SPOTHOLE

(POTHOLE DETECTION USING CNN)

Software Requirement Specification

KMEC Group 270

INDEX

INTRODUCTION

Purpose

In order to ensure that the road still meets the needs of services generally, human visual inspection is the primary way of assessing the condition of the road. Manually examining and evaluating visual road data, however, is a time-consuming and expensive process. Also, the subjectivity and expertise of the human raters have a significant impact on the outcomes. Also, these techniques are exceedingly laborious and uncomfortable for inspectors. The development of science and technology, as well as the acceptance of the deep learning model in the engineering community, has made it possible to replace humans with sophisticated, intelligent systems that are inexpensive and low-cost. Our pothole detection system uses deep learning, computer vision, and convolutional neural networks to accurately detect potholes on the road.

Scope

Approaches for roadway pothole detection technology are being developed to provide offline data collection for road repair or real offline vehicle control (for vehicular applications or automated driving). The potential of pothole detecting technology to increase highway safety and lower the cost of road repair has made it a crucial research area. Researchers have been researching numerous methods for detecting potholes on roads, and one of the most promising is deep learning algorithms. Artificial neural networks are used by deep learning algorithms to extract patterns and characteristics from massive amounts of data. Several applications, including object identification and picture recognition, have successfully exploited this strategy. Pothole detection on road photos can be done using the same methods. Convolutional neural networks (CNN) are one such method. CNNs have been demonstrated to perform with excellent accuracy in a variety of image recognition applications because they are built to process images by learning relevant characteristics and patterns. Researchers have been investigating a number of methods for spotting potholes on roads all around the world due to these factors. Mass transit is supported by roads, which provide a significant economic contribution. In the transportation networks, potholes in the roads are a major source of worry. Many studies have recommended using deep learning algorithms, which deal with diverse image analysis and object detection methods, to automate pothole detection. The pothole detection method needs to be automated with the highest level of accuracy and dependability.

It has been demonstrated that in terms of pothole identification, CNN-based models outperform conventional

Abbreviations

CV – Computer Vision

CNN – Convolutional Neural Network

References

- CNN-based Real-time Pothole Detection for Avoidance Road Accident published by Arizona State university

System Overview

Brief Description

The pothole detection system follows a structured workflow, integrating user interaction, machine learning, and backend processing. The process begins with a user logging into the system to gain access. Once authenticated, the user uploads an image of a road, which is then sent to a FastAPI endpoint. This endpoint processes the image by forwarding it to a machine learning model designed to detect potholes. The image undergoes preprocessing before being analysed by the model, which generates a prediction result. The predicted result is then sent back to the FastAPI backend and subsequently to the front-end interface in the form of a json containing the prediction and the predictability percentage. At this stage, a decision is made based on whether a pothole is detected. If a pothole is found, the image, along with additional metadata such as location and description, is sent to a Node.js backend for further storage and processing. The detected pothole details are then recorded in a database, ensuring that the information can be retrieved later to send this to the required authorities for further action. Simultaneously, the data is also made available in the "History" section of the profile tab for user reference. Conversely, if no pothole is detected, the user is redirected back to the image upload section to try again with a different image. The system also allows users to review past submissions in the "History" section, providing a comprehensive record of all detected potholes. Finally, users have the option to log out of the system once they have completed their tasks. This workflow efficiently integrates machine learning, API-based communication, and database storage to provide a seamless pothole detection experience.

Key Features

The pothole detection system incorporates essential security and usability features to enhance the user experience and ensure data protection. One of the key features is authentication using JWT (JSON Web Tokens) along with encryption. This ensures that only authorized users can access the system by securely verifying their identity. When a user logs in, their credentials are authenticated, and a JWT token is generated. This token acts as a digital passport, allowing the user to make secure requests to the backend without needing to log in repeatedly. Additionally, sensitive user information and authentication tokens are encrypted to prevent unauthorized access or data breaches. This enhances the security of user sessions and protects personal data from cyber threats.

Another significant feature is the History section on the profile page, which allows users to keep track of past image uploads and pothole detection results. Every time a user uploads an image, the system processes it, and if a pothole is detected, the relevant details—including the image, its location, and additional metadata—are stored in a database. These records are then accessible in the user's profile under the History section, providing a structured and easily navigable log of all past detections. This feature enhances usability by allowing users to review previous results, monitor pothole detection trends, and access historical data for reference or reporting purposes.

By integrating secure authentication and encryption with a well-structured history management system, this project ensures both data security and an efficient user experience, making it a robust solution for pothole detection and tracking.

Operating Environment

Software Requirements

To develop and run the pothole detection system efficiently, the following software requirements must be met:

1. Operating System

- Windows 10/11 (64-bit)
- macOS (latest version)
- Linux

2. Programming Languages

- Python 3.8+ (For the machine learning model and FastAPI backend)
- JavaScript (ES6+) (For the front-end and Node.js backend)
- HTML5 & CSS2 (For front-end development)

3. Backend Technologies

- FastAPI (For handling API requests and image processing)
- Node.js (Latest LTS version) (For managing the database and backend logic)
- Express.js (For handling API routes in Node.js)

4. Frontend Technologies

- React.js (For building the web interface)
- CSS2 (For UI design and responsiveness)

5. Database

- MongoDB (NoSQL) (For storing user data, pothole detection history, and image metadata)
- MongoDB Compass (a GUI tool to access MongoDB)

6. Machine Learning & Image Processing

- TensorFlow / PyTorch (For training and running the pothole detection model)

- PIL (For image preprocessing and analysis)
- NumPy, Pandas (For data handling and processing)
- ONNX (for storing and using the cnn model)

7. Authentication & Security

- JWT (JSON Web Tokens) (For secure authentication)
- bcrypt (For password encryption)

8. API & Server Management

- Postman (For API testing and debugging)
- Uvicorn (For running the FastAPI backend)

9. Development & Deployment Tools

- Git & GitHub (For version control)

10. Additional Libraries & Dependencies

- Fast API (For making HTTP requests in the frontend)
- Mongoose (For MongoDB database management)
- Po (For image handling in Python)

Hardware Requirements

Minimum Hardware Requirements (For Basic Development & Testing)

- Processor: Intel Core i5 (8th Gen) / AMD Ryzen 5 or equivalent
- RAM: 8 GB
- Storage: 256 GB SSD (or HDD with slower performance)
- GPU: Integrated graphics (not suitable for training but sufficient for running pre-trained models)
- Internet: Stable broadband connection for API calls and database interactions
- Display: 1080p resolution for clear UI development

Functional Requirements

Data Collection and Storage

The pothole detection system relies on structured data collection and efficient storage mechanisms to ensure accurate predictions and seamless retrieval of past results.

1. Data Collection

The primary source of data is user-uploaded images captured using mobile phones, dashcams, or surveillance cameras. These images are sent to the backend for processing and prediction. The system may also integrate publicly available datasets or government road monitoring data for model training and improvement. Additional metadata such as GPS location, timestamp, user ID, and device information is collected along with the images to provide context and enhance detection accuracy.

2. Data Structure

The collected data is stored in a structured format, ensuring efficient retrieval and analysis.

- Image Data: Stored in the database in the form of a base64 string.
- Metadata: Stored in a NoSQL database (MongoDB) with the following fields:
    - Image ID specific to each user (Unique identifier)
    - User ID (For tracking submissions)
    - File Path/URL (Location of stored image)
    - Timestamp (Date and time of submission)
    - Address
    - Probability
    - Detection Result (Pothole detected)
    - Description (As given by the user)

3. Storage Mechanism

- Database: MongoDB for flexible schema and fast querying
- Session Storage: Temporary caching on the server for quick access before long-term storage.

By defining a clear data source and structured storage approach, the system ensures efficient processing, retrieval, and historical tracking of pothole detection data.

Data Preprocessing

1. Image resizing

The image is resized to 28x28 and converted to greyscale for the best results to standardize the input to the cnn model.

2. Image transformation

The pixel values of the image ranging from 0 to 255 are scaled from [0,1] making them suitable for processing. The image is then converted to tensor.

Model Development

The system leverages Convolutional Neural Networks (CNNs) due to their exceptional performance in image classification and object detection tasks. The cnn models are considered based on their efficiency, accuracy, and deployment feasibility.

1. Model Training Process

The model undergoes extensive training using a labelled dataset of pothole and non-pothole images. This includes the following steps:

- Dataset preparation:
  Acquiring around 6000 images of labelled pothole and non-pothole images such that the dataset isn't unbalanced. Applying data augmentation techniques like flipping, rotation, brightness adjustment, etc to improve generalization. Then splitting the dataset into training (80 %) and testing (20 %) sets.
- Loss function & Optimizer:

  Binary Cross Entropy was used as the loss function along with momentum based gradient descent with a batch size based on available GPU memory for efficient training.

- Hyperparameter tuning:
  Learning rate optimization starting with 0.001 and randomized weights and biases to avoid saddle point and vanishing gradient descent.

- Model Architecture:
  The number of layers in the model along with the activation functions like ReLu, Sigmoid etc are decided and architecture is finalized depending on the needs.

2. Model Testing

After the training is done and loss is minimized the testing images are given to the model to check the accuracy and if the accuracy is low this testing and training process is repeated.

Model Deployment

After training and evaluation, the best-performing model is deployed using FastAPI for backend integration. The model is exported using ONNX for efficiently exporting the model architecture and the weights and biases of every layer in the model.

Non-Functional Requirements

Performance Criteria

1. Accuracy & Model Performance

- Detection Accuracy: The model should achieve at least 90% accuracy in classifying potholes vs. non-potholes.
- Precision & Recall: The system must maintain a precision of at least 85% and a recall of at least 80% to minimize false positives and false negatives.

2. Availability & Reliability

- System Uptime: The application must maintain 99.9% uptime to ensure uninterrupted access.
- Fault Tolerance: The system should handle server failures gracefully, with auto-recovery mechanisms in place.
- Backup & Data Integrity: Daily automated backups should be in place to prevent data loss.

3. Cross-Platform Performance

- The system should run smoothly on desktop with responsive design.
- The model should be optimized for on-device inference on mobile platforms using PyTorch \ ONNX.

Security

To maintain the integrity, confidentiality, and availability of user data, the pothole detection system implements robust security measures. These measures ensure that user-submitted images, metadata, and authentication details remain protected from unauthorized access and cyber threats.

1.User Authentication & Access Control

JWT-Based Authentication:

- The system uses JSON Web Tokens (JWT) for secure user authentication, ensuring that only authorized users can access sensitive data.

- Tokens are encrypted and have an expiration time to prevent misuse.

2. Data Encryption

Hashed Passwords:

- User passwords are hashed using bcrypt before being stored in the database, ensuring they cannot be easily compromised.