Real-Time Pothole Detection Using Convolutional Neural Networks and YOLOv8

T. Shiva Priya[1], Dr. K. Suresh Babu[2]

[1]M.Tech Student, Department of Computer Science and Engineering, JNTUH College of Engineering, Hyderabad, Telangana, India

[2]Professor, Training & Placement Officer(TPO) & PTPG Coordinator, Department of Computer Science and Engineering, JNTUH College of Engineering, Hyderabad, Telangana, India

Abstract

Potholes on roads are a major threat to vehicle safety and infrastructure. This project proposes a two-stage deep learning pipeline for real-time pothole detection. A custom dataset was created with 500 plain road images and 2,000 pothole road images and all pothole road images are annotated in YOLO format. In the first stage, a CNN based on ResNet50 is used to classify images as either "Plain road image" or "Pothole road image". Images classified as pothole road images are then passed to a YOLOV8n model for accurate localization using bounding boxes. This combined method helps reduce false positives and improves computational efficiency. The ResNet50 classification model achieved an accuracy of 95%, while YOLOv8n had a mAP@0.5 of 86.53%, mAP@0.5:0.95 of 59.68%, precision of 88.77% and recall of 75.96%. A Gradio-based user interface is created to provide an interactive experience, allowing users to choose between single or bulk image detection modes, visualize detection results, and download outputs. These results show strong real-time performance, making the system suitable for smart vehicles and road monitoring systems.

Keywords: Potholes, ResNet50, YOLOv8n, Convolutional Neural Network

1. Introduction

Potholes are one of the most common and dangerous road surface problems. They lead to accidents, traffic jams, and increased vehicle maintenance costs. Potholes present a significant challenge to road safety, especially in developing countries where infrastructure monitoring is weak. Timely identification and repair of potholes are essential to maintain road quality and prevent damage to vehicles and injuries to commuters. Traditional methods for detecting potholes, like manual inspections or vibration-based sensors in vehicles, are either labor-intensive Recent developments in computer vision and deep learning, time-consuming, or inconsistent in accuracy. With the increasing demand for smart transportation systems, there is a growing need for automated, accurate, and real-time pothole detection solutions. Recent developments in computer vision and deep learning have enabled image-based detection systems that can analyze road conditions using captured images or video streams. While object detection models like YOLO have shown promising results, applying them to every frame can be computationally expensive, especially in real time applications. To overcome these challenges, this project proposes a hybrid deep learning pipeline that combines image classification and object detection. A Convolutional Neural Network (CNN) based on ResNet50 is first used to classify images as either "Plain road image" or "Pothole road image." Only the pothole-classified images are then passed to YOLOv8n for accurate detection and localization of potholes using bounding boxes. This two-stage approach improves detection efficiency, reduces unnecessary computations, and maintains high accuracy, making it well-suited for use in smart vehicles and automated road monitoring systems.

2. CNN & YOLO

The CNN & YOLO proposed method uses a two stage deep learning pipeline for efficient and accurate pothole detection. This method combines the strengths of a Convolutional Neural Network (CNN) for classifying images and YOLOv8 for detecting objects, taking advantage of the strengths of both models.

2.1 CNN Classification Using ResNet50

In the first stage, a CNN model based on ResNet50 is used to classify road images into two categories: “Plain road” and “Pothole road”. ResNet50 uses residual learning blocks, which helps training of deep networks by overcoming the vanishing gradient problem. It effectively extracts important spatial features from road images, allowing for accurate classification. This stage acts as an intelligent pre-filter. It helps the system to discard non-relevant images and reduce unnecessary detection work.

2.2 YOLOv8n Object Detection

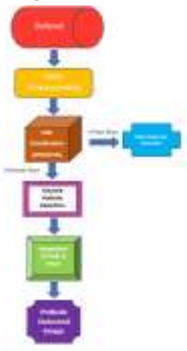
In the second stage, only the images identified as pothole-containing are passed to the YOLOv8n model, a lightweight version of the YOLOv8 object detection family designed for real-time applications. YOLOv8n locates potholes by generating bounding boxes with associated confidence scores. It can detect potholes of different sizes and shapes in various lighting and environmental conditions. Non-Maximum Suppression (NMS) is applied to refine overlapping predictions and improve detection accuracy.

2.3 CNN & YOLO Integration

The CNN & YOLO model has several benefits. It improves efficiency by skipping detection on plain road images, which reduces the computational load and allows the system to work on devices with limited resources. It also increases accuracy by minimizing false positives, particularly when road textures or shadows look like potholes. Its modularity allows independent updates to either the classification or detection stage without redesigning the full system. Finally, it provides strong scalability, making it useful for large-scale road monitoring, autonomous vehicles, and smart traffic systems. The system is evaluated using classification accuracy for the ResNet50 model and mean Average Precision (mAP), precision, and recall for the YOLOv8n detector. The evaluation shows the effectiveness and real-time potential of the combined approach.

3. Block Diagram Of CNN & YOLO

Figure 1: Block Diagram Of CNN & YOLOV8



4. Implementation

4.1 Dataset Details

A custom dataset was created with 2,500 road images, which included both plain road and pothole road images. From this, there are 2,000 annotated pothole images and 500 plain road images was used for training and testing. The pothole images have YOLO-format bounding box annotations, which were directly used for object detection training.

4.2 Data Preprocessing

All images were resized and normalized before being input into the models. For the CNN classifier (ResNet50), images were resized to 224×224 pixels and normalized to the [0, 1] range. For YOLOv8n, images were resized to 640×640 pixels to match model requirements. A preprocessing function is applied to each image, and the data is organized into two classes: plain road images were labelled as 0 and pothole road images as 1. The script processed 500 plain road images and the 2,000 pothole images, applying the preprocessing steps and store them in separate lists. We then convert these lists into NumPy arrays to make model training easier. The dataset is divided into training and testing sets using an 80:20 ratio with the train_test_split() function, and shuffling is enabled. This ensured a balanced and randomized distribution of both classes in the train and test sets, providing a solid foundation for training the ResNet50 classification model.

4.3 CNN Classification

The first stage of the pipeline uses a binary image classifier based on the ResNet50 architecture. This deep convolutional neural network features residual connections that allow for efficient training of very deep models. The goal of this model is to classify each road image as either a “plain road” or a “pothole road.” Before training, the class labels (0 for plain and 1 for pothole) were one-hot encoded with the to_categorical() function. This conversion turns them into two-element vectors that are suitable for categorical classification. A transfer learning method is used to take advantage of the pre-trained ResNet50 model with ImageNet weights. The top fully-connected layer of ResNet50 is removed by setting include_top=False. Then a custom classification head is added. This new head included a global average pooling layer, a dropout layer to reduce overfitting, a dense layer with 64 neurons and ReLU activation, and a final output layer with two neurons and softmax activation for binary classification. The model is compiled using the Adam optimizer with a learning rate of 1e-4, categorical cross entropy loss (since output was one-hot encoded), and accuracy as the evaluation metric. The network is trained on 80% of the dataset with the remaining 20% for validation. Training is carried out over 15 epochs with a batch size of 32. The training and validation accuracy were monitored using the history object returned by the model.fit() function. After training, the ResNet50-based classifier achieved a high accuracy of 95%. This makes it an effective first-stage filter for pothole detection. By identifying and discarding plain road images early in the process, the CNN classifier significantly reduce the load on the following YOLO detection model.

4.4 YOLOV8 Detection

The second stage of the pipeline is the object detection phase, the YOLOv8n model is chosen because of its lightweight design and ability to work in real-time. The training dataset included 2,000 annotated pothole road images, each with a matching label file in YOLO format. Next, the dataset is divided into 80% for training and 20% for validation. This is done programmatically by shuffling the image list and moving files into the correct train and validation subfolders for both images and labels. This split is important for assessing the model's ability to generalize during training. A custom data.yaml file is created to define the dataset structure for the YOLOv8 training pipeline. It specified the train and validation image paths, the number of classes (nc: 1), and the class name (pothole). With the data prepared, the YOLOv8n model is initialized using its configuration file (yolov8n.yaml) and trained using the Ultralytics framework. Training was conducted for 100 epochs with an image size of 640×640 pixels and a batch size of 16. The model learned to detect potholes using bounding boxes. After training, the model achieved a mAP@0.5 of 86.53%, mAP@0.5:0.95 of 59.68%, precision of 88.77%, and recall of 75.96%. These results show strong detection performance for real time pothole localization tasks.

4.5 Integration

To enable real-time decision-making, the two-stage classification and detection models were integrated into one pipeline. The process starts by passing an input image through the trained ResNet50 CNN classifier. This helps determine if the image shows a pothole or a plain road. If the classifier predicts a pothole, then pass the image to the YOLOv8n object detection model for precise localization. This conditional routing reduces unnecessary computational work by stopping the detection model from running on images that do not contain potholes. The integration is implemented in Python using TensorFlow/Keras and the Ultralytics YOLOv8 library. A custom function is created to preprocess the input image by resizing it to 224×224 pixels and normalizing its pixel values before classification. Once classified, pothole images were passed to YOLOv8n, which returns bounding box coordinates, confidence scores, and class labels. These results were visualized using OpenCV and Matplotlib, with potholes highlighted using rectangles and labelled with their detection confidence. The system supports both inline visualization (e.g., in Colab or Kaggle) and pop-up window display for desktop applications. This pipeline combines the strengths of both models. The CNN provides fast binary classification, while YOLOv8n ensures high precision bounding box predictions. The visual output, along with printed predictions and confidence levels, allows for transparent debugging and performance evaluation. By avoiding unnecessary detections and only analysing relevant images, the system finds a balance between accuracy and real-time efficiency, making it ideal for use in smart vehicles or road monitoring systems.

4.6 Gradio Interface

To make the proposed system interactive and easy to use, a front-end interface is developed with the Gradio Python library. Gradio allows for quick prototyping and deployment of machine learning models through a straightforward graphical interface. The interface offers two detection modes, Single Image and Bulk Image. Users can upload either one image at a time or several images for batch processing. In Single Image mode, users can upload an image, view the classification and detection results, and download the processed image with annotated bounding boxes. In Bulk Image mode, multiple images can be uploaded at the same time. After processing, users can download all detected images as a ZIP file. A "Thank You" button takes the user back to the welcome screen and improves usability. This interactive setup bridges the gap between model performance and real-world application by allowing non-technical users to utilize the system effectively in real time. It shows how practical the system is for deployment and supports its use in intelligent transportation monitoring scenarios.

4.7 Hardware and Software Setup

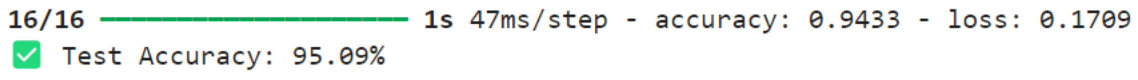
The entire project is developed and executed in the Google Colab environment and Kaggle environment. This setup provided the necessary computational resources for training and testing deep learning models. The hardware included an NVIDIA Tesla T4 GPU with 12GB RAM, which was enough to run both CNN and YOLOv8 models efficiently. The implementation use Python 3.10 and various libraries and frameworks for different parts of the pipeline. TensorFlow built and trained the ResNet50 CNN classifier.

PyTorch and the Ultralytics YOLOv8 framework is used for object detection tasks. Additional libraries such as OpenCV, NumPy, and Matplotlib were used for image processing, numerical computations, and result visualization respectively. To improve usability and allow real-time interaction, a Gradio-based graphical user interface is created. This interface lets users select between Single Image and Bulk Image detection modes, upload images, view detection results, and download the outputs. The use of Gradio shows that the system is ready for real-world use and supports its possible applications in intelligent transportation systems and road monitoring platforms.

5. Results

5.1 CNN Classification Accuracy:

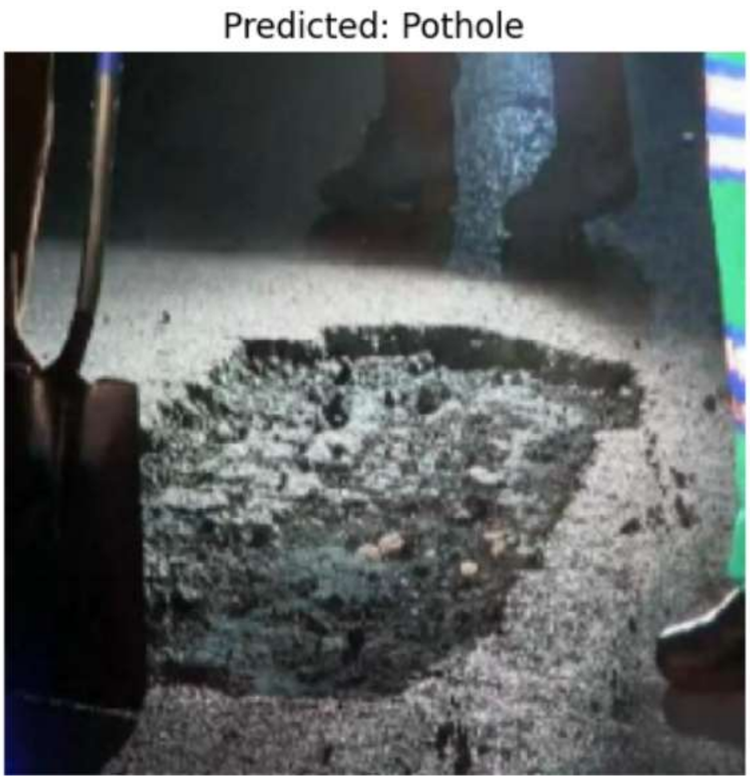
Figure 2:CNN Classification Accuracy



```
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 47ms/step - accuracy: 0.9433 - loss: 0.1709
✅ Test Accuracy: 95.09%
```

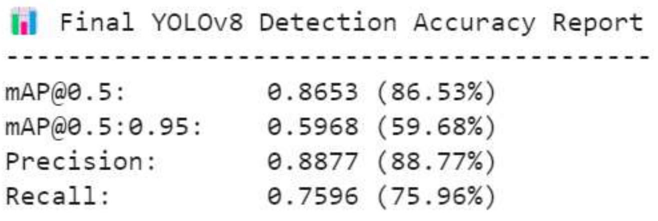
5.2 CNN Classification Prediction:

Figure 3:CNN Classification Prediction



5.3 YOLOV8 Detection Accuracy:

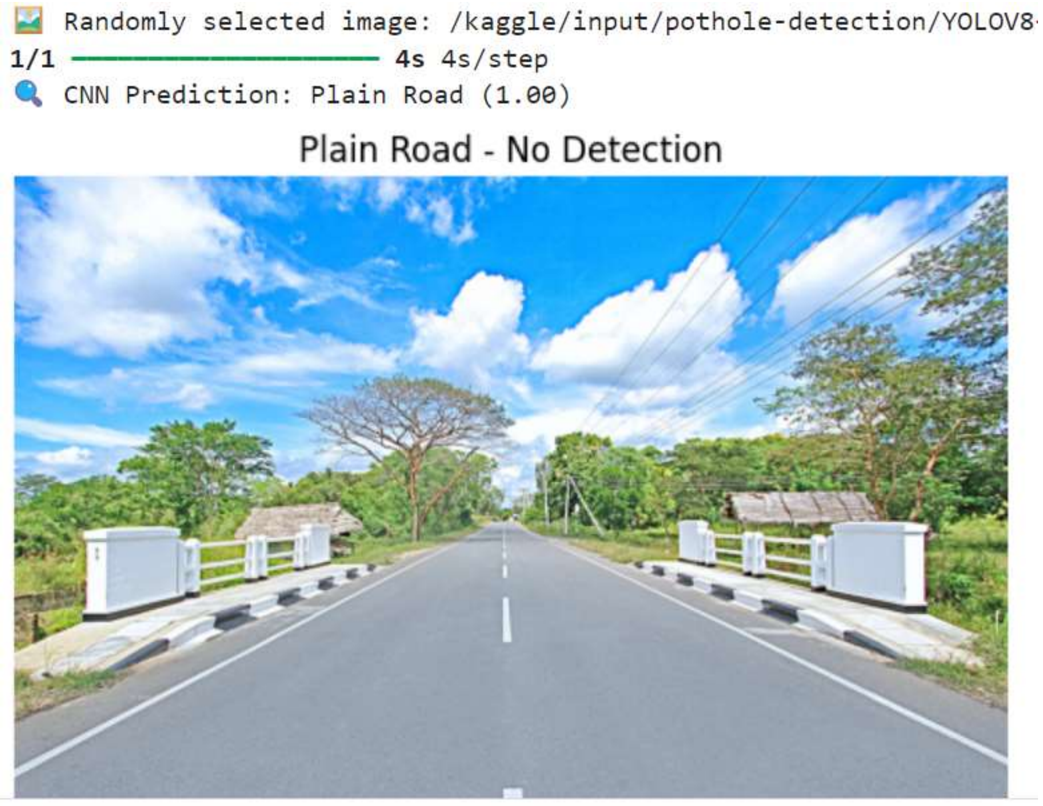
Figure 4:YOLOV8 Detection Accuracy



```
📊 Final YOLOv8 Detection Accuracy Report
------------------------------------------
mAP@0.5:          0.8653 (86.53%)
mAP@0.5:0.95:     0.5968 (59.68%)
Precision:        0.8877 (88.77%)
Recall:           0.7596 (75.96%)
```

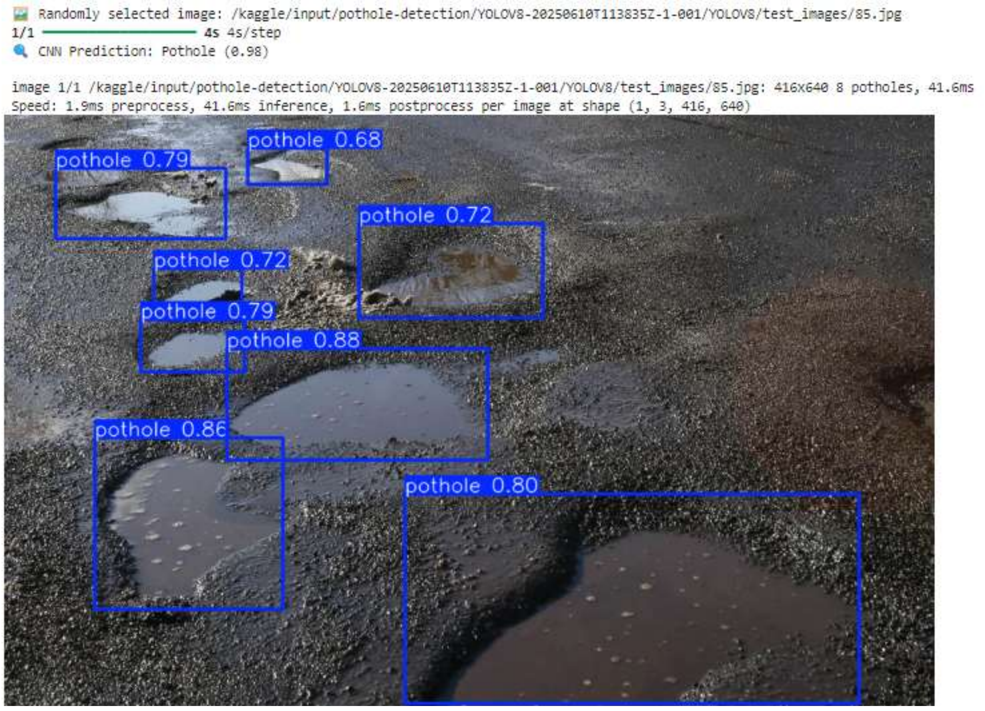
5.4 YOLOV8 Detection :

