine learning inference process, while the ML service itself responded in 2 to 4 seconds. The communication between the frontend and backend was not very active, and the route planner provided a response in 1 to 2 seconds because of good processing on the client side for geospatial data. The system kept working smoothly even when many requests came in at the same time, showing it can manage real-world situations well. The modular design lets you improve each part on its own without messing up how the whole system works, which makes the system faster and more efficient. Moreover, handling requests efficiently and using asynchronous processing help keep user interactions smooth during busy times, and the system keeps providing reliable and consistent results no matter the conditions. The system shows a useful, flexible, and effective way to monitor road damage in real time and support smart city projects.

VI. CONCLUSION AND FUTURE SCOPE

This project presents Street Info Hub, a web platform focused on helping citizens report road issues. It uses modern web tools along with deep learning techniques and is built on a scalable microservice architecture. The system has a React front end, a Node.js back end, and an ML service made with FastAPI, which lets users send reports in real time, check their accuracy, and view them visually. A big part of this work involves a two-step detection process. First, MobileNetV3 is used for a quick check, and then YOLOv8n is used to detect any damage. This method makes computing faster without lowering performance, and it works well to block out non-road data, correctly finding different kinds of damage and judging how serious each is. The platform also has safe login methods and offers tools such as analytics and visual maps that highlight dangerous areas on routes, making it simpler to use. The system shows that using automated methods led by citizens to check on infrastructure works really well, saves a lot of time, and can get bigger as more people join, making it a strong beginning for smart city projects.

Future efforts will focus on improving the system's accuracy, its ability to handle larger tasks, and its overall performance. Making the dataset better and training more detailed YOLO models can help improve how well they detect objects.

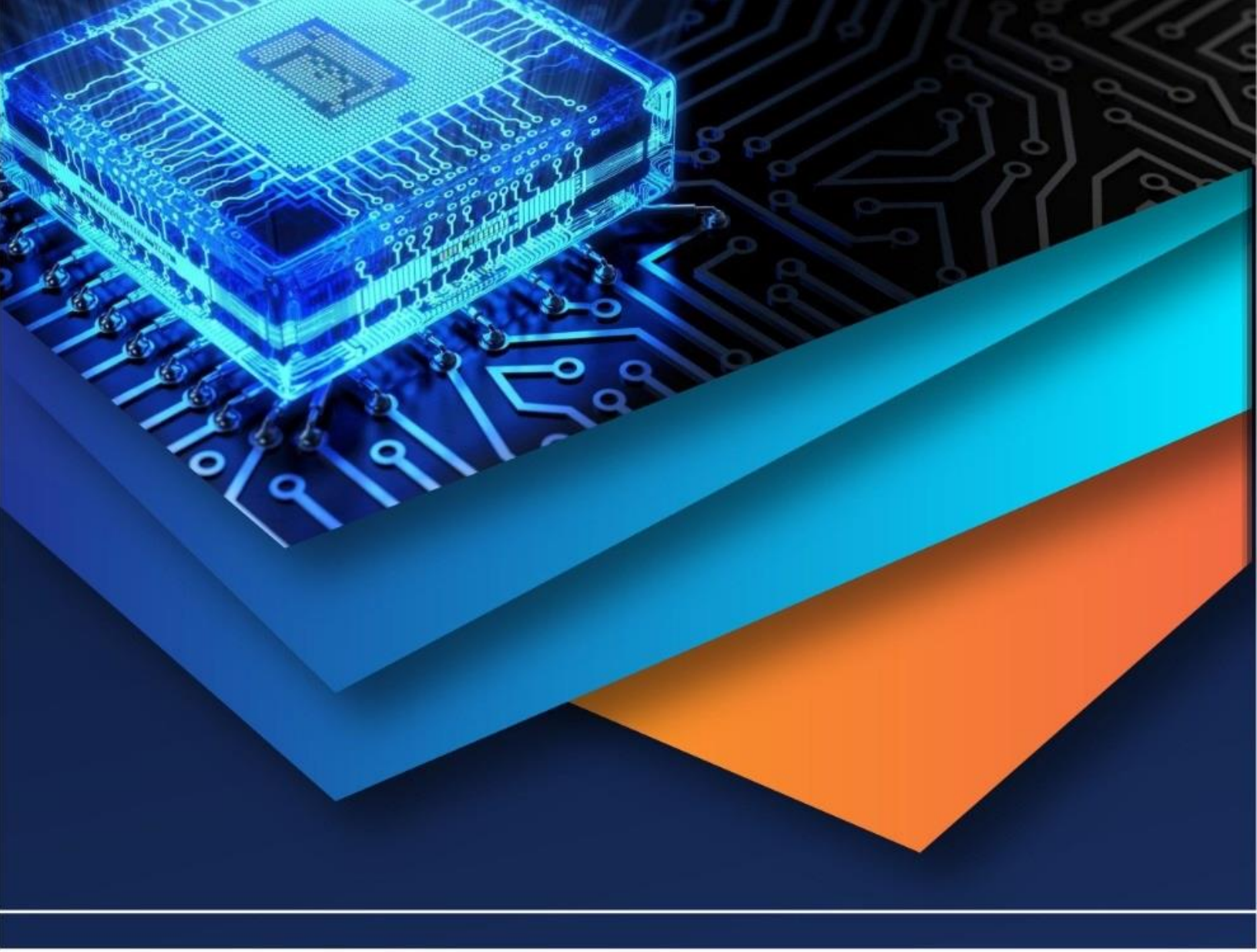
Building an administrative dashboard for officials and a mobile app can make the system more helpful and simpler to use. A big improvement is using LiDAR data along with RGB images to help analyze road damage in three dimensions. This would let us measure depth, volume, and how bad the structure is, which helps in getting the right amount of materials and doing a better check on the infrastructure. Other features such as deleting duplicate reports, sending alerts, and looking at data over time can help make the platform even more useful. This makes it a complete and intelligent system for managing large infrastructure projects.

