**Road Damage Surveyor Agent**

**Team Lead**: G.Bhuvaneswari [22B21A4310]

**Team Members**:

B.Bhagya Lakshmi [22B21A4322]

P. Santhosh [22B21A4340]

Ch. Ravi Kumar [22B21A4346]

M. Hemanth [22B21A43A3]

**Department:** Computer Science and Artificial Intelligence

**Institution:** Kakinada Institute of Engineering and Technology

**Academic Year**: 2025

**Abstract:**

This project proposes the development of a fully autonomous Road Damage Surveyor Agent that detects potholes in road images through deep learning techniques developed completely from scratch. Unlike approaches relying on pretrained models, the system begins with raw annotated image datasets, designing and training convolutional neural networks (CNN) from the ground up to identify and localize potholes with precision. The system also extracts GPS location information directly from image metadata or synchronous sensors to geotag damages accurately. Automatic generation of detailed maintenance reports—including pothole count, severity classifications, precise latitude and longitude, and timestamps—enables data-driven road repair prioritization. This end-to-end automation reduces dependence on slow, labor-intensive manual inspections, accelerates road maintenance, and enhances commuter safety by ensuring timely hazard identification and remediation.

**Introduction & Problem Statement:**

Road infrastructure is a fundamental component of modern society, playing a vital role in facilitating transportation, economic activities, and daily commuting. Over time, road surfaces naturally deteriorate due to various factors such as weather conditions, repeated heavy vehicle loads, and the aging of construction materials. This deterioration often results in defects like potholes, which are bowl-shaped depressions in the road surface. Potholes not only cause significant damage to vehicles but also lead to accidents, increased travel time, and pose serious safety risks, especially for two-wheelers and high-speed traffic.

Traditional inspection methods for detecting potholes primarily rely on manual surveys conducted by road authorities or complaints from the public. These approaches are labor-intensive, time-consuming, and subject to inconsistencies due to human error or inadequate coverage. With the advent of computer vision and artificial intelligence, new possibilities have emerged for automating pothole detection. Real-time detection using deep learning techniques on images captured from vehicle-mounted cameras or drones can significantly improve the efficiency and accuracy of road damage monitoring. When integrated with location data, such automated systems provide actionable insights for timely and targeted road maintenance.

**Problem Statement:**

Current road maintenance lacks automated, real-time pothole detection with precise geolocation, causing delays and safety risks. There is a need for a deep learning system trained from scratch that detects potholes, integrates GPS data, and automates detailed reporting to improve repair efficiency and road safety. The major problem is the absence of an integrated system that accurately detects potholes in real-time, links them to precise GPS locations, and automatically generates maintenance reports.

**Proposed Methodology:**

The system employs CNN-based deep learning models trained on annotated road damage datasets to detect and classify potholes. It preprocesses input images using resizing and noise reduction techniques for consistency. GPS data is integrated from image metadata or synchronized sensors to geotag detected damages accurately.

**Dataset**

Publicly available annotated datasets (e.g., from Japan and India) along with custom-collected images provide diverse pothole examples. Data augmentation strategies such as flips, rotations, and noise addition increase robustness.

Link: <https://www.kaggle.com/datasets/andrewmvd/pothole-detection>

**PreTrained By Using Models:**

* The Road Damage Surveyor Agent will be developed using a combination of computer vision models, supervised machine learning, and geospatial mapping tools.
* Techniques Used:
  + Image preprocessing: resizing, augmentation, noise reduction.
  + Detection model: CNN-based architecture (YOLOv5 / Faster R-CNN) for pothole detection and classification.
  + Location tagging: GPS integration with images using geotag metadata or external GPS modules.
  + Reporting system: Automated generation of reports (damage severity, count, GPS coordinates, timestamp).
* Tools & Frameworks:
* **Python** – Core programming language for development.
* **TensorFlow/PyTorch** – Model building, training, and evaluation.
* **OpenCV** – Image/video preprocessing and augmentation.
* **GPS Libraries** – Extract and integrate geographic coordinates.
* **Report Generation Tools** – PDF/Excel libraries (ReportLab, OpenPyXL) for automated report creation.

**Scratch Model:**

A CNN-based detection model (inspired by YOLOv5/Faster R-CNN) is trained from scratch in a supervised manner to identify pothole locations and classify their severity.

**1. Data Collection and Preparation**

Road images and video frames are captured using drones or vehicles to create a diverse dataset of pothole scenarios. Each image is annotated using tools like LabelImg or Roboflow, where bounding boxes are manually drawn around potholes and severity levels are labeled (Minor, Moderate, Severe). The dataset is split into:

* **Training Set**: 70–80% of images for model learning.
* **Validation/Testing Set**: 20–30% of images for performance evaluation.

**2. Data Preprocessing**

Using **OpenCV**, images undergo:

* **Resizing** to a consistent dimension (e.g., 640×640 or 512×512).
* **Noise Reduction** and brightness/contrast adjustments for uniform quality.
* **Data Augmentation** (flipping, rotation, cropping) to improve generalization.  
  Annotation files are converted to the required model format (YOLO text files or COCO JSON).

**3. Model Architecture**

A custom **CNN-based object detection model** is built from scratch using **PyTorch**, with possible approach:

* **Basic CNN Detector**: Convolution + BatchNorm + ReLU layers producing feature maps, followed by region proposal/anchor boxes, and separate heads for classification and bounding box regression.

**4. Training Configuration**

The training loop is designed with:

* **Loss Functions**: Cross-entropy for classification, Smooth L1/IoU loss for bounding boxes.
* **Optimizers**: Adam or SGD.
* **Hyperparameters**: Batch size 8–32, learning rate 0.001–0.0001, 50–150 epochs.  
  A **GPU-enabled environment** (Google Colab, Kaggle, or a local GPU machine) is used for efficient training.

**5. Model Training**

* Load training data in batches.
* Perform forward passes, compute losses, backpropagate, and update weights.
* Track training and validation losses to monitor convergence.
* Save checkpoints of model weights periodically to avoid loss of progress.

**6. Evaluation and Tuning**

Model performance is evaluated using the validation/test dataset with metrics such as:

* Precision, Recall, mAP (mean Average Precision).
* **IoU (Intersection over Union)** for bounding box accuracy.  
  Hyperparameters or model depth are adjusted to improve performance as needed.

**7. System Integration**

The trained model is integrated into the full pipeline to process live images or video frames. Detected potholes are linked with GPS coordinates using location libraries. Automated PDF/Excel reports are generated using **ReportLab** or **OpenPyXL**, containing pothole images, locations, and severity details.

**8. Deployment**

The complete system—including the model, preprocessing, and reporting pipeline—is packaged for deployment. It can run on:

* **Edge Devices** such as Raspberry Pi or NVIDIA Jetson for mobile/field use.
* **Cloud Servers** for large-scale processing of urban or regional road networks.

**System Flow Diagram:**

1. **Input Acquisition:** Road images or video feed captured by vehicle-mounted cameras or UAVs.
2. **Pre-processing:** Noise removal, resizing, and normalization of images.
3. **Pothole Detection:** Deep learning model outputs bounding boxes around detected damages.
4. **Location Tagging:** Extract GPS information from image metadata or external sensors.
5. **Report Generation:** Automated generation of repair reports detailing damage counts, severity, GPS coordinates, and timestamps.
6. **Output:** Structured maintenance report provided to municipal author

Start

↓

Input Acquisition

↓

Pre-processing

↓

Pothole Detection

↓

Location Tagging

↓

Report Generation

↓

Output

↓

End

**Expected Working:**

The system captures images in real-time as a vehicle or drone moves, covering large road networks efficiently. Captured images are preprocessed (noise reduction, resizing, normalization) to ensure consistent input quality. A deep learning model trained on diverse pothole images then detects and classifies potholes by severity. The system supports both batch and real-time data processing, and generates reports in PDF/Excel formats, which can be integrated with municipal dashboards for streamlined monitoring and resource allocation.

**Conclusion:**

The Road Damage Surveyor Agent aims to automate pothole detection and reporting effectively through deep learning and GPS integration. This project reduces reliance on slow manual inspections, expedites maintenance actions, and promotes safer roads by enabling data-driven infrastructure management. Future work may include real-time video processing, dashboard integration, and expanded coverage across urban and rural areas.

**References:**

* Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road Damage Detection Using Deep Neural Networks with Images Captured Through a Smartphone. *arXiv preprint arXiv:1801.09454*.
* Zhang, Z., & Chen, D. (2019). Automatic Pothole Detection System Using Deep Learning for Pavement Inspection. *IEEE Access*.
* Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*.
* OpenCV Library - Computer Vision and Image Processing Framework. Available at: <https://opencv.org/>
* TensorFlow & PyTorch Documentation for deep learning experimentation.