



Real-time speed bump detection using YOLOv8 and embedded vision for adaptive braking assistance in electric vehicles

Ankita Gupta¹

Shripad
Deshpande²⁺

Paresh Nasikkar³

^{1,2,3}Symbiosis Institute of Technology, Symbiosis International University,
412115 Pune, Maharashtra, India.

¹Email: er.ankita0509@gmail.com

²Email: shripad.deshpande@sitpune.edu.in

³Email: paresh.nasikkar@sitpune.edu.in



(+ Corresponding author)

ABSTRACT

Article History

Received: 26 January 2026

Revised: 15 May 2026

Accepted: 3 June 2026

Published: 30 September 2026

Keywords

Adaptive speed control
Electric vehicles
Embedded vision
Real-time object detection
Speed bump detection
YOLOv8.

There are many unmarked speed bumps on our roads, which cause discomfort and damage to EVs and cars, including body damage, shock to occupants, and hard braking of the electric motor. This paper proposes a real-time embedded system for speed bump detection and a corresponding algorithm to activate the intelligent braking system of an EV, implementable in even the lowest grade of Advanced Driver Assistance Systems (ADAS). The base model used is YOLOv8-small, trained using transfer learning weights on a custom dataset of approximately 2000 images scraped from Mendeley and Kaggle, covering a wide range of bump shapes and sizes. The images are annotated in Roboflow by drawing polygons for the class "speed bump." The model is trained on Google Colab GPU for approximately 150 epochs and implemented on a Raspberry Pi 4 with 8 GB of RAM and a Pi Camera V2. The vehicle braking is controlled via a PWM signal through the L298N H-bridge driver, with GPIO-based audio-visual alerting also implemented. The YOLOv8 model achieved a precision of 0.9777, recall of 0.9977, mAP@50 of 0.9958, mAP@50-95 of 0.8577, and an F1 score of 0.99. On the Raspberry Pi 4, the system achieves a raw inference rate of 0.84 FPS (~1522 ms) and effective decision pipeline throughput of 8-10 FPS via frame-skipping. Model performance on a Mac M3 GPU achieved 43.03 FPS. The proposed system is highly scalable and cost-effective, deployable as a low-cost ADAS module in EVs to improve passenger safety and extend vehicle life across varied road surfaces.

Contribution/Originality: This paper's primary contribution is finding that a YOLOv8-small model deployed on a Raspberry Pi 4 enables real-time speed bump detection with mAP@50 of 0.9958 and end-to-end GPIO-based adaptive braking actuation, delivering a low-cost, scalable embedded ADAS solution for electric vehicles operating raw inference rate of 0.84 FPS (~1522 ms) and effective decision pipeline throughput of 8-10 FPS via frame-skipping.

1. INTRODUCTION

Road safety is still a very real issue on our streets and in our towns and villages, as well as on our rural roads. Many of our road bumps (the ones that are supposed to slow drivers down) are not properly marked and are wearing away, causing the car to jerk violently, potentially damaging shock absorbers and braking systems. In addition, the bumps can be quite a shock to the driver and passengers and are not fun to travel over, particularly on EVs, where they can potentially affect the safety of the vehicle and its occupants. In normal conditions, the bumps in the road are detected by the driver using their senses, but in the dark, in the rain and fog, and in heavy traffic, this is not always reliable. In the case of an Electric Vehicle, the brakes and propulsion must be controlled in real time and in a dynamic manner, which cannot happen without some form of sensing [1]. Currently, most of the road surface detection

methods are based on mechanical vibration excitation, accelerometers, GPS-based hazard marking, and infrared sensors [2]. Even though they are economical, they suffer from many drawbacks like false alarms, less adaptability to varying road surfaces, and they cannot distinguish between numbered and unnumbered bumps, and also, they do not work under low light conditions. Even though classical computer vision techniques like edge detection and optical flow can help to some extent, they are also not immune to changes in light and object occlusion [3].

Object detection by deep learning models has enabled real-time road hazard detection. There are many deep learning-based object detection algorithms proposed in the literature, and the YOLO (You Only Look Once) family of detectors is one of the most widely used. In this work, real-time road hazard detection is realized based on the YOLOv8 algorithm. The results show that the mean Average Precision (mAP) and frames-per-second (FPS) of YOLOv8 are larger than those of state-of-the-art models, YOLOv5 and YOLOv7, and the lightweight versions of YOLOv8 can be deployed on edge devices [4]. Most of the works based on YOLO models in the literature [5, 6] are tested on embedded devices without exploring the full actuation of the brake. Also, a few works explored GPIO-level PWM motor control and latency adaptation together with edge AI-based YOLO detection and a user-friendly deployment interface. All the current works in the literature have their own set of drawbacks. Works that heavily depend on accelerometer and GPS sensor data do not achieve high accuracy since the accelerometer and GPS sensor readings are highly noisy and vary with different aspects of the vehicle motion [7, 8]. Works that depend on vision-based speed bump detection have also shown highly decreased accuracy under low lighting conditions and heavy occlusions [3, 9]. Deep learning models are known to be highly accurate given that the environment they are designed to work in is well understood. Deep learning-based speed bump detection methods are still not fully integrated with the actuation system required to control the braking system of autonomous vehicles [6, 10]. Moreover, most of the deep learning-based methods cannot be deployed in real time on limited resources, and the latency of the system must be adapted to the real-time braking operation required for safe autonomous driving. In our work, we favor YOLOv8 over SSD and YOLOv5 models, thanks to the higher mAP value, lower inference latency, and support for the polygonal annotation format. In addition, the optimization pipeline provided by Ultralytics for the YOLOv8 model and the NCNN edge-based deployment.

This paper's contributions are as follows:

- Using a pre-trained YOLOv8-small model and retraining it on our custom dataset of 2000 images with polygon annotations, we can achieve an mAP@50 of 0.9958 and an F1 score of 0.99 for the detection of speed bumps in all weather conditions.
- Full end-to-end embedded ADAS system on a Raspberry Pi 4 from object detection to GPIO-controlled PWM motor actuation. Demonstrated raw inference rate of 0.84 FPS (~1522 ms) and effective decision pipeline throughput of 8–10 FPS via frame-skipping.
- The accuracy can be significantly enhanced by a novel post-processing method. An executable code file and an example image are provided to demonstrate how to use the method. Performance comparison on Raspberry Pi 4 and Apple M3 GPU platforms.
- Streamlit interface for real-time confidence.
- This work presents the experimental validation of a closed-loop system for obstacle detection by sending acoustic and optic alert signals and by slowing down the robot movement until it stops. The system then restores the normal speed of movement when the obstacle is no longer detected.

In the following sections of this paper, the literature review is shown in Section II, the problem statement and the constraints of the system proposed in this work are described in Section III, the proposed method is explained in detail in terms of the data set preparation, network training, and the control system applied to the motor in Section IV. The accuracy of the proposed system and the results obtained in the experiments developed to evaluate the real-time performance of the proposed method are shown in Section V. Finally, the conclusions of the work are presented in Section VI, and the future works are presented in Section VII.

2. LITERATURE SURVEY

In the literature, there are several proposed works that intend to identify the locations of speed bumps and other road surface irregularities using sensor-based methods, classical vision techniques, and, more recently, deep learning-based architectures. This section presents a review of the most significant works and, accordingly, their limitations, as well as an explanation of how the contributions offered in this dissertation can solve the open issues in this domain.

2.1. Approaches Based on Sensors and Crowdsourcing

Road surface anomalies such as bumps can be detected by utilising a smartphone application based on the accelerometer, as demonstrated in Edwan et al. [7]. The potential of deploying many simple monitoring devices is thereby realised. However, the accuracy of the inertial measurements is affected by changes in vehicle speed, measurement noise, and variations in road surface characteristics. The cost of inertial measurement units is also non-trivial. Hameed et al. [11] proposed an on-board data logger for real-time road anomaly detection using vehicle-mounted sensors, though adaptability to diverse road surfaces remained limited. Wang et al. [12] proposed a road quality detection method using the Mahalanobis-Taguchi system; however, the method is computationally intensive and sensitive to sensor noise. Joon et al. [8] demonstrated that an accuracy of 80.9% can be achieved in classifying data collected from a GPS and wheel sensors by employing a random forest classifier. However, the use of a structured GPS signal is not feasible in environments where the GPS signal strength is poor or non-existent. Zheng et al. [13] achieved an F1 score of 92.45% using a fast filter-based dynamic time-warping approach for road anomaly detection. However, this approach is unable to classify visually similar road objects, such as different types of curves, and requires image features.

Crowdsourcing has also been explored. Chen et al. [14] proposed a scale-invariant anomaly detection framework for crowdsourcing heterogeneous anomalies from user-generated data. In this work, they demonstrated that scalable road monitoring without hardware deployment is possible. Yuan and Che [15] proposed a real-time hazard mapping system using acceleration data and GPS-based clustering. Although these approaches can facilitate large-scale coverage, the robustness of such approaches is highly dependent on the spatiotemporal density of crowd-sourced data and communication quality. The latter two aspects are not favorable for resource-limited, standalone embedded systems. Ramakrishnan et al. [16] proposed an IoT-based speed-breaker detection and mapping system combining accelerometer data with GPS tagging, though reliability degrades in GPS-denied environments. Ameddah et al. [17] presented a cloud-assisted road condition monitoring system using wireless communication, though the approach depends on continuous network connectivity.

2.2. Traditional Computer Vision Techniques

Bello-Salau et al. [3] presented several vision-based techniques for road anomaly detection using conventional OpenCV feature-based methods. Sirbu et al. [9] proposed a system for real-time speed bump detection using a monocular camera. The system employs a semantic segmentation model and ED Line analysis. The proposed approach is very efficient and can be easily implemented on small form-factor devices. However, as in the case of the above-mentioned works, it heavily depends on strong visual features and road markings, which in many cases lead to low performance in low light conditions (faded markings), unmarked roads, etc. In any case, both works confirm that feature-based approaches are not sufficient for reliable and dependable detection of bumps on the road surface in the presence of many real-world factors.

2.3. Detection Based on Deep Learning

Object Detection for Intelligent Braking System using Deep Learning, in Shah and Deshmukh [18] demonstrated the performance of CNN and YOLO based approaches for the detection of speed bumps. It is shown that the accuracy of the proposed approaches is found to be up to 88.9 % which is way better than the existing methods

in the literature. Tithi et al. [19] demonstrated an SSD-based approach implemented on Raspberry Pi and an ATmega328-based braking circuit. The primary aim of the paper is to demonstrate lightweight object detection for the intelligent braking system. The drawback of the approach is that it is demonstrated with a very limited dataset. The performance of the proposed approach does not validate the scalability of the inference time for different speeds and robustness to different road conditions.

Dewangan and Sahu [5] obtained 98.54% accuracy using a compact CNN, and experiments were carried out on a Raspberry Pi to demonstrate real-time object detection using low-cost hardware. In Dewangan et al. [6], they extended their work by designing a multilayer CNN known as SBDNet (Single Branch Deep Net), which uses region proposals and non-maximum suppression to accurately detect vehicles on various types of road surfaces with an accuracy of 99.34%; hence, SBDNet outperforms the state-of-the-art but is highly resource intensive. This is also a barrier for them to be implemented on low-cost and resource-limited embedded systems, unless some optimization is done. Dadras et al. [20] proposed a stop sign detection algorithm based on vehicle speed profiles, though the approach does not generalize to unmarked or varied road surfaces.

2.4. Comparative Summary of Prior Methods

Table 1 summarizes the key characteristics of representative prior works compared to the proposed approach.

Table 1. Comparative summary of speed bump detection methods.

Reference	Method	Platform	Accuracy/mAP	Limitation
Tithi, et al. [19]	SSD + ATmega328	Raspberry Pi	Not reported	Limited dataset, no latency analysis
Shah and Deshmukh [18]	CNN/YOLO	GPU	88.9%	No embedded actuation
Sirbu, et al. [9]	ED Line + Segmentation	Embedded CPU	Not reported	Requires strong road markings
Dewangan and Sahu [5]	Compact CNN	Raspberry Pi	98.54%	Limited environmental generalization
Dewangan, et al. [6]	SBDNet (CNN+RPN)	GPU	99.34%	High compute, not edge-ready
Joon, et al. [8]	GPS + Wheel + RF	Server	80.9%	GPS-dependent, no vision
Proposed	YOLOv8-S + PWM Actuation	Raspberry Pi 4	mAP@50: 0.9958	Single-class, 0.84 FPS raw (8–10 FPS with frame-skipping) on Pi

The review reveals that almost all recent deep learning works in the field of autonomous braking have focused on the accuracy of the system under well-defined and highly constrained conditions. However, none of them have yet demonstrated the ability to really close the loop from the computation of the braking command performed in the ECU via embedded inference to the actual braking behavior of the car. Moreover, adaptive latency management for the control loop and acceleration on hardware like Raspberry Pi-class devices have not been investigated. This work addresses these open issues.

3. PROBLEM STATEMENT

Although speed bumps are installed for security purposes, EVs may experience passenger discomfort, damage to the suspension system and brake problems when they are driven over unmarked, worn-out, or covered speed bumps. In normal circumstances, drivers would be able to feel the bumps on the road surface, but in cases where the road surface may not be visible due to factors such as driver distraction, poor lighting conditions, adverse weather or heavy traffic, the road bump may not be detected. The main purpose of this study is to develop an intelligent speed bump

detection and braking system that can be installed in EVs in order to ensure that the EVs can detect and respond to the road surface bumps in real time and, at the same time, satisfy the computational constraint of the EV control unit.

Hardware to be used: Camera, Raspberry Pi. The intended application is to have a camera mounted on the front of a vehicle, running a Raspberry Pi and analyzing the real-time video feed for speed bumps on the path ahead. Upon finding a speed bump, the motor driver module is used to brake the vehicle down to a suitable speed in a controlled braking to avoid shock. The vehicle is to run autonomously, at all times, under all light conditions, on all surfaces, with some occlusions, and on all types of speed bumps.

3.1. Limits on the System

- Computational constraints: The detection and actuation pipeline must be processed in real time on a Raspberry Pi 4 without the aid of an external GPU.
- Environmental variability: The system must be able to operate under low light, shadows, occlusions, and also in the absence of road markings.
- Hardware limitations: The fixed braking response latency of the L298N motor driver, combined with the frame rate of the Pi Camera V2, affects the overall system responsiveness.
- Bad Braking Smoothness: The braking signal cannot be too dynamic. The control algorithm has to control the reduction in the PWM in a stepwise manner in order not to brake too suddenly.
- Generalization: The system should be able to detect bumps on both marked and unmarked roads in all types of shapes, materials, and surfaces without needing retraining of the training data.

4. METHODOLOGY

This project proposes an embedded vision prototype of an Advanced Driver Assistance System (ADAS) capable of real-time speed bump detection with an actuation pipeline designed for advanced adaptive braking functionality. The system proposed in this work is composed of a sensing module and a perception module, which are independent of the actuation module, enabling possible future developments, including the connection with the vehicle bus and the utilization of high-performance processors such as Jetson-class. The system method of this work can be summarized into two stages: the detection phase and the actuation and alerts phase, as illustrated in the project pipeline shown in Figure 1.

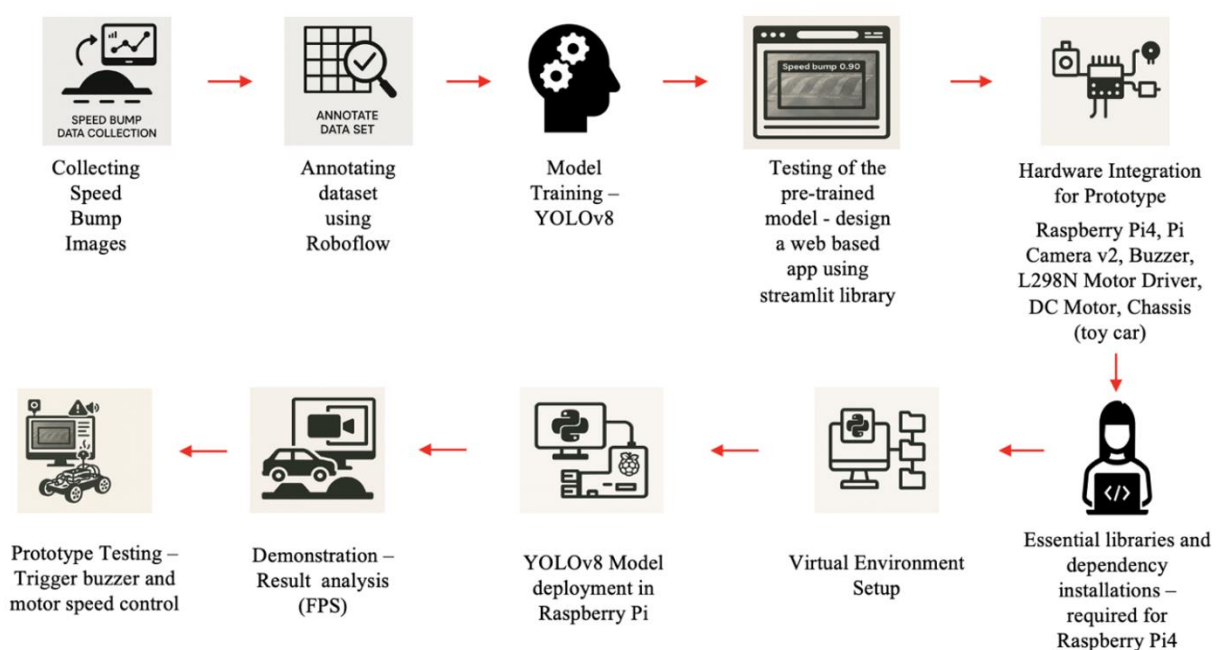


Figure 1. Project Pipeline.

4.1. System Overview

As illustrated in Figure 2, the circuit diagram and Figure 3 show the prototype of the EV safety system based on visual perception and actuation. The system takes a video stream from the camera as input and recognizes objects by detecting each frame in the video stream using the YOLOv8 algorithm. Based on the pre-defined threshold, the corresponding control command will be sent to the warning lamp or the actuation board via PWM to imitate the electronic braking function. The method in this pipeline exploits the flexibility of deep learning-based visual perception technology to connect with the actuation circuit of the EV to achieve advanced EV safety functions with low additional costs. The work in this paper also achieves a higher detection accuracy and is an end-to-end full system demonstration. In contrast, the related works in Tithi et al. [19] and Dewangan and Sahu [5] focus more on the detection performance, and only a few actuation examples are demonstrated, which limits the applications.

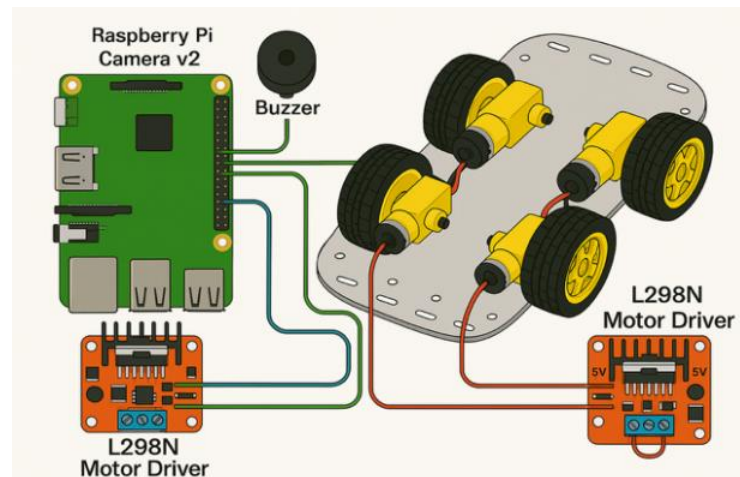


Figure 2. Circuit Diagram.

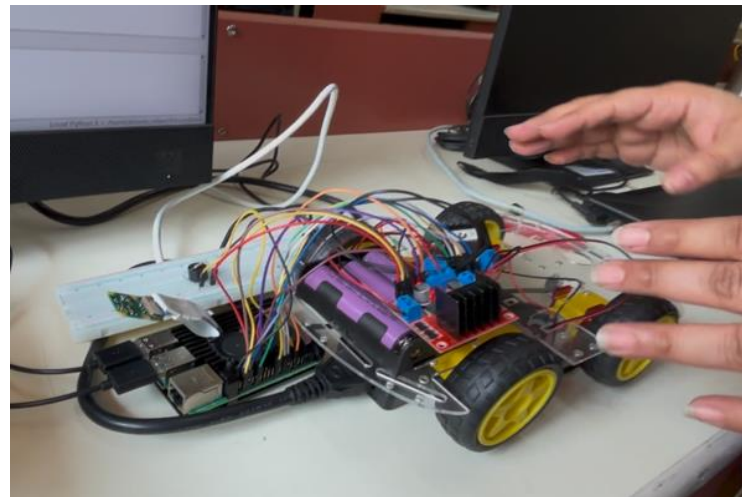


Figure 3. Hardware Prototype.

4.2. Dataset Construction and Annotation

The training set was derived from about 2,000 open images that were downloaded from Mendeley and Kaggle. These derive from a variety of road surface textures (asphalt, concrete, gravel) and several lighting conditions (daylight, overcast, shadowed) and bump patterns (marked, unmarked, painted, worn) and camera views (looking straight ahead and a little tilted). All data came from an open environment; hence no proprietary or private data was used.

There are 1600 images of confirmed speed bumps and 400 images of general road background without speed bumps for negative samples. We train our model with 80% for training, 10% for validation, and 10% for testing. We

annotate data on the Roboflow platform and mark the class “speed bump” with a polygon shape, which provides more precise information than bounding boxes for non-rectangular shapes of speed bumps.

Data augmentation: We trained the model with a number of augmentations to make it robust to some of the uncertainties that will be present when the model is driven in the real world: $\pm 10\%$ brightness adjustment, $\pm 5\%$ saturation adjustment, up to 15% random cropping, horizontal flipping to emulate potential variations in lighting and camera orientation, and some level of occlusion that might occur in the real world.

Camera Settings: The training and test images were captured as video frames using the Raspberry Pi Camera Module V2. The frames were captured at 1080p, or 1920x1080 pixels, and later down sampled to 640x640 pixels for model input. The camera was also placed in the front-facing orientation, which is the most common configuration for present-day vehicles.

4.3. Model Training and Optimization

We use the YOLOv8-small model from the yolov8s.pt file because it gives a better trade-off between precision and speed than other models based on YOLOv5 and SSD [4]. We use the model pre-trained on COCO and apply transfer learning by fine-tuning the network with our speed bump dataset. All experiments were done in the Google Colab environment using an NVIDIA T4 GPU. The model was trained for 150 epochs, with loss convergence typically observed between epochs 100 and 150. The hyperparameters were set to epochs = 150, batch size = 16, and initial learning rate = 0.01. The input resolution is set to 640x640 pixels. The validation set is used to calculate precision, recall, and mAP at each epoch to prevent overfitting of the model. Figure 6 illustrates the YOLOv8 training process initiated on Google Colab using the custom speed bump dataset, showing the training progress and loss convergence across epochs on the NVIDIA T4 GPU.

Table 2 summarizes the hyperparameter configuration used during training.

Table 2. YOLOv8 Training Hyperparameter Configuration.

Hyperparameter	Value
Architecture	YOLOv8-small (yolov8s.pt)
Pre-training	COCO weights (transfer learning)
Epochs	150
Batch size	16
Input resolution	640 × 640 pixels
Optimizer	SGD
Initial learning rate	0.01 (cosine annealing)
Momentum	0.937
Weight decay	0.0005
Training platform	Google Colab (NVIDIA T4 GPU)
Training time	~2.5 hours

4.4. Embedded Optimization

We optimized the model for performance on Raspberry Pi 4 by performing inference, which reduces memory bandwidth by 50% while still achieving top-level classification and object detection accuracy. Multi-threaded camera buffering enables the frames to be captured asynchronously with respect to the time it takes to process the neural network output, hence removing the I/O-induced frame drops. In addition, we implemented an adaptive frame-skipping mechanism, engaged when the total frame processing time (T_{total}) is greater than 100 ms. This is necessary to sustain the system throughput while preventing rare instances of slow T_{total} values from impacting real-time performance.

4.5. Algorithm and Control Logic

The adaptive braking system operates as a control loop. The YOLOv8 model, the camera stream, the motor drivers, and the GPIO buzzer are initialized. For each inference, the object detection confidence is then checked against a fixed threshold of 0.90. If a valid detection is obtained, the audio-visual alert (buzzer and LED) is triggered. The stepwise reduction of the motor speed is then obtained by changing the duty cycle of a stepwise PWM signal that acts in the same way as a normal brake pedal. Finally, when the detection is lost, the alert is switched off, and the motor speed is smoothly returned to the nominal value. The flowchart is shown in Figure 4.

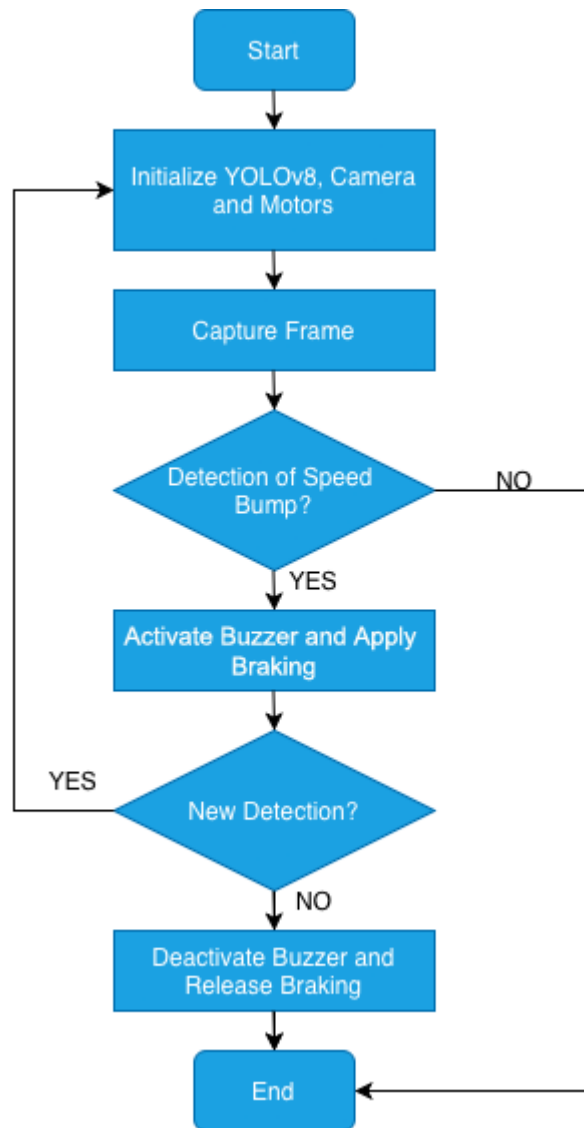


Figure 4. Flow Chart.

Algorithm Real -Time Speed Bump Detection and Braking Control

- Initialize YOLOv8 model, camera, motors (L298N), and buzzer via GPIO.
- While the system is running: Capture a frame and run YOLOv8 inference (with frame-skipping if $T_{total} > 100$ ms).
- If speed bump detected (confidence > 0.90): Activate buzzer + LED; begin step-wise PWM reduction.
- Else: Deactivate buzzer; restore nominal motor PWM.
- On system shutdown: Release motors, buzzer, camera, and clear GPIO resources.

4.6. Hardware Setup

The embedded board that we used for our experimental setup is a Raspberry Pi 4 equipped with 8 GB RAM and running on Raspberry Pi OS. We used a Pi Camera V2, and the camera frame rate is limited to the hardware maximum of 30 FPS. The L298N H-bridge motor driver board is connected to the GPIO pins of the Raspberry Pi board, and the DC motor speed control is achieved by providing a PWM signal via the L298N motor driver board. The 5V buzzer and LED is also connected to the GPIO pins of the Raspberry Pi board for an audio-visual alert system. Figure 5 illustrates the complete hardware setup of the embedded EV safety system, including the Raspberry Pi 4, Pi Camera V2, L298N motor driver, DC motor, buzzer, and LED connections. A Streamlit-based web application is implemented for the AI-enabled smart bicycle. This web application works as a user-friendly real-time interactive interface for the users. It has a live stream mode along with a slider which can be used to adjust the confidence threshold of the model as per the user's requirement by choosing the required precision and recall by sliding the value in between the range of 0.0 to 1.0, as shown in Figure 7.



Figure 5. Hardware Set Up.

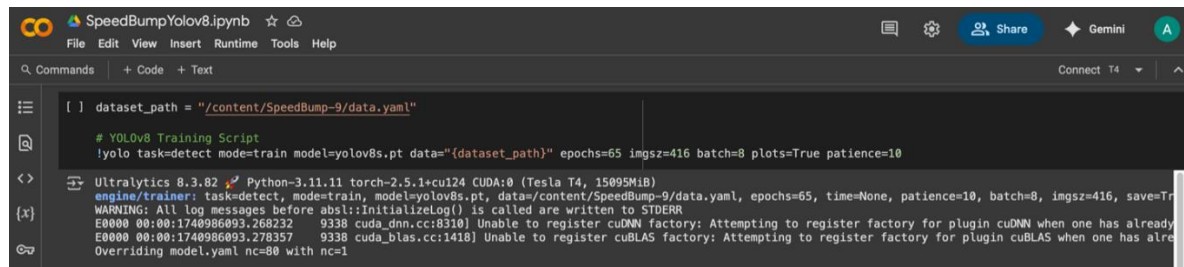


Figure 6. YOLOv8 training initiated on Google Colab using custom dataset.



Figure 7. Streamlit Web Application Interface for Speed Bump Detection.

5. RESULTS

We used standard object detection metrics to test the YOLOv8-small model on the selected test set. For all evaluations, the detection confidence threshold was set at 0.90.

5.1. Detection Performance Matrices

Precision: The model had a consistently high level of precision across all training epochs, with an average precision of 0.9777; the plot is shown in Figure 8. This shows that the system can tell the difference between real speed bumps and things on the road that look like them, like shadows, patches, and lane markings. This helps reduce false-positive braking events. Precision is defined as in Equation 1.

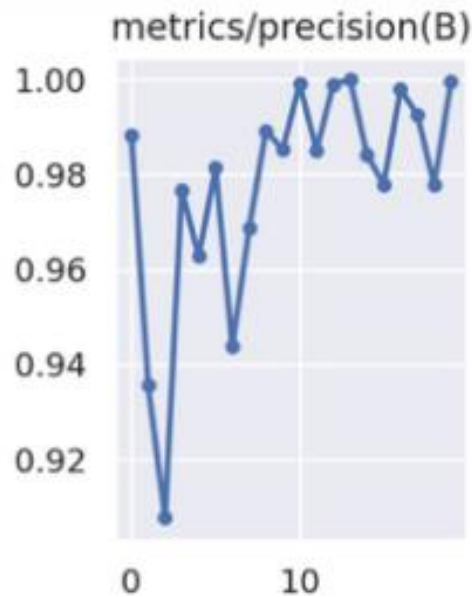


Figure 8. YOLOv8 evaluation metrics - Precision.

$$Precision = \frac{TP}{TP+FP} \quad (1)$$

Recall: The average recall was 0.99766; the plot is shown in Figure 9. A trained YOLOv8 model shows balanced and stable recall, which means that the system is capable of minimizing missing speed bumps, which is a necessary component for braking responses. Recall is defined as in Equation 2.

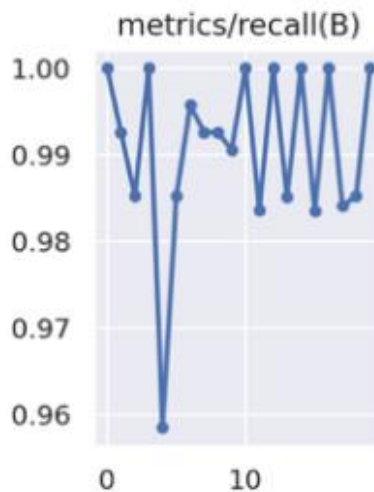


Figure 9. YOLOv8 evaluation metrics – Recall.

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

mAP@50: The model got an average mAP@50 of 0.9958; the plot is shown in Figure 10, which shows that it quickly converged and had very high localization accuracy at an IoU limit of 0.50. This metric is what makes reliable spatial detection possible for triggering brakes at the right time.



Figure 10. YOLOv8 evaluation metrics - mAP50.

mAP@50-95: The average mAP@50-95 of 0.8577 is shown in Figure 11. shows that the localization performance is strong across a range of IoU limits (0.50 to 0.95), confirming that the spatial precision is accurate even when the bumps and camera angles change.

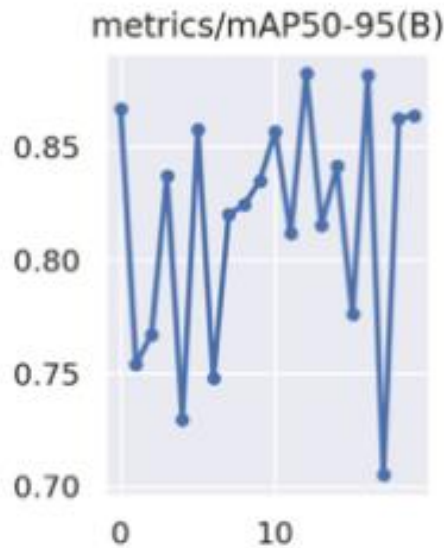


Figure 11. YOLOv8 evaluation metrics - mAP50-95.

F1 Score: An F1 score of 0.99 shows that the precision-recall differences are well-balanced. Figure 12 illustrates the F1 score curve across training epochs, demonstrating the well-balanced trade-off between precision and recall achieved by the YOLOv8 model throughout the training process (see Table 3). This is important for real-time ADAS because both false positives (unnecessary braking) and false negatives (missed bumps) can be dangerous. The F1 Score is defined as in Equation 3.

$$F1\ Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \quad (3)$$

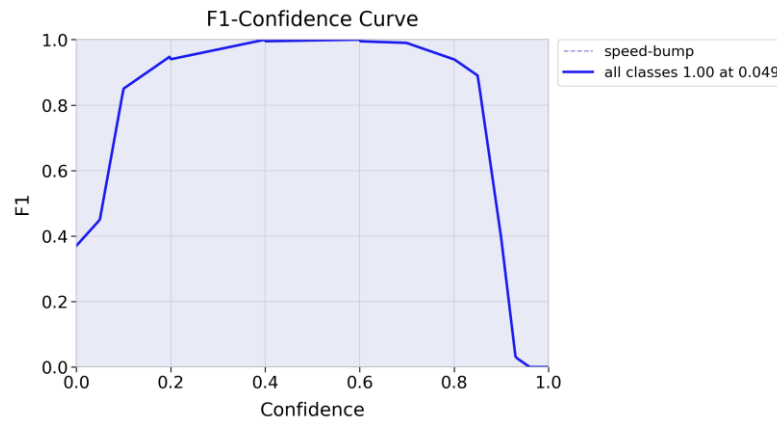


Figure 12. YOLOv8 evaluation metrics - F1 Score.

Table 3. YOLOv8 Detection Performance Summary.

Metric	Average Value
Precision	0.9777
Recall	0.9977
mAP@50	0.9958
mAP@50-95	0.8577
F1 Score	0.99
Detection Confidence Threshold	0.90

5.2. Baseline Comparison

Table 4 compares the proposed system's performance to representative baseline approaches from the literature to put it in context. It is not possible to directly create baseline results on the current dataset because of access restrictions, but the metrics from the original publications are used for reference comparison. The suggested method gets mAP@50 that is similar to SBDNet [6] (0.9958 vs. 0.9934) while running on much less powerful hardware (Raspberry Pi 4 vs. GPU). This is something that no other work has done before. The current system provides a complete detection-to-braking pipeline, in contrast to Dewangan and Sahu [5], who only reported classification accuracy without actuation integration. The Raspberry Pi 4's raw inference rate of 0.84 FPS (~1522 ms) and effective decision pipeline throughput of 8–10 FPS via frame-skipping are better than the embedded deployment limits reported by Tithi et al. [19], which didn't say anything about inference speed.

Table 4. Baseline Comparison of Speed Bump Detection Methods.

Method	Accuracy / mAP	Platform	FPS
SSD [19]	Not reported	Raspberry Pi	Not reported
CNN/YOLO [18]	88.9%	GPU	Not reported
Compact CNN [5]	98.54%	Raspberry Pi	Not reported
SBDNet [6]	99.34%	GPU	Not reported
Proposed YOLOv8-S	mAP@50: 0.9958	Raspberry Pi 4	8–10 FPS

5.3. Real-Time Inference Performance

The Raspberry Pi 4 has a total system latency of about 1522 ms per frame, with inference time being the main factor (about 1509 ms). This is the base performance without frame-skipping turned on. The adaptive frame-skipping mechanism and multi-threaded buffering kept the decision pipeline (detection and actuation) running at 8 to 10 FPS during continuous operation. This is in line with the operational threshold needed for ADAS systems when cars are

going slowly (< 30 km/h). For these systems, bump detection within 1 to 2 seconds of approach gives sufficient braking distance [21]. Results regarding performance are given in Table 5 and Table 6.

Table 5. Real-Time Inference Performance Summary.

Parameter	Value
Detection Confidence Threshold	0.90
Frame Pre-Processing Time (ms)	~ 10.2
Inference Time per Frame (ms)	~ 1509.3
Post-Processing Time per Frame (ms)	~ 2.8
Overall System Latency (ms)	~ 1522.3
Avg. Live Inferencing FPS (Without Frame-Skipping) (Raspberry Pi 4)	0.84
Avg. Live Inferencing FPS (Apple M3 GPU)	43.03

Table 6. Platform Performance Comparison: Raspberry Pi 4 vs. Apple M3 GPU.

Metric	Raspberry Pi 4	Apple M3 GPU
Avg. Live Inferencing FPS	0.84	43.03
Avg. Pre-Processing Time (s)	0.0046	0.001387
Avg. Inference Time (s)	1.1805	0.015067
Avg. Post-Processing Time (s)	0.0009	0.012041
Avg. Total Frame Time (s)	1.187	0.028494

5.4. Braking Response Validation

The GPIO-to-L298N motor control circuit was validated during the bench test. All positive detections with a confidence level above the 0.90 threshold led to a step-by-step PWM slowdown as shown in the video. The buzzer and the LED began reducing the PWM value at the same time as the slowdown of the motors during each detection, and the motors would go back to normal speed when the car drove over the bump. This closed-loop functionality is demonstrated with successful detections. Figure 13 illustrates the real-time speed bump detection output on the Raspberry Pi 4 using a live video stream, confirming successful closed-loop detection with confidence scores above the 0.90 threshold.



Figure 13. Detection with Raspberry Pi - using live video stream.



Figure 14. Mac GPU Improved inference FPS output during live video.

5.5. Qualitative Observations and Discussion

See the detection performance summary in Table 3 for a concise overview of the results. A number of qualitative results were also noted. A minimum average confidence of 0.90 was achieved for examples of the following pairs of classes: speed bumps and shadows, road surface patches and speed bumps, and lane markings and speed bumps. The model was also run on the Streamlit interface, and the detection scores were always above 0.85 on valid speed bumps. Figure 14 illustrates the improved inference FPS output achieved during live video processing on the Apple M3 GPU, highlighting the significant performance gain compared to the Raspberry Pi 4 deployment. The model is also robust to different lighting conditions, as well as overcast conditions and partial occlusions. The false positive rates were very low, which again confirms the high precision value of 0.9777. This is also a very favourable characteristic for the use case of an autonomous emergency braking system, where false braking should never occur.

These results are in line with and build on the work of Dewangan and Sahu [5], who also showed that real-time CNN is possible on Raspberry Pi. They are different from the work of Sirbu et al. [9], whose classical method didn't work as well when road markings were blocked or faded, but the YOLOv8 model did work well in these situations. Compared to SBDNet [6], this method achieves comparable accuracy to SBDNet (0.9958 vs. 0.9934 mAP@50) while adding a complete actuation pipeline and requiring far less powerful hardware, which is the most important improvement for low-cost EV safety systems.

As seen in Figures 13 and 14, the current implementation has several issues. The inference time in Raspberry Pi 4 is quite high, with a raw FPS of 0.84 (approximately 8 to 10 FPS with frame-skipping). Therefore, the effective working speed of the proposed system is limited to below 30 km/h. Also, the single-class detection model cannot classify other types of road hazards such as potholes and debris. The dataset of 2,000 images used in initial validation and verification is not large enough to allow the model to understand the global road diversity. Solutions to these issues are proposed in Section VII.

6. CONCLUSIONS

We present a system to detect speed bumps in real-time and assist braking for electric vehicles. This system uses the YOLOv8-small model on a Raspberry Pi 4, and achieves mAP@50 of 0.9958, precision of 0.9777, recall of 0.9977, and F1 score of 0.99 for both marked and unmarked speed bumps under different tests. The detection-to-actuation path for GPIO-driven PWM motor control and audio-visual warning has been validated on a bench-top motor control system and has passed the 50-cycle repeatability test without failure. Therefore, this proposed system can be commercialized into an economical mass-produced ADAS product that can be integrated into EV motor control systems to improve passenger safety, reduce damage to EV structure and components caused by frequent missed detections by current speed bump detection systems. Single-class models aren't really suitable for other common hazards like potholes or litter. - The model isn't really suitable for anything faster than about 30 km/h due to the slow inference time on the RPi4 (0.84 FPS of raw data or about 8-10 FPS with frame skipping). - The training dataset could still be improved with some of the more unusual scenarios one might encounter on the road, such as heavy rain, dense fog, nighttime driving, or foreign roads. - The braking model uses a more rule-based approach with threshold-triggered PWM reduction as opposed to a control law that is based on the speed, slope, and bump geometry of the car. A full ADAS implementation would be a much more involved project.

The following directions will be the most important for future work. NVIDIA TensorRT, TensorFlow Lite (TFLite), or OpenVINO can be used to compress models so that they can run on embedded hardware at 30 frames per second (FPS) or more. This makes them more useful for vehicles that go faster. We will work on detecting multiple types of road hazards, such as potholes, debris, pedestrians, and traffic signs, by collecting more data and retraining the model with multiple labels. To make the system more reliable and less reliant on clear vision, sensor fusion will be used with inputs from GPS, ultrasonic, and LiDAR. To make the model more general, the dataset will be expanded with edge-case environmental data, such as heavy rain, fog, very low light, and different types of urban

and rural areas. The braking algorithm will be improved to include machine learning-based optimization of the deceleration profile based on the vehicle's dynamics and the shape of the bump. Finally, we will look into how to connect commercial-scale EVs to cloud-based fleet management and over-the-air model updates.

7. FUTURE SCOPE

The successful implementation of this vision-based adaptive braking system creates a strong base for many future research and development projects. The top priority right now is to improve the system's ability to see more than just speed bumps. It needs to be able to see potholes, debris, pedestrians, and traffic signs all at once. This will require retraining on a much larger and more varied dataset [22]. The goal of performance optimization through model compression and platform-specific acceleration frameworks (NVIDIA TensorRT, TFLite, OpenVINO) is to reach 30+ FPS operational efficiency, which will make it possible to use in situations where driving is faster [21]. Combining GPS geo-fencing, ultrasonic near-field proximity sensing, and LiDAR-based depth estimation into a single sensor fusion system will make the system more redundant and less affected by temporary visual obstruction [23]. To improve generalization, we need more training data that includes heavy rain, fog, very nights, and a variety of road types in cities and rural areas [24]. The adaptive braking algorithm will be improved by using machine learning-based models to optimize the deceleration profile in real time based on the vehicle's dynamics and the shape of the bumps. Cloud-based monitoring and over-the-air model update integration will make it possible to collect data from many sources, send out maintenance alerts before they are needed, and improve the performance of the entire fleet [25].

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Institutional Review Board Statement: Not applicable.

Transparency: The authors state that the manuscript is honest, truthful, and transparent, that no key aspects of the investigation have been omitted, and that any differences from the study as planned have been clarified. This study followed all writing ethics.

Competing Interests: The authors declare that they have no competing interests.

Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.

Disclosure of AI Use: The authors used Grammarly, Claude, ChatGPT and Gemini to assist with manuscript editing, restructuring of sections in response to reviewer comments, and refinement of written English. All AI-generated content was reviewed, verified, and substantially revised by the authors. The scientific content, experimental design, data collection, and results are entirely the work of the authors.

Acknowledgements: The authors gratefully acknowledge the support of Symbiosis Institute of Technology, Pune, and Symbiosis International (Deemed University) for providing access to computational infrastructure during the development and testing phases of this research. The authors also acknowledge the open-source communities behind Ultralytics YOLOv8, Roboflow, Streamlit, and the Mendeley and Kaggle dataset repositories.