VII. ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the Department of Computer Science and Engineering at G. H. Raisoni University, Amravati, for providing the necessary support to carry out this project. We would also like to thank our faculty mentors for their valuable guidance, encouragementand continuous feedback throughout the development of this project. Special appreciation is extended to all contributors and peers who provided insights and suggestions that helped improve the system. Finally, we would like to acknowledge the use of open-source tools, libraries and datasets that played a crucial role in the successful implementation of this project.

REFERENCES

[1] Nilima Pagar et al., Road Damage Detection and Reporting System Using Fully Connected CNN, International Advanced Research Journal in Science, Engineering and Technology (IARJSET), Vol. 11, Issue 4, April 2024.
[2] Adi, T. J. W., Suprobo, P., &Waliulu, Y. E. P. R. (2024). iRodd (intelligent-road damage detection) for real-time infrastructure preservation in detection, classification, calculation, and visualization. Journal of Infrastructure, Policy and Development, 8(11), 6162.
[3] Dasari, D., Vijaya Lakshmi, D. N. V. S., & Ashok, T. (2024). Real-time detection of road damages using YOLOv8: An innovative deep learning approach. 11(12), h459–h465. ISSN: 2349-5162Journal of Emerging Technologies and Innovative Research (JETIR).
[4] Shim, S., Kim, J., Lee, S.-W., & Cho, G.-C. (2022). Road damage detection using super-resolution and semi-supervised learning with generative adversarial network. Automation in Construction, 135, 104139.
[5] React Documentation. Available: https://react.dev/
[6] Node.js Documentation. Available: https://nodejs.org/
[7] FastAPI Documentation. Available: https://fastapi.tiangolo.com/
[8] MongoDB Inc., “MongoDB Documentation,” Available: https://www.mongodb.com/docs/
[9] OpenRouteService API Documentation. Available: https://openrouteservice.org/
[10] Leaflet JavaScript Library. Available: https://leafletjs.com/
[11] OpenStreetMap Contributors, “OpenStreetMap: A Collaborative Project to Create a Free Editable Map,” Available: https://www.openstreetmap.org



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY