Comparative Analysis of Traditional and Deep Learning Approaches for Road Pothole Detection

Computer Vision - Major Project

Project Report

Prepared By-

- Akansha Gautam (M23CSA506)
- Anchit Mulye (M23CSA507)
- Om Prakash Solanki (M23CSA521)
- Shyam Vyas (M23CSA545)

Table of Contents

1. Objective

The main objective of this project is to develop an intelligent computer vision system capable of detecting potholes on roads using images and video data. The system leverages both traditional computer vision techniques and modern deep learning approaches to detect potholes with high accuracy. A comprehensive comparative analysis between these methodologies is also performed based on multiple metrics such as accuracy, computational complexity, and real-time usability. This project aims to provide an automated solution that can assist municipal authorities in road maintenance and reduce the risk of accidents due to undetected potholes.

2. Motivation

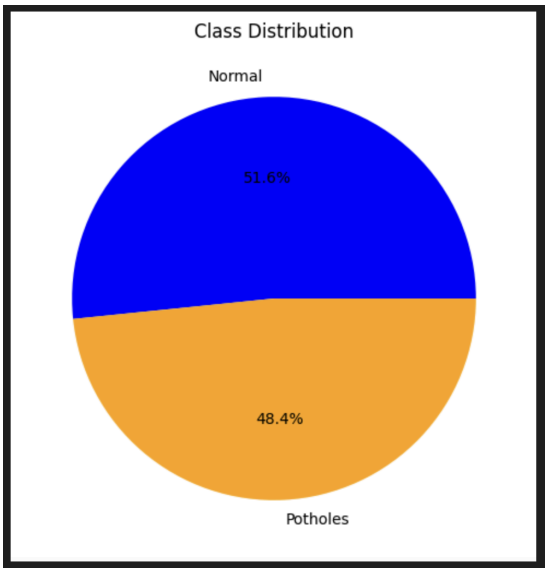
India suffers from a severe issue of poor road conditions, primarily due to potholes, leading to numerous accidents annually. Manual surveys for identifying potholes are inefficient, costly, and time-consuming. An automatic detection system that uses roadside cameras or vehicle dashcams can revolutionize road monitoring systems. The increasing accessibility of affordable camera sensors and advances in machine learning offer a perfect opportunity to implement such intelligent systems for public infrastructure monitoring. Additionally, with the growth of autonomous vehicles and smart transportation, pothole detection is becoming a key feature in ensuring safe navigation and proactive maintenance planning.

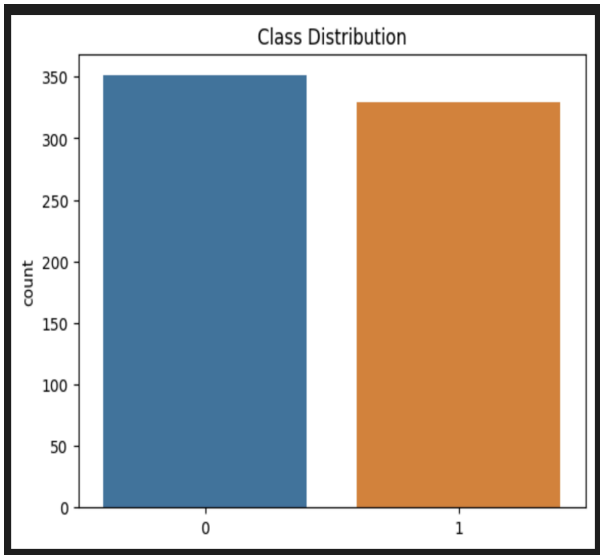
3. Dataset

The dataset used for this project includes images from multiple sources. Two key public datasets were used: Kaggle's Pothole Detection dataset and the Indian Pothole Dataset. These datasets consist of diverse images of roads with and without potholes, taken in varying lighting and environmental conditions. This dataset contains 352 images of smooth roads and 329 images of roads with potholes.

Data Preprocessing steps included:
- Resizing all images to 256x256 pixels

- Normalization of pixel values to range [0, 1]
- Data augmentation using random rotations, flips, zoom, and contrast adjustments
- Label encoding for classification





Sample Images -

Normal Images -

Pothole Images -

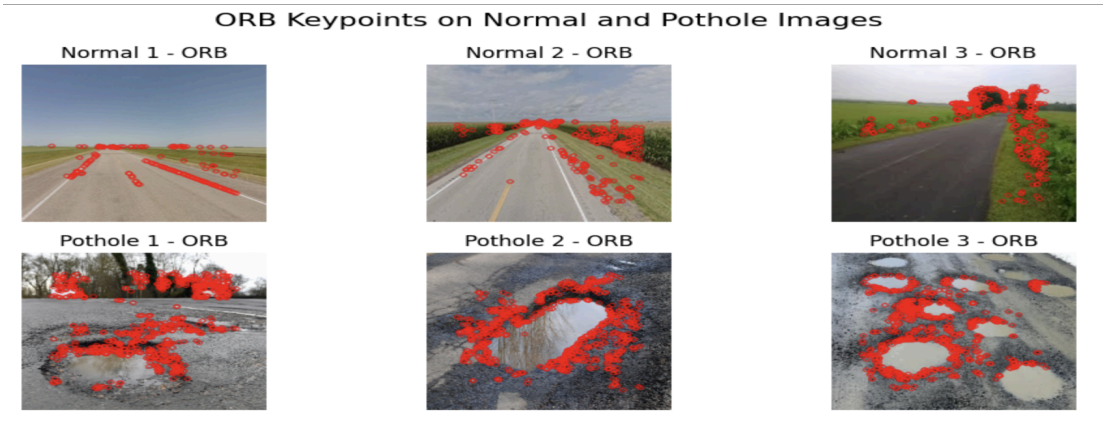
4. Approach Used

We implemented the following techniques for pothole detection:

4.1 Traditional CV Approach - ORB (Oriented FAST and Rotated BRIEF):

- **Definition:**
  - ORB is a fast, rotation-invariant feature descriptor.
  - The process included grayscale conversion, noise reduction, ORB keypoint extraction, and classification using an SVM.
  - Lightweight and suitable for edge devices but suffers in low-contrast or cluttered backgrounds.
- **Architecture:**
  - Detects keypoints using the FAST algorithm.
  - Computes binary descriptors using BRIEF.
  - Matches descriptors using brute-force or FLANN matcher.
  - A classification stage (e.g., SVM) is used to label pothole/non-pothole regions.
- **Advantages:**
  - Very fast and low-compute; ideal for embedded systems.
  - Rotation-invariant features.
  - Open-source and patent-free.

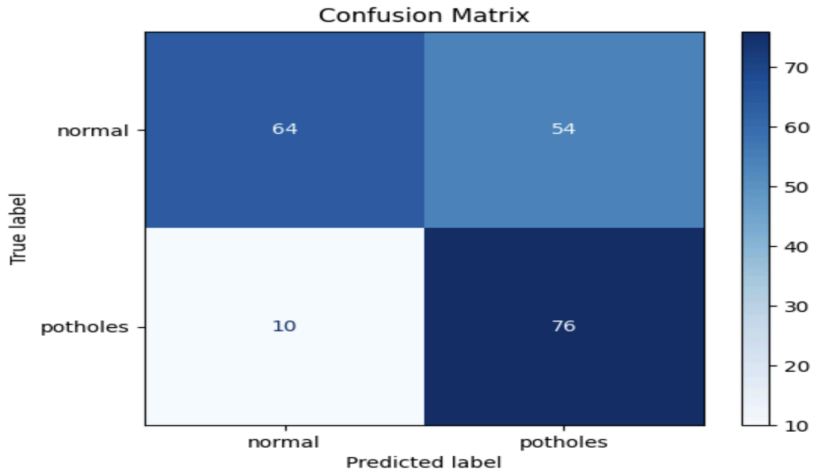
- **Limitations:**
  - Sensitive to illumination changes.
  - Struggles with noisy, blurred, or low-contrast images.
  - Limited descriptive power compared to deep learning.
- **Sample Images:**



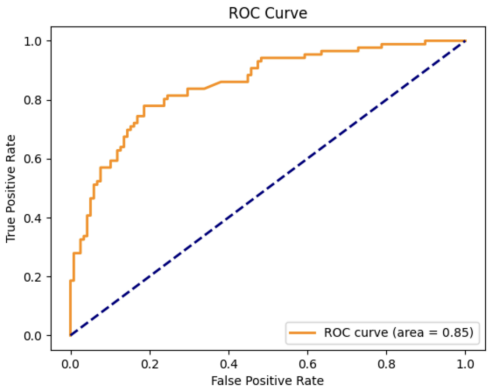
- **Classification Report:**

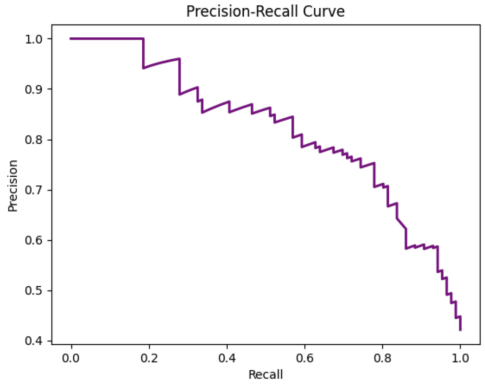
```
Clustering 500007 descriptors into 1000 clusters...
✅ Final Accuracy: 83.09%

Classification Report:

              precision    recall  f1-score   support

      normal       0.96      0.77      0.85        86
     pothole       0.70      0.94      0.80        50

    accuracy                           0.83       136
   macro avg       0.83      0.85      0.83       136
weighted avg       0.86      0.83      0.83       136
```

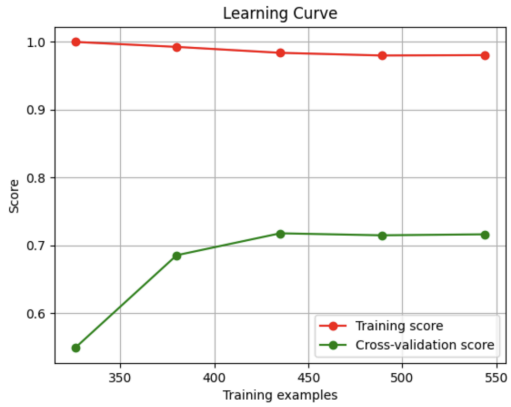
- **Confusion Matrix:**

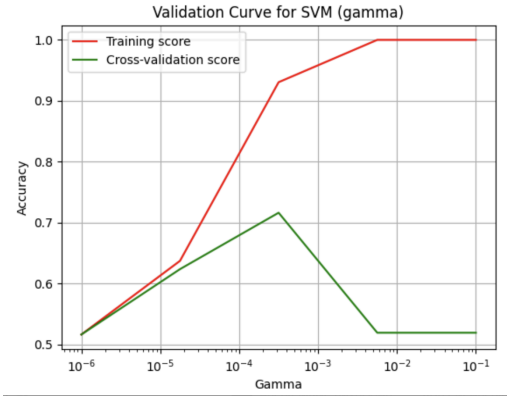


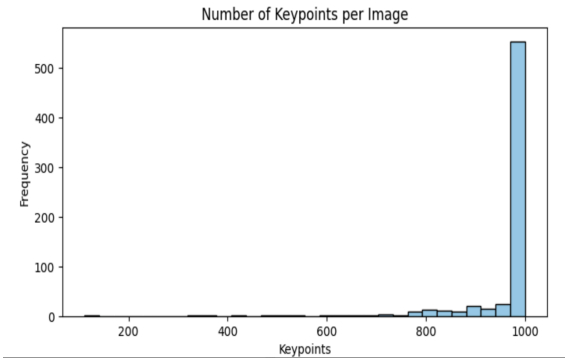
- **Curves: & Bars**

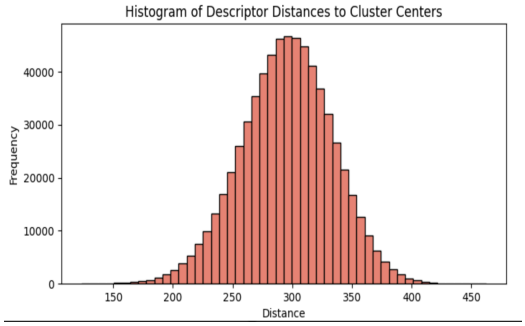








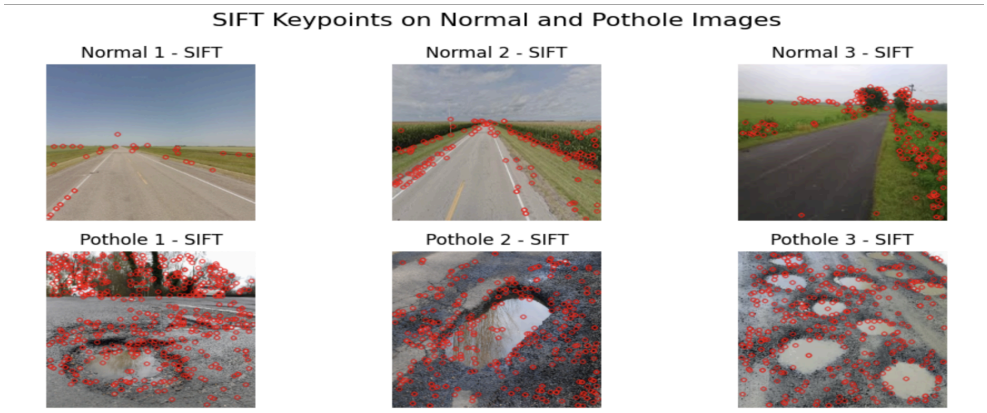




4.2 Traditional CV Approach - SIFT (Scale-Invariant Feature Transform):

- **Definition:**
  - We first extracted SIFT descriptors from each image.
  - Then, we applied K-Means clustering to build a Bag of Words model, converting each image into a fixed-length histogram based on visual words.
  - These histograms were used as input to train a Support Vector Machine (SVM) for binary classification.
- **Architecture:**
  - Keypoints detected using the Difference-of-Gaussians (DoG) method
  - Each keypoint described by a 128-dimensional vector
  - Descriptors matched using FLANN or brute-force matcher
  - Histogram of visual words is passed to an SVM classifier
- **Advantages:**
  - Invariant to changes in scale and rotation
  - Provides detailed and robust feature descriptors
  - Performs well even in complex or cluttered scenes
- **Limitations:**
  - Computationally heavier than ORB
  - Slower inference, not ideal for real-time deployment

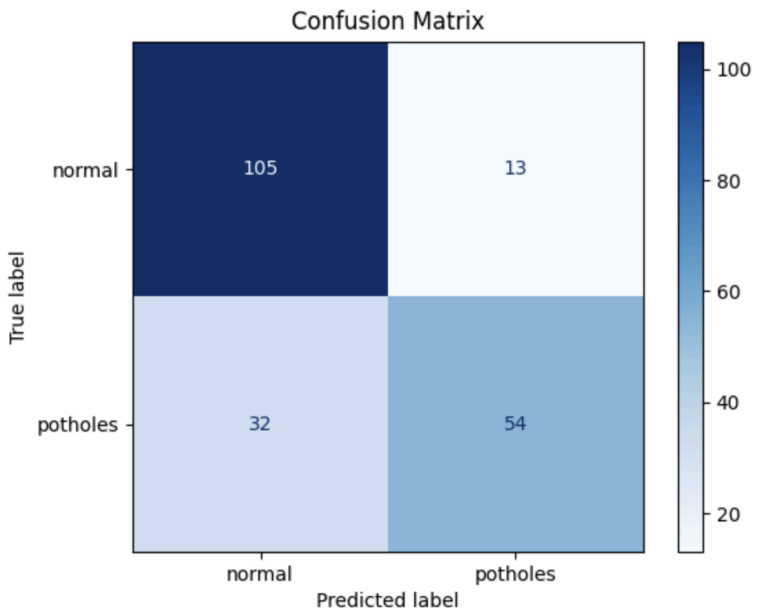
- **Sample Images:**



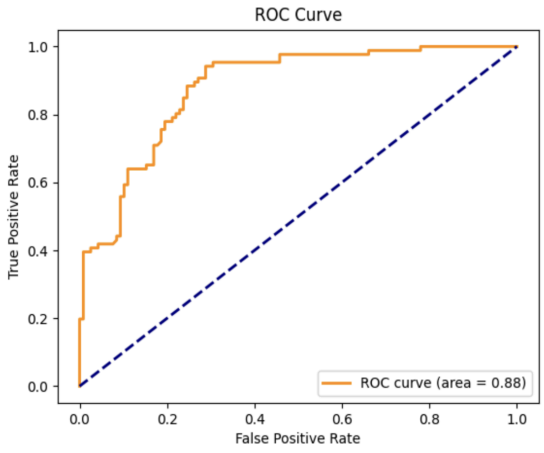
- **Classification Report:**

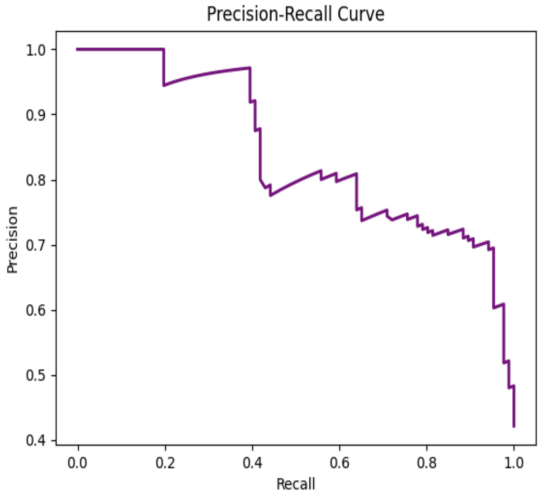
```
Clustering 308278 descriptors into 1000 clusters...
✅ Final Accuracy: 89.71%

Classification Report:

              precision    recall  f1-score   support

      normal       0.96      0.87      0.91        86
     pothole       0.81      0.94      0.87        50

    accuracy                           0.90       136
   macro avg       0.89      0.91      0.89       136
weighted avg       0.91      0.90      0.90       136


Process finished with exit code 0
```

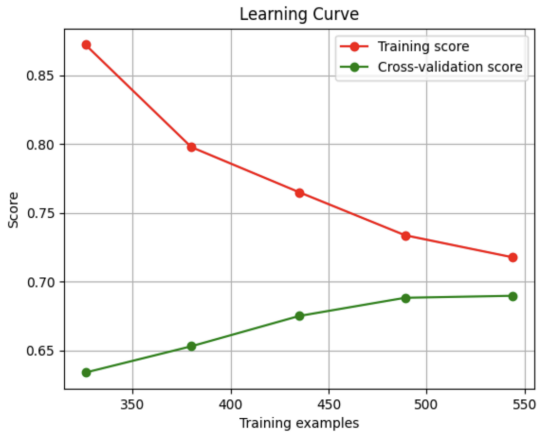
- **Confusion Matrix:**

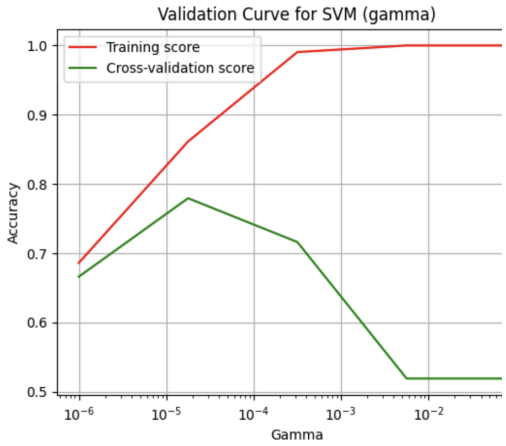


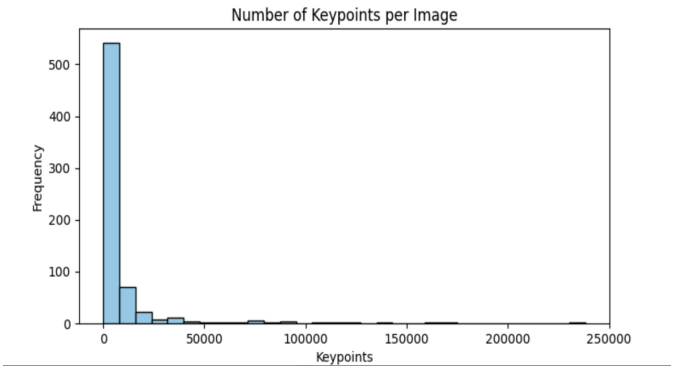
- **Curves: & Bars**

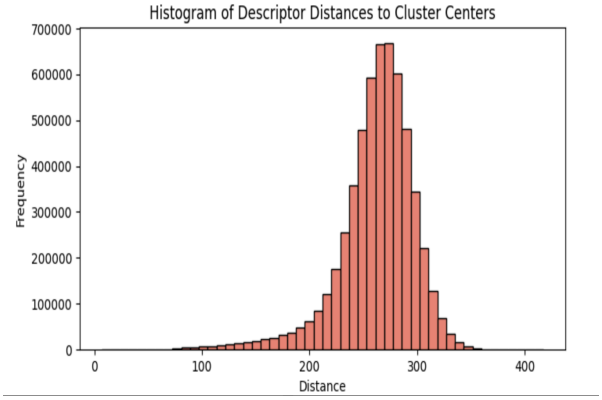








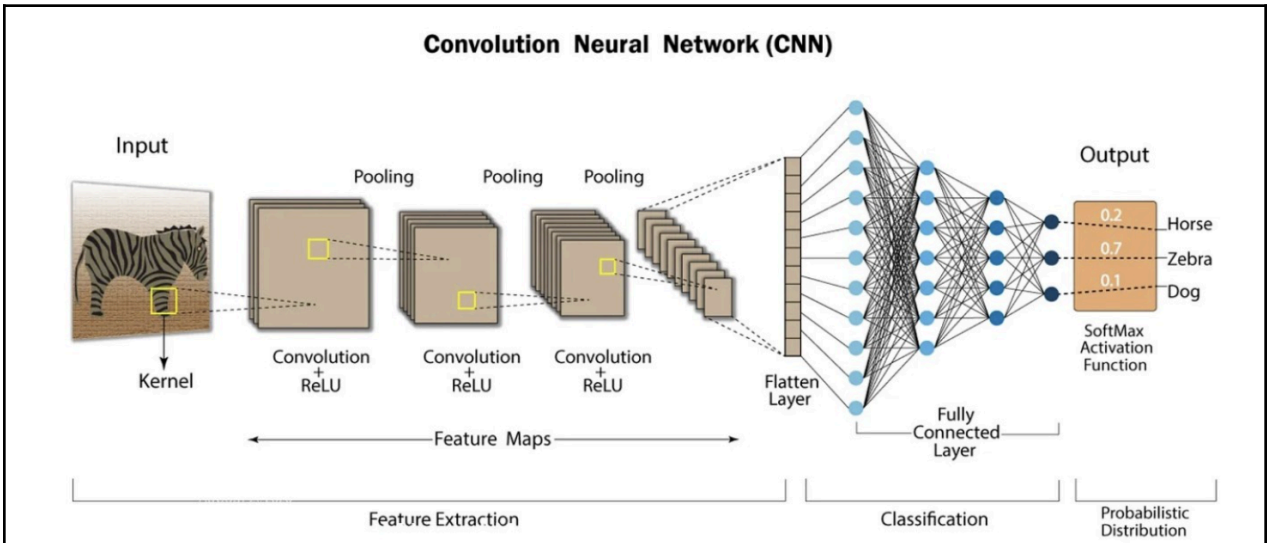




4.3 Deep Learning - Baseline CNN:

- **Definition:**
  - A simple CNN model with:
    - 3 convolutional layers
    - MaxPooling and Dropout layers
    - Fully connected layers at the end
  - Trained on 256x256 pothole images with binary cross-entropy loss.
  - Achieved decent accuracy but lacked fine-grained feature understanding.
- **Architecture:**
  - Input layer: 256x256 RGB images.
  - Conv Layer 1: 32 filters, 3x3 kernel → ReLU → MaxPooling.
  - Conv Layer 2: 64 filters → ReLU → MaxPooling.
  - Conv Layer 3: 128 filters → ReLU → MaxPooling.

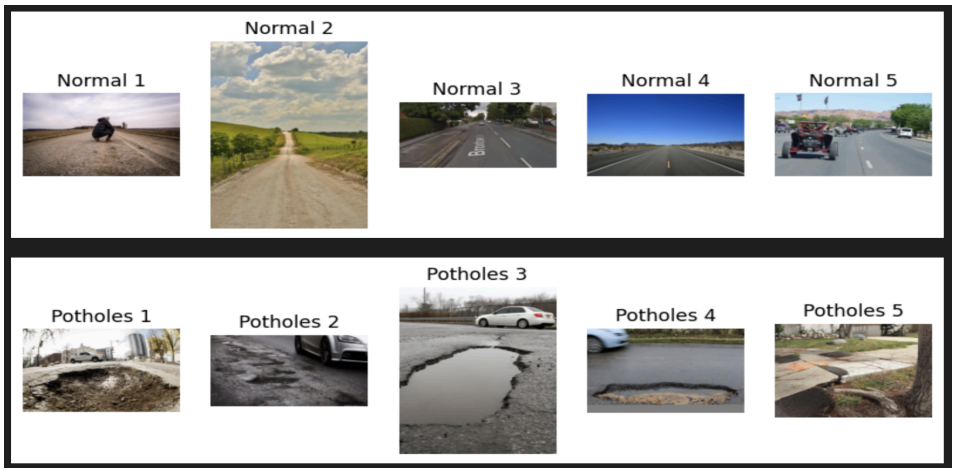
- Flatten → Dense(128) → Dropout(0.5) → Output(Sigmoid for binary classification).



Convolutional Neural Network — CNN architecture

- **Advantages:**
  - Easy to build and train on moderate datasets.
  - Learn hierarchical features automatically.
  - Faster inference compared to transformers.
- **Limitations:**
  - Struggles with generalization on complex backgrounds.
  - Needs data augmentation to avoid overfitting.
  - Lacks long-range spatial understanding.

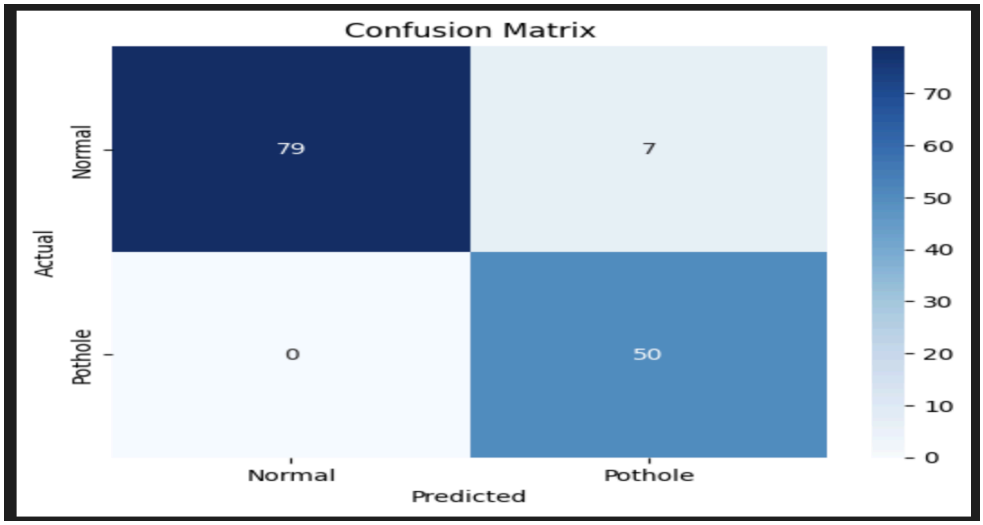
- **Sample Images:**



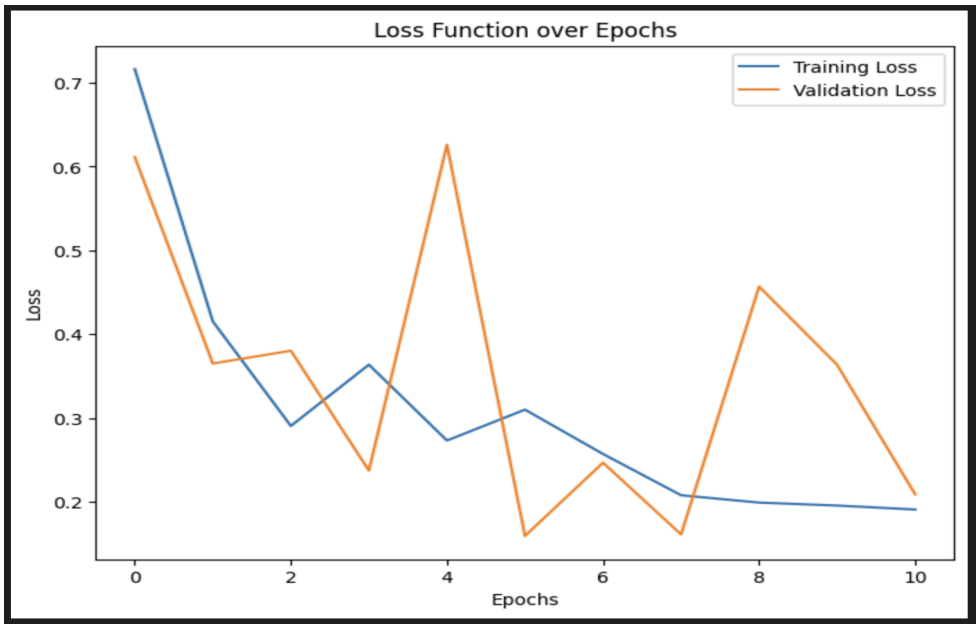
- **Classification Report:**

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        86
           1       0.88      1.00      0.93        50

    accuracy                           0.95       136
   macro avg       0.94      0.96      0.95       136
weighted avg       0.95      0.95      0.95       136

Validation Accuracy: 0.9485294117647058
```

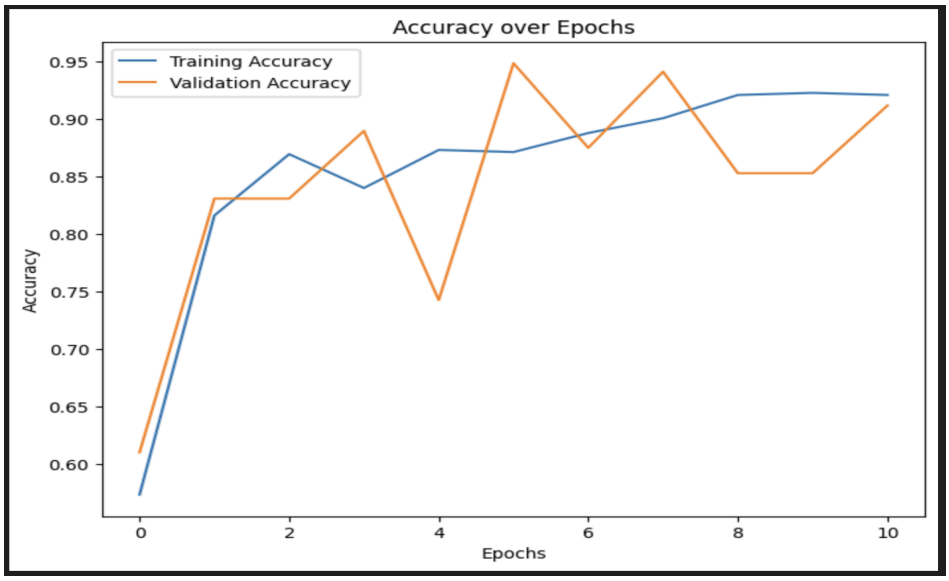
- **Confusion Matrix:**



- **Loss Function:**



- **Accuracy Graph:**

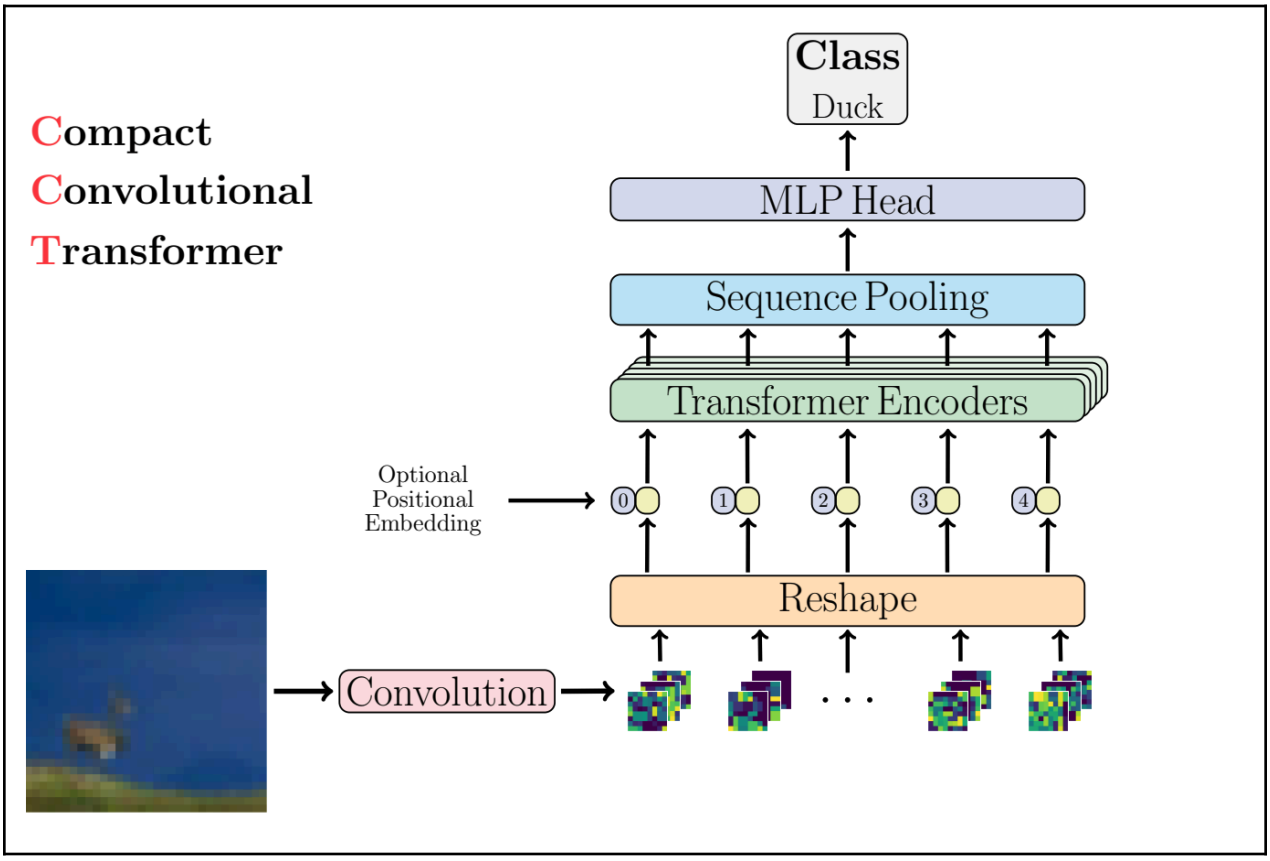


4.4 Deep Learning - Custom Compact Convolutional Transformer (CCT):

- **Definition:**
  - Combines convolutional layers with transformer encoders to capture both local and global spatial dependencies.

- Compact architecture suitable for training on mid-sized datasets without requiring large GPU memory.
- Delivered the best performance in terms of precision and generalization.

- **Architecture:**
  - **Convolutional Stem:** Extracts local features and embeds patches.
  - **Reshape Layer**: Flattens and arranges the output feature maps into a sequence format suitable for transformer input.
  - **Transformer Encoder:** Uses self-attention to capture global dependencies.
  - **Sequence Pooling**: Aggregates the token information into a single vector representation
  - **MLP Head** for final classification.



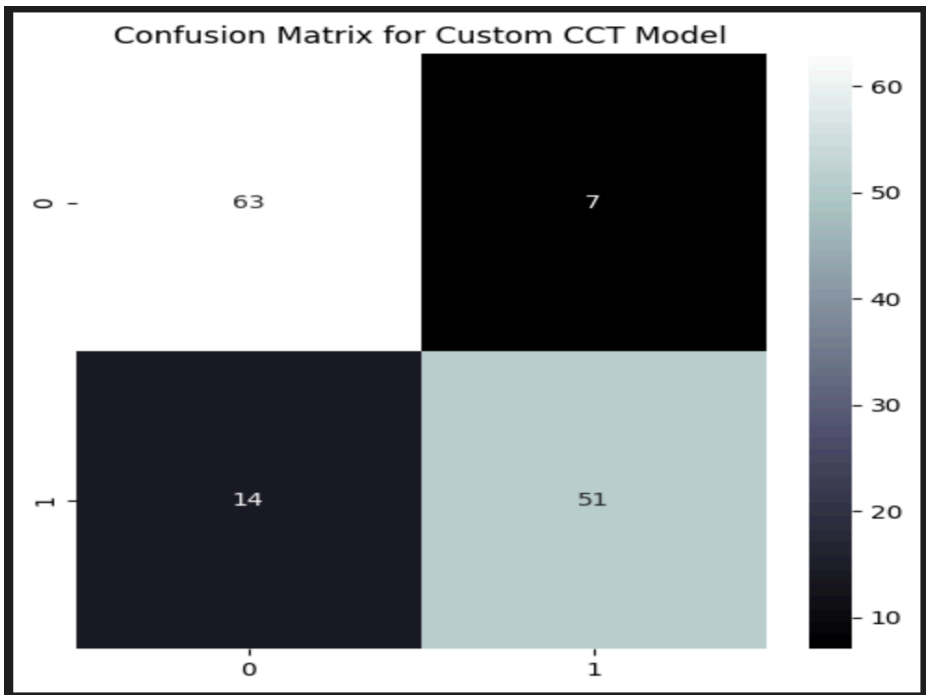
- **Advantages:**
  - Combines local feature strength of CNNs with global reasoning from Transformers.
  - Better generalization to unseen data and complex patterns.

    - Relatively compact and trainable on mid-sized datasets.
- **Limitations:**
    - Requires more compute during training.
    - Model tuning (e.g., attention heads, depth) is non-trivial.
    - Slightly longer inference time than CNN.

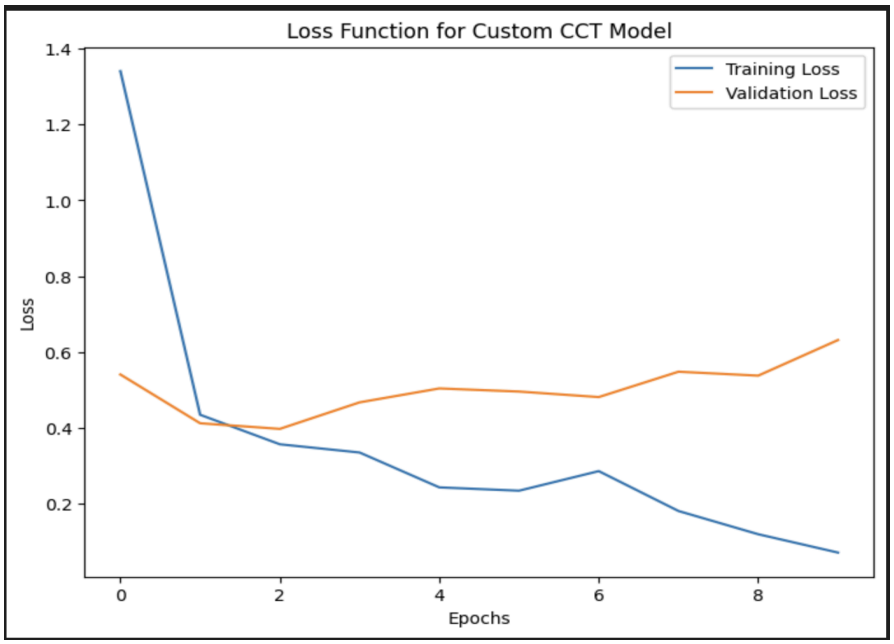
- **Classification Report:**

```
Classification Report for Custom CCT Model:
              precision    recall  f1-score   support

           0       0.82      0.90      0.86        70
           1       0.88      0.78      0.83        65

    accuracy                           0.84       135
   macro avg       0.85      0.84      0.84       135
weighted avg       0.85      0.84      0.84       135
```

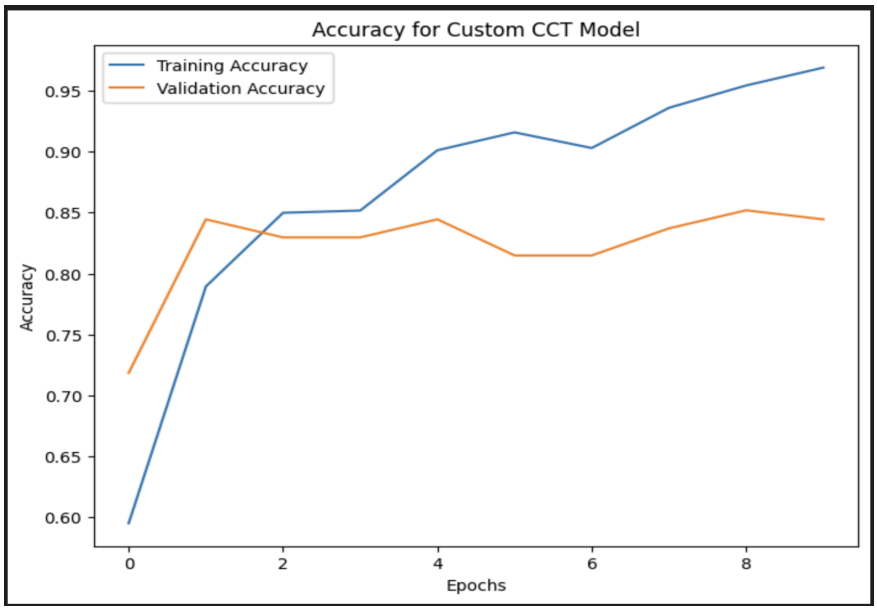
- **Confusion Matrix:**



- **Loss Function:**



- **Accuracy Graph:**



5. Model Evaluation Metrics

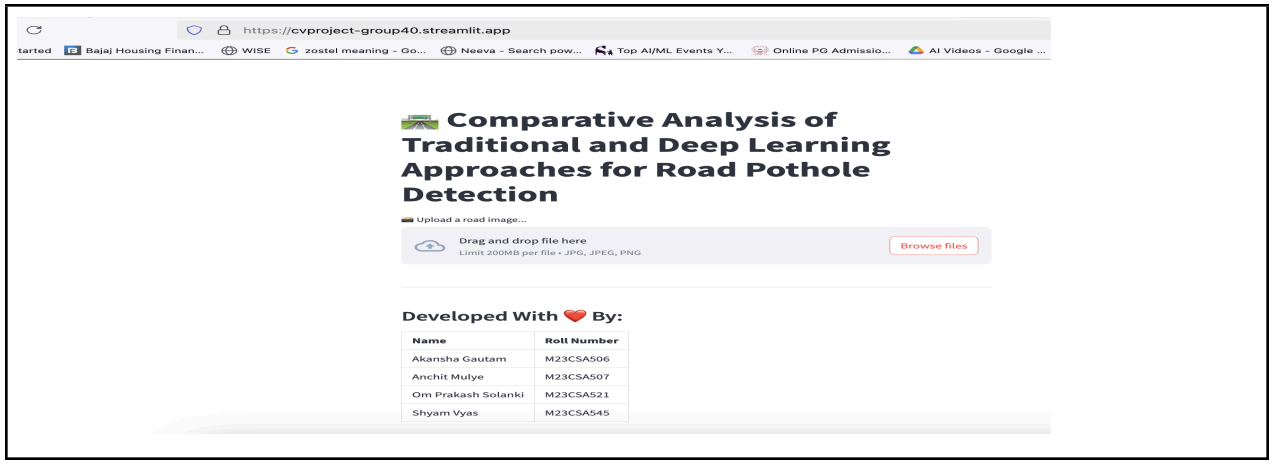
To ensure a fair comparison among the different models, the following metrics were used:

- Accuracy: Percentage of correct predictions
- Precision: True Positives / (True Positives + False Positives)
- Recall: True Positives / (True Positives + False Negatives)
- F1-Score: Harmonic mean of precision and recall

Sample Images		ORB + SVM	SIFT + SVM	CNN	CCT
Pothole Img 1	Prediction	POTHOLES	POTHOLES	POTHOLES	POTHOLES
	Confidence	89.88%	94.56%	91.64%	88.11%
Pothole Img 2	Prediction	POTHOLES	POTHOLES	POTHOLES	POTHOLES
	Confidence	60.80%	98.67%	56.73%	78.01%
Normal Img 1	Prediction	NORMAL	NORMAL	NORMAL	NORMAL
	Confidence	94.65%	75.63%	99.72%	97.95%
Normal Img 2	Prediction	NORMAL	NORMAL	NORMAL	NORMAL
	Confidence	92.93%	75.63%	99.84%	99.90%

6. Deployment Link and Screenshots

Deployment Link: https://cvproject-group40.streamlit.app/



7. GitHub Repository

GitHub Repository: https://github.com/anchitmulye-iitj/CV_Project

8. Member Contributions

Akansha Gautam: **Traditional CV SIFT Approach** - Worked on the traditional ML approach. Used Scale-Invariant Feature Transform (SIFT) to extract the image features. Trained the SVM model on the extracted features and analyzed the trained model's performance. Also, worked on the presentation required for the project's video submission.

Anchit Mulye: Traditional ORB+SVM Approach - Designed, trained, tested, and fine-tuned the ORB+SVM model, analyzed confusion matrix, configured Streamlit for deployment, and managed hosting and reporting.

Om Prakash Solanki - Developed, trained, tested, and fine-tuned the **Compact Convolutional Transformer (CCT)** model for pothole detection, performed data preprocessing and augmentation, evaluated performance using classification metrics, configured the model for efficient inference, and contributed to deployment on Streamlit and project documentation.

Shyam Vyas: Developed, trained, and optimized a **Convolutional Neural Network (CNN)** model for pothole detection; carried out data preprocessing and augmentation, evaluated performance using accuracy and confusion matrix, fine-tuned model parameters for improved results, and worked on deployment and project documentation and report.

9. Conclusion

Through this project, we explored both traditional computer vision techniques and deep learning models to detect road potholes using images. We started with classic approaches like ORB and SIFT, which were relatively simple to implement and worked

well in some scenarios. However, we noticed that they struggled with noisy images, complex backgrounds, and varying lighting conditions.

When we moved to deep learning, the results improved significantly. The baseline CNN already performed better than traditional methods, and the Custom Compact Convolutional Transformer (CCT) gave us the best results overall. It was able to understand both local textures and the overall structure of the road, which helped in making more accurate predictions.

From our comparison, it's clear that deep learning models, especially the transformer-based architecture, are more powerful and generalize better to new images. That said, traditional models still have value in low-resource settings due to their speed and simplicity.

10. Future Work

- Integration with GPS for geo-tagged pothole reports. Implement cloud-based solutions and edge computing for scalable data processing.
- Extending detection to include cracks, manholes, and road signs.
- Use LiDAR and radar sensors for better pothole detection.
- Combine data from multiple sensors for robust detection.
- Optimize SVM models with hyperparameter tuning and feature selection.
- Enable real-time data sharing between vehicles and infrastructure.
- Develop a mobile app for user reporting and community engagement.
- Implement edge detection algorithms for accurate pothole identification.
- Use predictive analytics to forecast pothole development and schedule maintenance.
- Combine SIFT and ORB algorithms to create a hybrid model for improved detection.

11. References

- Kaggle Pothole Image Dataset
https://www.kaggle.com/datasets/shubhamgoel27/pothole-image-dataset
- OpenCV Documentation - https://docs.opencv.org/

- Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints." International Journal of Computer Vision, 60(2), 91-1101.
- Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). "ORB: An efficient alternative to SIFT or SURF." IEEE International Conference on Computer Vision (ICCV)2.
- Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks." Machine Learning, 20(3), 273-2973.
- Koch, C., & Brilakis, I. (2011). "Pothole detection in asphalt pavement images." Advanced Engineering Informatics, 25(3), 507-5154.
- Jahanshahi, M. R., Masri, S. F., Padgett, C. W., & Sukhatme, G. S. (2013). "An innovative methodology for detection and quantification of cracks through incorporation of depth perception." Machine Vision and Applications, 24(2), 227-2415.
- Safyari, Y., Mahdianpari, M., & Shiri, H. (2024). "A Review of Vision-Based Pothole Detection Methods Using Computer Vision and Machine Learning." Sensors, 24(17), 56526.