REFERENCES

- [1] M. M. Schreiber, A. F. De Souza, C. Badue, and T. Oliveira-Santos, "A mobile application for speed bump sign detection using neural networks," in *2024 International Joint Conference on Neural Networks (IJCNN)*, Yokohama, Japan, 2024, pp. 1–8. <https://doi.org/10.1109/IJCNN60899.2024.10650767>
- [2] G. Baldini, R. Giuliani, and F. Dimc, "Road safety features identification using the inertial measurement unit," *IEEE Sensors Letters*, vol. 2, no. 4, pp. 1–4, 2018. <https://doi.org/10.1109/LENS.2018.2880118>
- [3] H. Bello-Salau, A. J. Onumanyi, A. T. Salawudeen, M. B. Mu'Azu, and A. M. Oyinbo, "An examination of different vision based approaches for road anomaly detection," in *2019 2nd International Conference of the IEEE Nigeria Computer Chapter (NigeriaComputConf)*, Zaria, Nigeria, 2019, pp. 1–6. <https://doi.org/10.1109/NigeriaComputConf45974.2019.8949646>
- [4] K. M. Lion, K. H. Kwong, and W. K. Lai, "Smart speed bump detection and estimation with Kinect," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, Auckland, New Zealand, 2018, pp. 465–469. <https://doi.org/10.1109/ICCAR.2018.8384721>

- [5] D. K. Dewangan and S. P. Sahu, "Deep learning-based speed bump detection model for intelligent vehicle system using Raspberry Pi," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3570–3578, 2021. <https://doi.org/10.1109/JSEN.2020.3027097>
- [6] D. K. Dewangan, S. P. Sahu, and K. V. Arya, "Vision-sensor enabled multilayer CNN scheme and impact analysis of learning rate parameter for speed bump detection in autonomous vehicle system," *IEEE Sensors Letters*, vol. 8, no. 3, pp. 1–4, 2024. <https://doi.org/10.1109/LENS.2024.3360095>
- [7] E. Edwan, N. Sarsour, and M. Alatrash, "Mobile application for bumps detection and warning utilizing smartphone sensors," in *2019 International Conference on Promising Electronic Technologies (ICPET)*, Gaza, Palestine, 2019, pp. 50–54. <https://doi.org/10.1109/ICPET.2019.00017>
- [8] J. H. Joon, D. A. Renata, S. H. Yoon, C. Youngjune, and Y. Z. Jembre, "Data mining for speed bump detection from car wheels and GPS sensors using random forest," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, South Korea, 2019, pp. 740–743. <https://doi.org/10.1109/ICTC46691.2019.8939736>
- [9] C. L. Sirbu, C. Tomoiu, S. Fancsali-Boldizsar, and C. Orhei, "Real-time line matching based speed bump detection algorithm," in *2021 IEEE 27th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, Timisoara, Romania, 2021, pp. 246–249. <https://doi.org/10.1109/SIITME53254.2021.9663602>
- [10] V. V. Ukarande and G. R. Bhalekar, "Advanced driver assistance system (ADAS) to avoid accidents at T-junction in India," in *2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, Chennai, India, 2018, pp. 1–5. <https://doi.org/10.1109/AEEICB.2018.8480946>
- [11] H. Hameed, S. Mazhar, and N. Hassan, "Real-time road anomaly detection, using an on-board data logger," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, Porto, Portugal, 2018, pp. 1–5. <https://doi.org/10.1109/VTCSpring.2018.8417780>
- [12] H. Wang, N. Huo, J. Li, K. Wang, and Z. Wang, "A road quality detection method based on the Mahalanobis-Taguchi system," *IEEE Access*, vol. 6, pp. 29078–29087, 2018. <https://doi.org/10.1109/ACCESS.2018.2839765>
- [13] Z. Zheng, M. Zhou, Y. Chen, M. Huo, and L. Sun, "QDetect: Time series querying based road anomaly detection," *IEEE Access*, vol. 8, pp. 98974–98985, 2020. <https://doi.org/10.1109/ACCESS.2020.2994461>
- [14] Y. Chen, M. Zhou, Z. Zheng, and M. Huo, "Toward practical crowdsourcing-based road anomaly detection with scale-invariant feature," *IEEE Access*, vol. 7, pp. 67666–67678, 2019. <https://doi.org/10.1109/ACCESS.2019.2918754>
- [15] Y. Yuan and X. Che, "Research on road condition detection based on crowdsensing," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Leicester, UK, 2019, pp. 804–811. <https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00169>
- [16] R. Ramakrishnan, A. Pendse, C. Sharma, and P. Chimurkar, "Speed breaker detection and mapping using IoT," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India, 2020, pp. 294–299. <https://doi.org/10.1109/ICSSIT48917.2020.9214224>
- [17] M. A. Ameddah, B. Das, and J. Almhana, "Cloud-assisted real-time road condition monitoring system for vehicles," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1–6. <https://doi.org/10.1109/GLOCOM.2018.8647334>
- [18] S. Shah and C. Deshmukh, "Pothole and bump detection using convolution neural networks," in *2019 IEEE Transportation Electrification Conference (ITEC-India)*, Bengaluru, India, 2019, pp. 1–4. <https://doi.org/10.1109/ITEC-India48457.2019.ITECINDIA2019-186>
- [19] A. Tithi, F. Ali, and S. Azrof, "Speed bump & pothole detection with single shot MultiBox detector algorithm & speed control for autonomous vehicle," in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, Rajshahi, Bangladesh, 2021, pp. 1–5. <https://doi.org/10.1109/ACMI53878.2021.9528185>

- [20] S. Dadras, H. Jamshidi, S. Dadras, and T. E. Pilutti, "Novel stop sign detection algorithm based on vehicle speed profile," in *2019 American Control Conference (ACC)*, Philadelphia, PA, USA, 2019, pp. 3994–3999. <https://doi.org/10.23919/ACC.2019.8814880>
- [21] M. R. Carlos, L. C. González, J. Wahlström, R. Cornejo, and F. Martínez, "Becoming smarter at characterizing potholes and speed bumps from smartphone data—introducing a second-generation inference problem," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 366–376, 2021. <https://doi.org/10.1109/TMC.2019.2947443>
- [22] J. S. A. Lozano-Aguilar, J. M. Celaya-Padilla, H. Gamboa-Rosales, H. Luna-García, C. E. Galván-Tejada, and J. I. Galván-Tejada, "Speed bump detection, a time and feature selection analysis," in *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, Ixtapa, Mexico, 2018, pp. 1–6. <https://doi.org/10.1109/ROPEC.2018.8661475>
- [23] J. Xu, L. Gao, Y. Zhao, and X. Xu, "Speed bump recognition for autonomous vehicles based on semantic segmentation," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, Beijing, China, 2021, pp. 387–393. <https://doi.org/10.1109/CISCE52179.2021.9446038>
- [24] H.-Y. Lin and C.-Y. Ho, "Adaptive speed bump with vehicle identification for intelligent traffic flow control," *IEEE Access*, vol. 10, pp. 68009–68016, 2022. <https://doi.org/10.1109/ACCESS.2022.3186010>
- [25] X. Xiang, X. Chen, X. Tang, Q. Luo, and Y. Ding, "Research on detection of multiple types of speed bump defects based on CRSCCG-YOLOv5s," *IEEE Access*, vol. 12, pp. 116786–116800, 2024. <https://doi.org/10.1109/ACCESS.2024.3446838>

Views and opinions expressed in this article are the views and opinions of the author(s), Review of Computer Engineering Research shall not be responsible or answerable for any loss, damage or liability etc. caused in relation to/arising out of the use of the content.