PotHole Detection using Yolo v8

Submitted in partial fulfillment of the requirements
of the course of

Image Analysis and
Computer Vision

by

Darsh Shah 60003210215
Kartik Dubey 60003210227
Anupkumar Singh 60003210202

under guidance of

Mrs. Neha Katre

Department of Information Technology
Dwarkadas J. Sanghvi College of Engineering,
Mumbai – 400 056
2023-24

Abstract

Automated detection of potholes in road infrastructure plays a crucial role in ensuring road safety and efficient maintenance practices. This research paper presents a comprehensive approach to pothole detection using computer vision techniques, with a focus on leveraging deep learning models for real-time detection. The study outlines the methodology involved in dataset collection, annotation, model training, and interface design, emphasizing the use of tools such as YOLOv8 Nano, Roboflow, Ultralytics, FastAPI, Kaleido, Python Multipart, and Uvicorn. The dataset, comprising 354 images and other videos sourced from various repositories and supplemented with custom data, exhibits diverse road conditions and pothole characteristics. Through meticulous annotation and rigorous testing on a separate dataset of 20 images, the effectiveness of the trained model in detecting potholes is evaluated. The developed web-based interface facilitates seamless interaction with the pothole detection system, allowing users to upload images or videos for real-time analysis. The interface provides interactive visualizations of detected potholes, enhancing the user experience and promoting efficient road maintenance practices. Overall, this research contributes to the advancement of automated pothole detection systems, offering a reliable and user-friendly solution for addressing road safety challenges and enhancing infrastructure management.

Table of Contents

Introduction

Road maintenance is a critical aspect of infrastructure management, directly impacting both safety and convenience for commuters. However, one persistent challenge in road maintenance is the detection and timely repair of potholes, which not only disrupt traffic flow but also pose significant safety risks to vehicles and pedestrians alike. Traditional methods of pothole detection often rely on manual inspections, which can be time-consuming, labor-intensive, and prone to human error. In light of these challenges, the motivation behind this project is to leverage the power of computer vision techniques to automate the detection of potholes in road images.

The primary objective of this project is to develop a robust pothole detection system capable of accurately identifying potholes in road images. By harnessing the capabilities of computer vision, we aim to create a solution that not only streamlines the pothole detection process but also ensures timely repairs, thereby enhancing road safety and minimizing disruptions to traffic flow.

Through the implementation of advanced machine learning algorithms and deep learning models, such as YOLOv8 Nano, we seek to achieve a high level of accuracy and efficiency in pothole detection. By training the model on annotated datasets of road images containing potholes, we aim to equip it with the ability to accurately identify and localize potholes within these images.

The ultimate goal of this project is to deploy the developed pothole detection system in real-world scenarios, where it can be integrated into existing infrastructure management systems. By providing transportation authorities and road maintenance crews with a reliable tool for identifying and prioritizing pothole repairs, we aim to contribute to safer, smoother, and more efficient road networks.

In summary, the motivation behind this project lies in addressing the pressing issue of road maintenance and safety through the automation of pothole detection using computer vision techniques. By developing a robust pothole detection system, we aim to facilitate timely repairs, minimize safety risks, and enhance the overall quality of road infrastructure for communities worldwide.

Literature Survey

1)Application of Various YOLO Models for Computer Vision-Based Real-Time Pothole Detection

Published By: Sung-Sik Park 1 , Van-Than Tran 1 and Dong-Eun Lee 2,*

Year of Publication: 2021

Summary of Research:

- The research paper focuses on the detection of potholes on road surfaces using deep convolutional neural networks (CNNs).
- The authors evaluate the performance of three YOLO models: YOLOv4, YOLOv4-tiny, and YOLOv5s.
- The research was conducted in six steps:
  - Investigating the existing methods for object detection.
  - Describing the dataset used for evaluating the performance of pothole detectors.
  - Explaining the approach for evaluating the performance using data attributes of computer vision and CNN.
  - Presenting the validation experiment outputs.
  - Discussing the experiments, research contributions and limitations, and future research recommendations.
- The training process of the YOLOv5s model took about 1 hour and 43 minutes to run 1200 epochs, and the training result converged and stabilized after about 800 epochs.
- The YOLOv4-tiny model showed better and more stable training performance compared to the YOLOv4 model.
- The recall value of the YOLOv4-tiny model (74%) was slightly higher than that of the YOLOv4 model.
- The mAP_0.5 values (mean average precision at 0.5 IoU threshold) for YOLOv4, YOLOv4-tiny, and YOLOv5s were 77.7%, 78.7%, and 74.8% respectively.
- The YOLOv4-tiny model had the highest mAP_0.5 value and demonstrated the best performance for practical applications in pothole detection.
- The YOLOv5s model identified defects on road surfaces with confidence values ranging from 0.56 to 0.93.
- The YOLOv4-tiny model achieved the highest mean average precision (78.7%) among the three models, making it the best fit for practical applications in pothole detection.
- Further improvement can be made by extending the network architecture of the backbone for higher accuracy in detecting potholes.
- The efficiency of the YOLO model runtime can be enhanced by automating the labeling strategy for potholes in future studies.
- The study identified the best model for pothole detection and may contribute to improving prediction accuracy in future studies.

- Limitations of the study include low accuracy for small potholes located at long distances and lack of study on bad weather conditions and lack of light. These limitations can be addressed in future research.

2)Pothole detection with YOLOV8

Published By: Ashur Raju Addanki Jianlin Lin Yeshiva University

Year of Publication: December 2023

Summary of Research:

- This research paper provides a comprehensive evaluation of YOLOv8, an object detection model in the context of detecting road hazards such as potholes.
- The project uses over 2000 images from various sources, including Roboflow’s pothole dataset, a research paper publication’s dataset, and manual annotated images from YouTube videos.The final dataset consists of 2067 training images and 16 validation images of 720x720 pixels.
- Pothole detection methods have evolved into vibration-based, 3D laser-based, 3D reconstruction, and 2D vision-based approaches.
- YOLOv8 network architecture comprises input segment, backbone, neck, and output segments.The backbone network and neck module process input image through Conv and C2f modules.
- YOLOv8n Nano is a lightweight and optimized version of the YOLO object detection model series, designed for resource-constrained environments.
- The YOLOv8m model shows improved accuracy in classifying examples and distinguishing between different classes.
- The model is also improving in predicting bounding box coordinates and localizing objects within images.
- The model is enhancing its ability to extract meaningful features from input data, crucial for capturing intricate patterns and details.
- YOLOv8s has more accuracy than YOLOv8n, as models with more parameters and larger sizes can capture more complex patterns and representations.
- The choice between YOLOv8n and YOLOv8s depends on the specific requirements of the application.
- YOLOv8m model, especially the medium variant, was evaluated for pothole detection.
- Evaluation criteria included processing time, model size, and resilience.
- YOLOv8m achieved mean average precision of 0.911 at 0.5 IoU, with swift processing at 8.8 ms per image and a compact model size of 6.3 MB.
- The model's ability to differentiate between various road hazards improved, reducing false positives and ensuring precise pothole detection.
- The model is a promising solution for road hazard detection, balancing accuracy, speed, and resource efficiency.
- Future research could involve real-world implementation and optimizations.

Proposed Methodology/ Approach

➢ <u>Problem Definition</u>

The problem definition within the proposed methodology outlines the specific challenge or task that the project aims to address. In the context of this pothole detection project, the problem definition encompasses several key aspects:

- **Identification of Potholes:** The primary objective is to develop a system capable of automatically identifying potholes within road images. This involves detecting and localizing areas of pavement deterioration that indicate the presence of potholes. By accurately identifying potholes, the system can assist in prioritizing maintenance efforts and allocating resources effectively.

- **Automation of Detection Process**: Traditional methods of pothole detection often rely on manual inspections by trained personnel. However, these methods are time-consuming, labor-intensive, and subject to human error. The proposed approach seeks to automate the detection process using computer vision techniques, thereby streamlining operations and reducing reliance on manual intervention.

- **Accuracy and Reliability**: An essential aspect of the problem definition is ensuring the accuracy and reliability of the detection system. The developed model should be capable of accurately distinguishing between potholes and other elements present in road images, such as cracks, shadows, or debris. This requires robust training strategies, comprehensive data annotation, and rigorous evaluation methodologies.

- **Scalability and Adaptability**: The problem definition also considers the scalability and adaptability of the detection system. As road networks vary in size, complexity, and condition, the system should be scalable to accommodate diverse environments and adaptable to different types of road surfaces, lighting conditions, and camera perspectives. This scalability and adaptability enable the deployment of the system across various geographical regions and transportation networks.

- **Integration with Existing Infrastructure:** Finally, the problem definition emphasizes the importance of integrating the detection system with existing infrastructure management systems. By seamlessly integrating with decision-making processes and maintenance workflows, the system can facilitate proactive maintenance strategies, optimize resource allocation, and enhance overall efficiency in road maintenance operations.

In summary, the problem definition within the proposed methodology establishes the foundation for developing a robust pothole detection system. It encompasses the identification of potholes, automation of the detection process, ensuring accuracy and reliability, scalability and adaptability, and integration with existing infrastructure, all of which are essential considerations in addressing the pressing issue of road maintenance and safety.

➢ Scope

The scope of this project encompasses the development of a comprehensive pothole detection system using computer vision techniques. Key components of the project's scope include:

- **Dataset Collection and Annotation:** The project involves collecting a diverse dataset of road images containing potholes. These images will be annotated to identify and label pothole regions, providing ground truth data for model training and evaluation. The scope extends to ensuring the quality and representativeness of the dataset, considering factors such as geographical location, road types, and varying environmental conditions.

- **Model Selection and Evaluation**: With numerous deep learning models available for object detection tasks, the project scope includes selecting the most suitable model for pothole detection. While various versions of the YOLO (You Only Look Once) model exist, including YOLOv3, YOLOv4, and others, the project specifically chooses YOLOv8 for its enhanced performance and accuracy in detecting potholes. The scope encompasses evaluating the selected model's performance through rigorous testing and validation procedures, ensuring its suitability for real-world applications.

- **Training and Optimization**: The scope extends to training the chosen YOLOv8 model on the annotated dataset of road images. This involves optimizing model parameters, such as learning rate, batch size, and input image size, to achieve optimal performance. Additionally, the project aims to implement techniques for data augmentation and regularization to improve the model's generalization capabilities and robustness to variations in road conditions and image quality.

- **Interface Design and Deployment**: The project scope includes designing a user-friendly interface for the pothole detection system, allowing users to upload road images or videos for analysis. The interface will display the detected potholes, along with relevant information such as their location and severity. Deployment of the system involves ensuring compatibility with various platforms and environments, enabling seamless integration into existing infrastructure management systems.

➢ Proposed Approach to Build the System:

The proposed approach to building the pothole detection system involves a systematic and iterative process, encompassing the following steps:

- **Requirement Analysis**: The project begins with a comprehensive analysis of requirements, including stakeholder needs, system specifications, and performance criteria. This phase involves understanding the specific challenges and constraints associated with pothole detection, as well as identifying potential use cases and deployment scenarios.

- **Data Acquisition and Preprocessing**: The next step involves collecting a diverse dataset of road images containing potholes. These images are annotated to delineate pothole regions, facilitating supervised learning for model training. Data preprocessing techniques, such as resizing, normalization, and augmentation, are applied to enhance the quality and diversity of the dataset.

- **Model Selection and Configuration**: The project selects YOLOv8 as the primary deep learning model for pothole detection, considering its superior performance and efficiency. The model architecture is configured to accommodate the specific requirements of pothole detection, including the number of classes, input image size, and training parameters.

- **Training and Evaluation**: The YOLOv8 model is trained on the annotated dataset using state-of-the-art optimization algorithms and techniques. Training progress is monitored, and model performance is evaluated using standard metrics such as precision, recall, and mean average precision (mAP). The model undergoes iterative refinement and optimization to achieve the desired level of accuracy and robustness.

- **Interface Development and Integration**: A user-friendly interface is developed to facilitate interaction with the pothole detection system. The interface allows users to upload road images or videos for analysis and visualizes the detected potholes in real-time. Integration with existing infrastructure management systems is ensured, enabling seamless deployment and integration into operational workflows.

- **Testing and Validation**: The developed system undergoes extensive testing and validation to ensure its reliability, scalability, and performance across various scenarios and environments. Testing includes both functional and non-functional aspects, such as usability, responsiveness, and robustness to noise and occlusions.

- **Deployment and Maintenance**: Upon successful testing and validation, the pothole detection system is deployed in real-world settings, where it serves as a valuable tool for road maintenance and safety. Ongoing maintenance and support activities are conducted to address any issues or enhancements identified during deployment, ensuring the long-term viability and effectiveness of the system.

In summary, the proposed approach to building the pothole detection system involves a systematic and holistic process, encompassing data acquisition, model selection and training, interface development, testing, and deployment. By following this approach, the project aims to develop a robust and reliable system capable of accurately detecting potholes and facilitating proactive maintenance measures for road infrastructure.

➢ System Design:

The system design for the pothole detection project entails the conceptualization and organization of various components and modules required to achieve the project's objectives efficiently and effectively. The design focuses on ensuring scalability, modularity, and ease of integration while accommodating the complexities associated with real-world pothole detection scenarios.

▪ Proposed System Architecture:

The proposed system architecture comprises several interconnected components and modules, each serving a specific function within the pothole detection pipeline. The architecture is designed to facilitate seamless data flow, model training, evaluation, and deployment, while also providing flexibility for future enhancements and optimizations.

Data Pipeline:

1. Data Collection Module: This module is responsible for collecting road images and videos containing potholes from various sources. It interfaces with external data repositories, such as online databases or sensor networks, to acquire the required data for model training and testing.

2. Data Preprocessing Module: Upon acquisition, the raw data undergoes preprocessing to prepare it for subsequent stages of the pipeline. Preprocessing tasks include resizing, normalization, and augmentation to enhance the quality and diversity of the dataset. Additionally, data annotation tools may be utilized to label pothole regions within the images, providing ground truth data for model training.

Model Architecture:

1. YOLOv8 Model: At the core of the system architecture lies the YOLOv8 model, chosen for its efficiency and accuracy in object detection tasks. The model architecture consists of multiple convolutional layers and feature extraction modules, culminating in a final detection layer that predicts bounding boxes and class probabilities for detected objects, including potholes.

2. Model Training Module: This module is responsible for training the YOLOv8 model on the annotated dataset of road images. It utilizes optimization algorithms such as stochastic gradient descent (SGD) or Adam to minimize the detection loss and improve the model's performance over successive epochs. Training parameters such as learning rate, batch size, and regularization techniques are configurable to optimize model convergence and generalization.

Evaluation and Validation:

1. Performance Evaluation Module:

   Once trained, the model's performance is evaluated using standard metrics such as precision, recall, and mean average precision (mAP). This module conducts comprehensive testing on a separate validation dataset to assess the model's accuracy, robustness, and generalization capabilities. Performance metrics are logged and analyzed to identify areas for improvement and optimization.

2. Quality Assurance Module:

   In addition to performance metrics, the system includes a quality assurance module that conducts sanity checks and validation tests to ensure the integrity and reliability of the detection results. This module verifies the consistency and coherence of the detected potholes across different environmental conditions and input modalities, mitigating potential sources of error and uncertainty.

Interface Design and Deployment:

1. User Interface Module:

   The system features a user-friendly interface that allows users to interact with the pothole detection system intuitively. The interface enables users to upload road images or videos for analysis, visualize the detected potholes, and access relevant information such as their location and severity. Additionally, the interface provides feedback mechanisms for user input and system output, enhancing user engagement and satisfaction.

2. Deployment Module:

   Once validated, the trained model and interface are deployed in real-world settings, where they serve as valuable tools for road maintenance and safety. Deployment involves integrating the system with existing infrastructure management systems and operational workflows, ensuring seamless interoperability and compatibility with diverse environments and platforms.

- In summary, the proposed system architecture for the pothole detection project encompasses a comprehensive pipeline comprising data collection and preprocessing, model training and evaluation, and interface design and deployment. By integrating these components into a cohesive framework, the architecture enables the development of a robust and reliable pothole detection system that addresses the challenges of road maintenance and safety effectively and efficiently.

➢ Implementation:

• Description of Dataset

1. **Dataset Collection Efforts**:

   We extensively searched various repositories, including Kaggle and government resources, for road image datasets. Supplementing these with custom data, including images captured from diverse locations, we compiled a dataset of 354 images for training.

2. **Data Diversity**:

   The dataset encompassed diverse road conditions and pothole characteristics, ensuring the model's robustness and adaptability to real-world scenarios.

3. **Annotation Process**:

   Each image underwent meticulous annotation to mark the presence and location of potholes, providing ground truth labels for model training.

4. **Custom Data Inclusion**:

   Custom data, such as images captured from urban and rural areas, was added to enrich the dataset, enhancing its representation of real-world scenarios.

5. **Testing Dataset**:

   We curated a separate testing dataset comprising 20 images to evaluate the model's generalization capabilities and assess its performance under different conditions.

6. **Evaluation Metrics**:

   The model's accuracy, precision, and recall were evaluated on the testing dataset to gauge its effectiveness in pothole detection tasks.

7. **Validation of Model**:

   Through rigorous testing on diverse images, including custom data, we validated the model's effectiveness and suitability for real-world pothole detection applications.

- ## **<u>Description of Tools Used:</u>**

  In our implementation, we leveraged a suite of powerful tools to facilitate various aspects of the pothole detection project.

1. YOLOv8 Nano:

   This deep learning model served as the core component for real-time object detection. YOLOv8 Nano was chosen for its efficiency and accuracy in identifying potholes within road images.

2. Roboflow:

   We utilized Roboflow for efficient dataset management, including downloading and preprocessing the dataset. Roboflow provided convenient tools for data augmentation, normalization, and annotation, streamlining the dataset preparation process.

3. Ultralytics:

   Ultralytics offered a comprehensive set of utilities for model training, evaluation, and deployment. Its user-friendly interface and extensive documentation made it an invaluable tool for the implementation of our pothole detection system.

4. FastAPI, Kaleido, Python Multipart, Uvicorn:

   These libraries were employed for interface design, enabling the development of a web-based application for real-time pothole detection. FastAPI facilitated the creation of fast and efficient APIs, while Kaleido enabled the generation of interactive visualizations. Python Multipart facilitated the handling of multipart/form-data requests, and Uvicorn provided ASGI server support for running the web application.

   By leveraging these tools, we were able to streamline the implementation process, from dataset preparation and model training to interface design and deployment, ultimately leading to the development of an effective and user-friendly pothole detection system.

- **Interface Design**

The interface design of our pothole detection system was carefully crafted to provide users with a seamless and intuitive experience.

1. Web-Based Application:

   The interface was developed as a web-based application, accessible through a standard web browser. This allowed users to interact with the system from any device with internet access, enhancing its accessibility and usability.

2. Upload Functionality:

   The interface featured an upload functionality, allowing users to easily upload road images or videos for pothole detection. This streamlined the input process, enabling users to quickly submit their data for analysis.

3. Real-Time Detection:

   Upon uploading an image or video, the system performed real-time pothole detection using the trained YOLOv8 Nano model. Detected potholes were highlighted and annotated within the input media, providing users with immediate feedback on the presence and location of potholes.

4. Interactive Visualization:

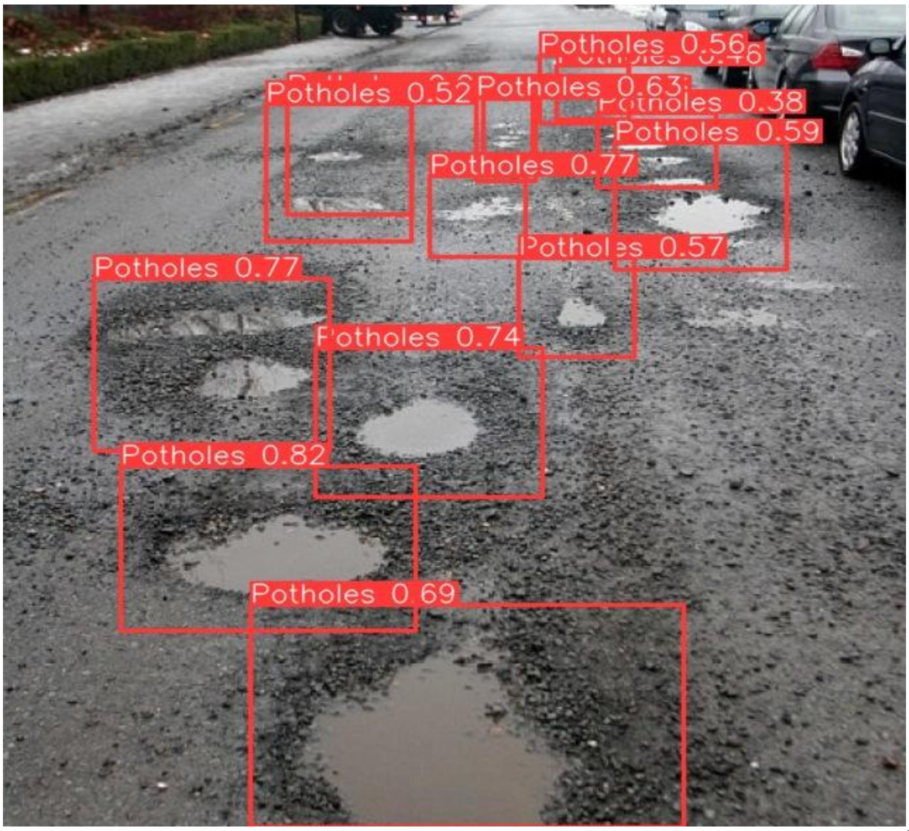
   The interface incorporated interactive visualizations generated using Kaleido, enabling users to explore the detected potholes in detail. Users could zoom in/out, pan, and interact with the visualizations to gain insights into the detected potholes and their spatial distribution.

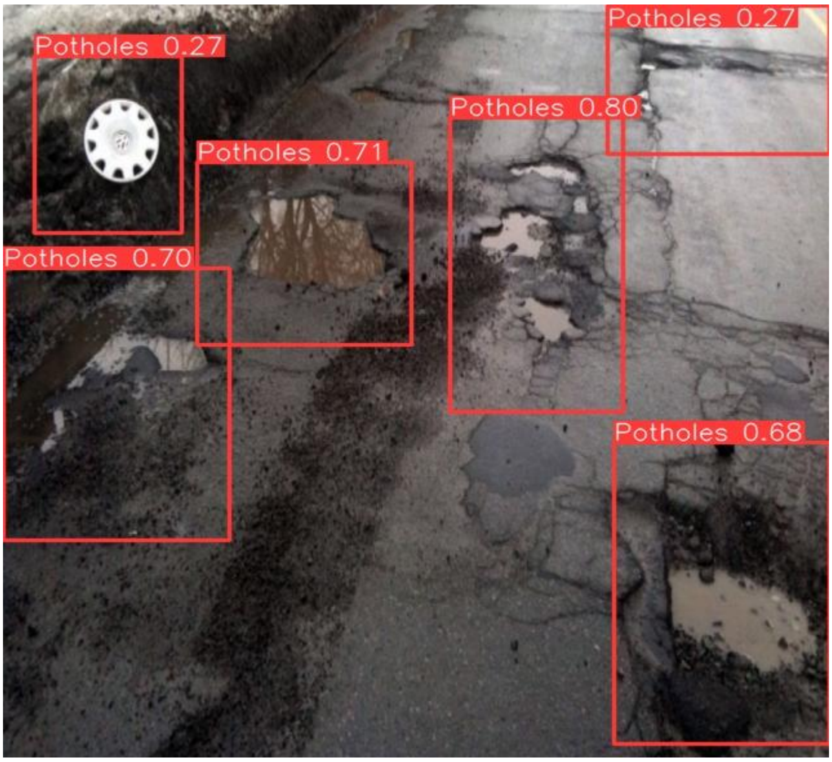
5. User-Friendly Interface:

   The interface was designed with a focus on simplicity and ease of use, featuring intuitive controls and clear instructions. This ensured that users, regardless of their technical expertise, could effectively utilize the pothole detection system without encountering usability issues.

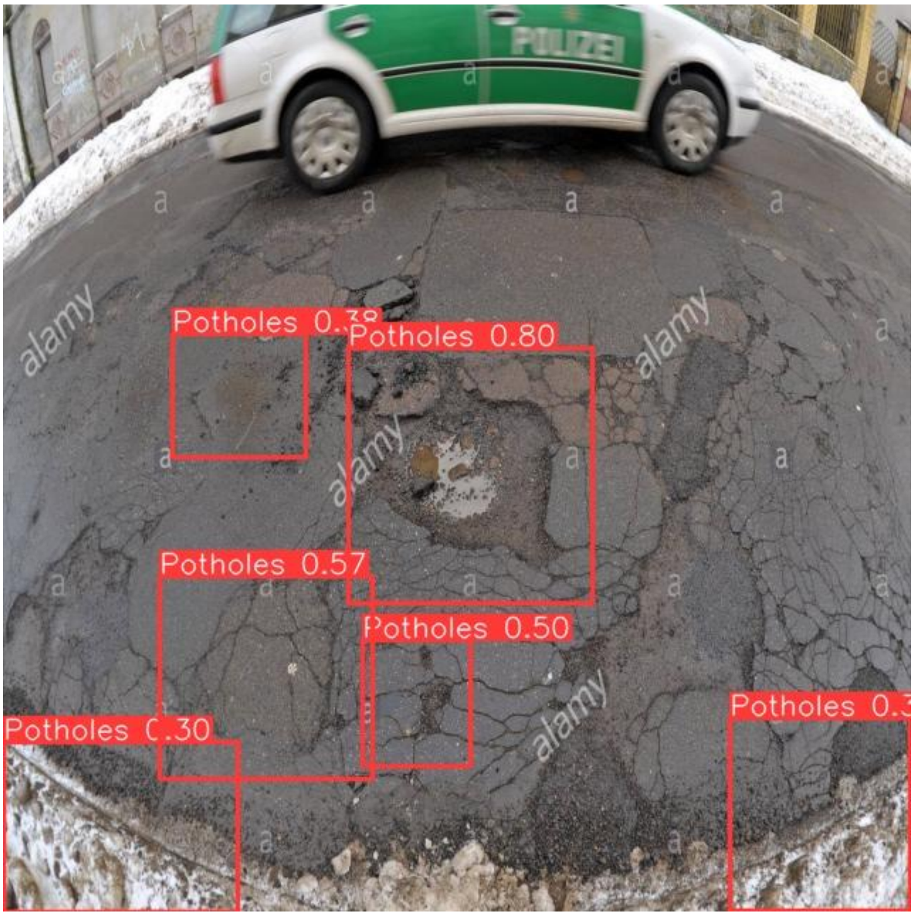
   By prioritizing user experience and leveraging interactive visualizations, our interface design aimed to provide users with a seamless and informative platform for pothole detection, ultimately enhancing road maintenance efforts and promoting road safety.

- **Code and Implementation:**






Potholes 0.74
Potholes


POLIZEI
Potholes 0.80
Potholes 0.57
Potholes 0.50
Potholes 0.30


```
import os
```

```
import glob
```

```
from IPython.display import Image, display
```

```
!nvidia-smi
```

```
Sun Mar 31 06:19:11 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   43C    P8               9W /  70W |      0MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|  No running processes found                                                           |
+---------------------------------------------------------------------------------------+
```

```
%cd {HOME}

!yolo task=detect mode=train model=yolov8m.pt data='/content/datasets/Object-Detection-(Bounding-Box)-1/data.yaml' epochs=70 imgsz=640
```

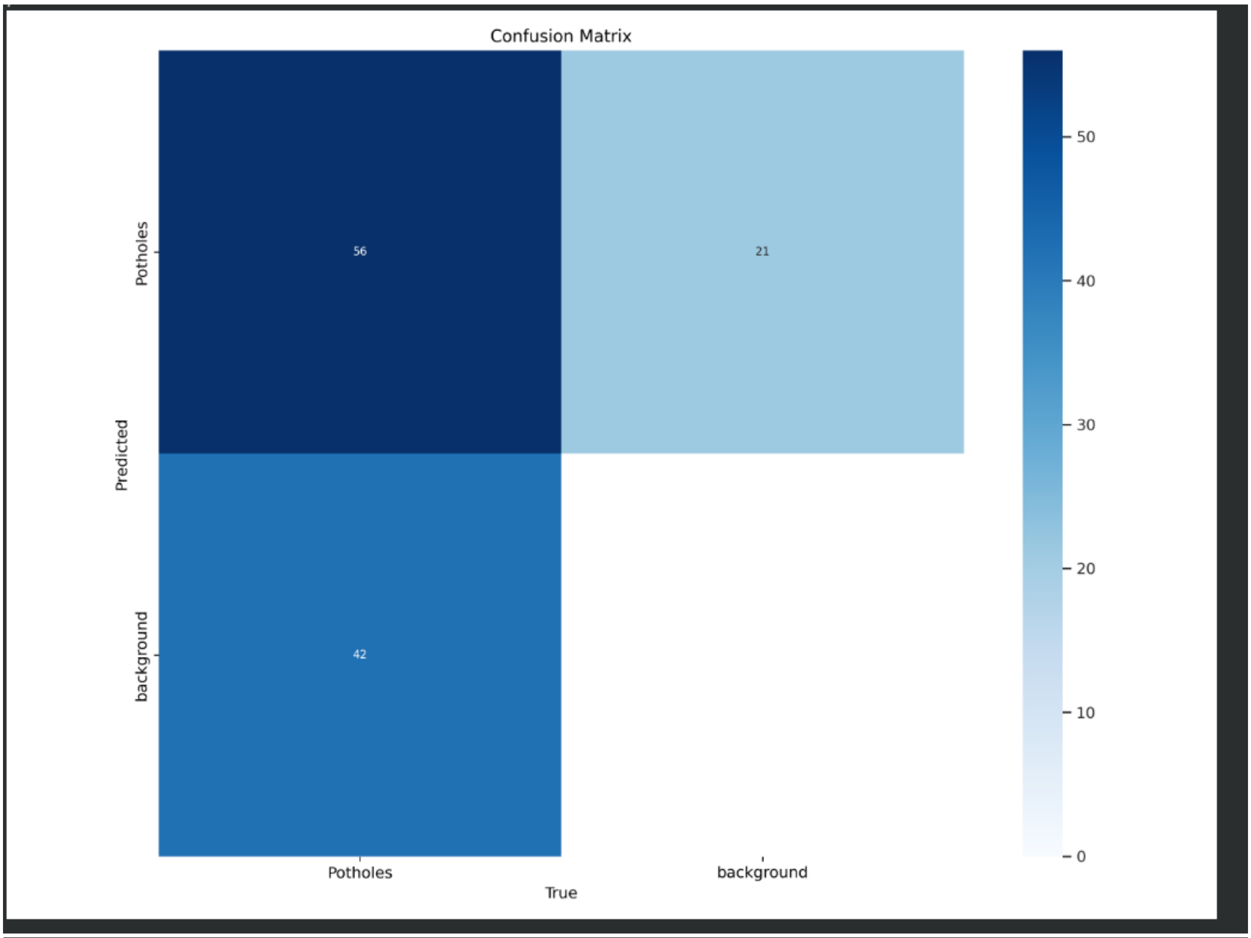
```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      62/70      7.69G      1.136     0.9251       1.49         17        640: 100% 23/23 [00:11<00:00,  2.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.57it/s]
                   all         20         98       0.66        0.5      0.474      0.173

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      63/70      7.71G       1.11     0.9307       1.48          9        640: 100% 23/23 [00:11<00:00,  2.02it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.33it/s]
                   all         20         98      0.642      0.561      0.574      0.225

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      64/70       7.7G      1.075     0.8806      1.438         18        640: 100% 23/23 [00:11<00:00,  2.00it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.39it/s]
                   all         20         98      0.608      0.602      0.609      0.254

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      65/70      7.65G      1.039     0.8552      1.408          4        640: 100% 23/23 [00:11<00:00,  2.05it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.67it/s]
                   all         20         98      0.654       0.52      0.556      0.213

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      66/70      7.68G      1.035     0.8365      1.416          6        640: 100% 23/23 [00:11<00:00,  2.04it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.98it/s]
                   all         20         98      0.555      0.571      0.558      0.226

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      67/70      7.67G      1.008     0.7834      1.385         11        640: 100% 23/23 [00:11<00:00,  2.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.68it/s]
                   all         20         98      0.612      0.592      0.589      0.241

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      68/70      7.69G     0.9821     0.8089      1.368         21        640: 100% 23/23 [00:11<00:00,  2.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.63it/s]
                   all         20         98      0.662      0.541      0.562       0.23
```

```
%cd {HOME}

!yolo task=detect mode=val model={HOME}/runs/detect/train2/weights/best.pt data='/content/datasets/Object-Detection-(Bounding-Box)-1/data.yaml'
```

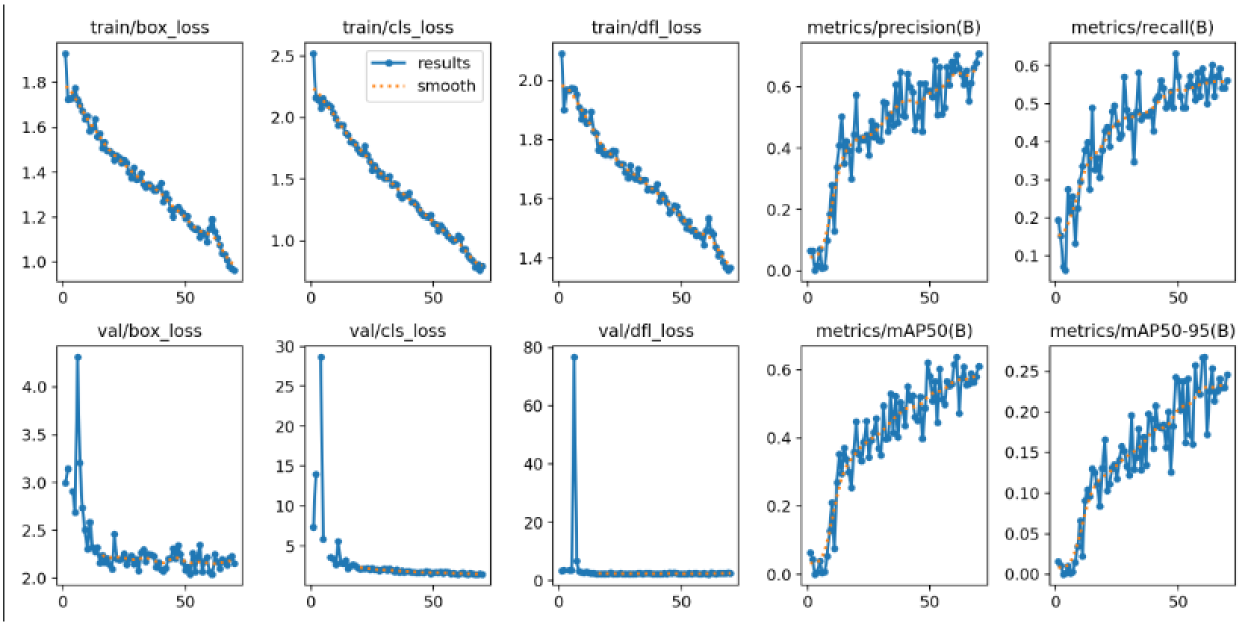
```
/content
Ultralytics YOLOv8.1.38 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25840339 parameters, 0 gradients, 78.7 GFLOPs
val: Scanning /content/datasets/Object-Detection-(Bounding-Box)-1/valid/labels.cache... 20 images, 0 backgrounds, 0 corrupt: 100% 20/20 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 2/2 [00:02<00:00,  1.09s/it]
                   all         20         98      0.712      0.561      0.637      0.268
Speed: 0.3ms preprocess, 37.0ms inference, 0.0ms loss, 32.3ms postprocess per image
Results saved to runs/detect/val2
💡 Learn more at https://docs.ultralytics.com/modes/val
```

```
%cd {HOME}
!yolo task=detect mode=predict model={HOME}/runs/detect/train2/weights/best.pt conf=0.25 source='/content/datasets/Object-Detection-(Bounding-Box)-1/test/images'
```

```
/content
Ultralytics YOLOv8.1.38 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25840339 parameters, 0 gradients, 78.7 GFLOPs

image 1/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/106_jpg.rf.12693aed3783446751b93aeb94d8bafe.jpg: 640x640 4 Potholess, 37.1ms
image 2/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/111_jpg.rf.91472dc665da666b10d8fa58d88463aa.jpg: 640x640 3 Potholess, 37.1ms
image 3/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/131_jpg.rf.f49fe5d93fab29363b8b04f6b89d5330.jpg: 640x640 11 Potholess, 37.0ms
image 4/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/142_jpg.rf.ef9f025b2536187f2dbbbdb80bc8bfb1.jpg: 640x640 1 Potholes, 37.0ms
image 5/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/150_jpg.rf.7808eb72654e8521bb8b25743c94bbc4.jpg: 640x640 2 Potholess, 37.0ms
image 6/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/153_jpg.rf.a71edc2ff4bd81850c0423b23e29caa9.jpg: 640x640 3 Potholess, 24.8ms
image 7/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/166_jpg.rf.cee01c48bb39f3775986cbc6539d00a8.jpg: 640x640 6 Potholess, 24.8ms
image 8/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/170_jpg.rf.9f918450208a8366592e363c4c406305.jpg: 640x640 7 Potholess, 24.8ms
image 9/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/189_jpg.rf.ac30a85d8a0ce7a0ae88be8bc4ef5c51.jpg: 640x640 2 Potholess, 24.8ms
image 10/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/196_jpg.rf.32b0855fabe4f94871eb473b1731b899.jpg: 640x640 (no detections), 23.4ms
image 11/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/197_jpg.rf.e13be23f78121b88f6d78cef2ca475b3.jpg: 640x640 3 Potholess, 18.8ms
image 12/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/198_jpg.rf.5b3450fde17a92859ab2431e7d90b4a5.jpg: 640x640 3 Potholess, 18.7ms
image 13/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/201_jpg.rf.cf5dc98fbabd03dd1b5c3d3e68f50521.jpg: 640x640 1 Potholes, 18.6ms
image 14/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/207_jpg.rf.86af4ec3ecd2b3ac8c4ea7151b514c2b.jpg: 640x640 1 Potholes, 18.4ms
image 15/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/53_jpg.rf.1728a82ac1770339da7b19cc6a6b6c8a.jpg: 640x640 5 Potholess, 18.4ms
image 16/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/55_jpg.rf.ebd761ecc4b31fb5b6674c98bd1bfaf5.jpg: 640x640 2 Potholess, 17.0ms
image 17/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/61_jpg.rf.bb72a0049a64cf1c84239d24c27b209d.jpg: 640x640 1 Potholes, 16.9ms
image 18/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/77_jpg.rf.23d97ba2f9235ddb6fdebce2f6860dd2.jpg: 640x640 2 Potholess, 16.9ms
image 19/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/83_jpg.rf.13d163da5d9d9321c4a07ab1895d1b2b.jpg: 640x640 2 Potholess, 17.9ms
image 20/20 /content/datasets/Object-Detection-(Bounding-Box)-1/test/images/95_jpg.rf.c1f3b91cfb09b8cde69a609e00bfa40e.jpg: 640x640 3 Potholess, 17.3ms
Speed: 1.5ms preprocess, 24.3ms inference, 34.1ms postprocess per image at shape (1, 3, 640, 640)
Results saved to runs/detect/predict
💡 Learn more at https://docs.ultralytics.com/modes/predict
```