Figure 5:YOLOV8 Detection-Plain Road



```
🖼️ Randomly selected image: /kaggle/input/pothole-detection/YOLOV8
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
🔍 CNN Prediction: Plain Road (1.00)
```

Figure 6:YOLOV8 Detection-Pothole



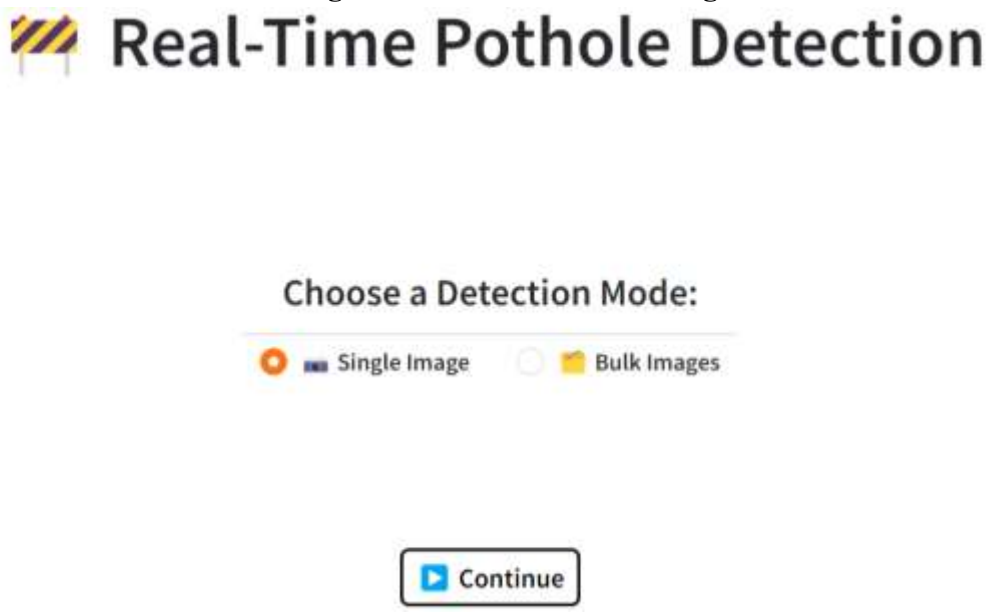
```
🖼️ Randomly selected image: /kaggle/input/pothole-detection/YOLOV8-20250610T113835Z-1-001/YOLOV8/test_images/85.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
🔍 CNN Prediction: Pothole (0.98)

image 1/1 /kaggle/input/pothole-detection/YOLOV8-20250610T113835Z-1-001/YOLOV8/test_images/85.jpg: 416x640 8 potholes, 41.6ms
Speed: 1.9ms preprocess, 41.6ms inference, 1.6ms postprocess per image at shape (1, 3, 416, 640)
```

5.5 Gradio Interface:

5.5.1 Welcome Page:

Figure 7:Gradio-Welcome Page



Real-Time Pothole Detection

Choose a Detection Mode:

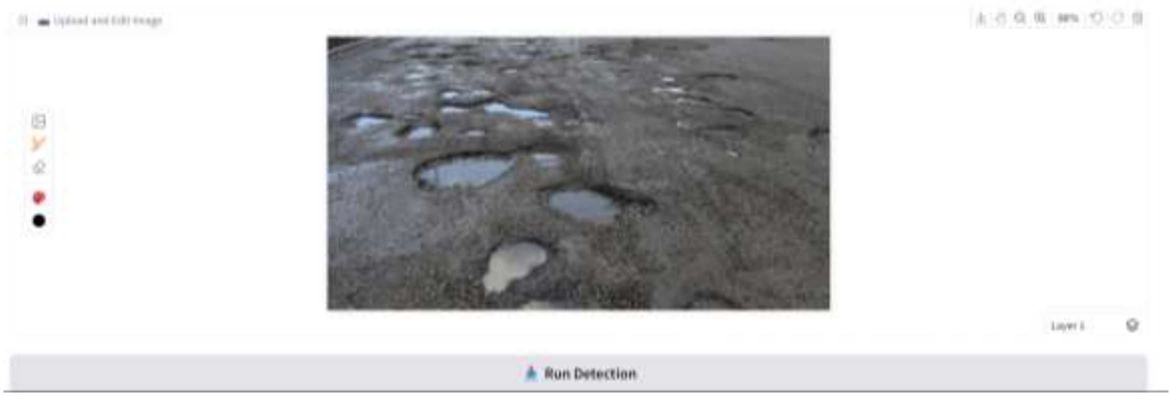
[✓ Single Image] Bulk Images

Continue

5.5.2 Upload Page:

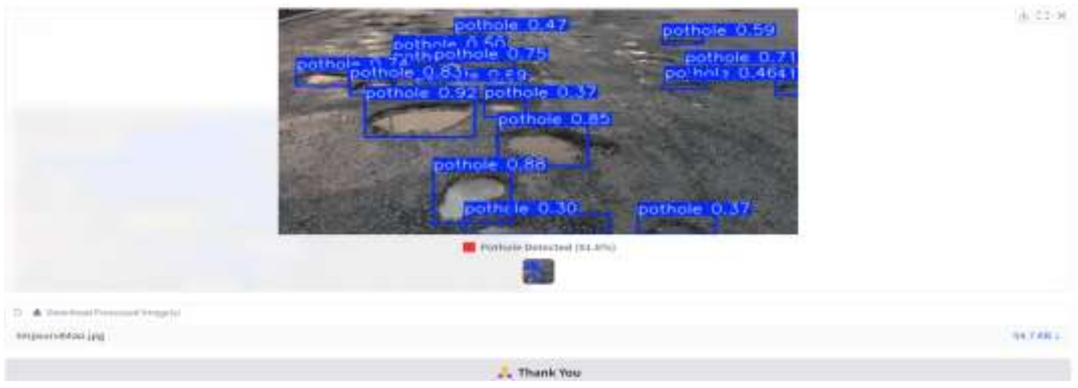
Single Image:

Figure 8:Gradio-Upload-Single Image



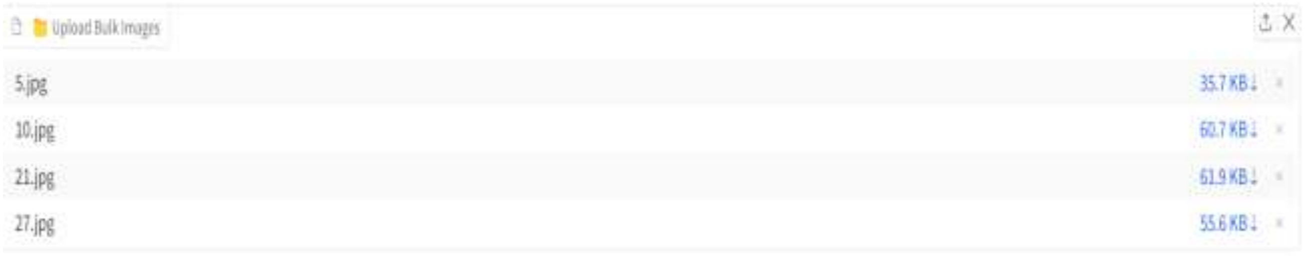
Single Image Detection:

Figure 9:Gradio-Single Image Detection



Bulk Images:

Figure 10:Gradio-Upload-Bulk Images

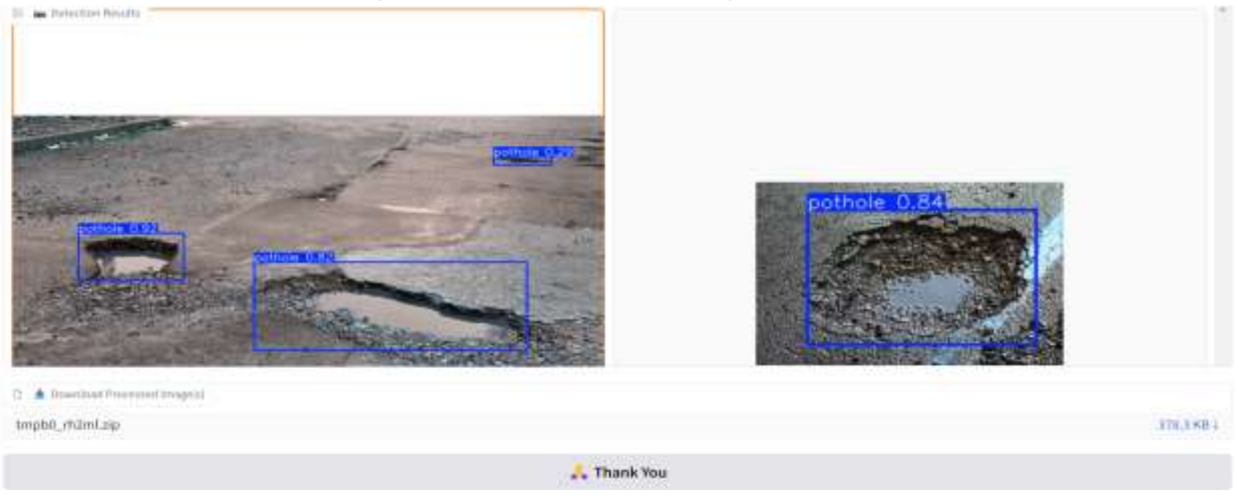


Upload Bulk Images

5.jpg	35.7 KB
10.jpg	60.7 KB
21.jpg	61.9 KB
27.jpg	55.6 KB

Bulk Images Detection:

Figure 11:Gradio-Bulk Images Detection



Detection Results

Download Processed Image(s)

tmpb0_rh2ml.zip 378.3 KB

Thank You

6. Conclusion

The project introduces a pothole detection system that combines CNN-based classification with YOLOv8 object detection for real-time analysis of road surfaces. The system uses ResNet50 to classify road conditions and YOLOv8 to accurately locate potholes. The ResNet50 classifier achieved a high test accuracy of 95.09%. This shows it performs well in differentiating between pothole images and plain road images. This initial classification step reduces computational demands by filtering out non-relevant inputs before detection. The YOLOv8 detection model, trained for 100 epochs, showed across multiple evaluation metrics. It achieved a mean Average Precision (mAP) of 86.53% at an IoU threshold of 0.5, indicating effective detection accuracy for identifying potholes with reasonable bounding box overlap. The more stringent metric, mAP@0.5:0.95, was 59.68%, reflecting the model's robustness across a range of IoU thresholds. In terms of detection quality, the model achieved a precision of 88.77%, meaning most predicted potholes were accurate, and a recall of 75.96%, showing a good rate of identifying real potholes in the dataset. These metrics together suggest a balanced model that can accurately and efficiently detect potholes. Overall, the system has strong potential for use in real-time smart road monitoring and can greatly contribute to safer and more effective transportation infrastructure.

7. References

1. Muhammad H.A., Saran K., Muhammad H.Y., et al., "Pothole Detection Using Deep Learning..."
2. Kaushik G., Soumyadip C., Arka K., "A Deep Learning Based Approach..."

3. Ashur R.A., Jianlin L., "Pothole Detection with YOLOv8"
4. Anurag M., "Detecting Potholes on Roads Using YOLOv8..."
5. Ken G., Elmo R., Lawrence R., Rue Nicole S., "Road Pothole Detection Using YOLOv8..."
6. Mohan A., et al., "Pothole Detection and Localization Using Deep CNN"
7. Adnan S., Haider A., Fareeha J., "Real-Time Pothole Detection Using YOLOv5"
8. Shubham G., Vishal R., Vinayak H., "Deep Learning-Based Pothole Detection System..."
9. John T.S., Ramesh K., "Automated Pothole Detection and Measurement..."
10. Pankaj V., Shubham S., "YOLO-Based Pothole Detection and Localization..."