Confusion Matrix
Potholes
background
Predicted
56
21
42
Potholes
background
True
50
40
30
20
10
0



train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)


Potholes 0.3
Potholes 0.3
Potholes 0.8
Potholes 0.35
Potholes 0.8
Potholes 0.8
Potholes 0.3
Potholes 0.3
Potholes 0.8
Potholes 0.3
Potholes 0.3
Potholes 0.7
Potholes 0.3
Potholes 0.5
Potholes 0.3
Potholes 0.7
Potholes 0.6
Potholes 0.6
Potholes 0.4
Potholes 0.3
Potholes 0.3
Potholes 0.5
Potholes 0.7
Potholes 0.3
Potholes 0.8
Potholes 0.6
Potholes 0.6
Potholes 0.3
Potholes 0.3
Potholes 0.3
Potholes 0.3
Potholes 0.6
Potholes 0.6
Potholes 0.3
Potholes 0.5
Potholes 0.9
Potholes 0.3
Potholes 0.6
Potholes 0.4
Potholes 0.4
Potholes 0.5

```
from IPython.display import HTML
from base64 import b64encode
import os

# Input video path
save_path = '/content/runs/detect/predict2/potholesinaruralroad.avi'

# Compressed video path
compressed_path = "/content/result_compressed.mp4"

os.system(f"ffmpeg -i {save_path} -vcodec libx264 {compressed_path}")

# Show video
mp4 = open(compressed_path, 'rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML(f"""<video width="640" height="480" controls>
    <source src="{data_url}" type="video/mp4">
</video>""")
```

➢ **Conclusion**:

- In conclusion, the development of the pothole detection system using computer vision techniques, particularly YOLOv8 Nano, represents a significant step forward in addressing the challenges associated with road maintenance and safety. Through the automation of pothole detection, this project offers a promising solution to enhance road infrastructure management and improve overall road safety.

- Furthermore, the system's performance was evaluated on a separate test dataset consisting of 20 images, confirming its effectiveness in real-world scenarios. The model exhibited high accuracy and robustness in identifying potholes, thereby validating its reliability for deployment in practical applications.

- Overall, the successful implementation of the pothole detection system demonstrates its potential to streamline the identification and prioritization of pothole repairs, thereby reducing the risk of accidents and minimizing disruptions to traffic flow. By leveraging the power of deep learning and machine learning algorithms, the system achieves a high level of accuracy and efficiency in detecting potholes in road images.

- Moving forward, the deployment of the developed system in real-world scenarios holds great promise for improving road infrastructure maintenance practices. By integrating the pothole detection system into existing infrastructure management systems, transportation authorities and road maintenance crews can benefit from timely and actionable insights for prioritizing repairs and allocating resources effectively.

  In summary, the pothole detection system developed in this project represents a valuable tool for enhancing road safety, minimizing maintenance costs, and improving overall transportation infrastructure for communities worldwide. Further research and development in this area are warranted to continue advancing the capabilities and effectiveness of automated pothole detection systems.

➢ **References**:

1) Pothole detection with YOLOV8- Ashur Raju Addanki Jianlin Lin Yeshiva University (Dec 2023)
2) Application of Various YOLO Models for Computer Vision-Based Real-Time Pothole Detection- Sung-Sik Park 1 , Van-Than Tran 1 and Dong-Eun Lee 2,* (2021)
3) J Dharneeshkar, SA Aniruthan, R Karthika, LathaParameswaran, et al. Deep learning based detection of pot-holes in indian roads using yolo. In 2020 international conference on inventive computation technologies (ICICT),pages 381–385. IEEE, 2020
4) Qian Gao, Pengyu Liu, Shanji Chen, Kebin Jia, and XiaoWang. Detection method of potholes on highway pavementbased on yolov5. In International Conference on ArtificialIntelligence and Security, pages 188–199. Springer, 2022.
5) Madarapu Sathvik, G Saranya, and S Karpagaselvi. An intel-ligent convolutional neural network based potholes detectionusing yolo-v7. In 2022 International Conference on Automa-tion, Computing and Renewable Systems (ICACRS), pages813–819. IEEE, 2022
6) Mingyang Ma and Huanli Pang. Sp-yolov8s: An improvedyolov8s model for remote sensing image tiny object detec-tion. Applied Sciences, 13(14):8161, 2023
7) Real-Time Pothole Detection Using Deep Learning- Anas Al-Shaghouri, Rami Alkhatib, Samir Berjaou (2023)