Real-time Pothole Detection

Mustafa Bin Amir
Lahore University of Management Sciences
Lahore, Pakistan
24100084@lums.edu.pk

Azmeer Faisal
Lahore University of Management Sciences
Lahore, Pakistan
24100164@lums.edu.pk

Muhammad Nameer Anjum
Lahore University of Management Sciences
Lahore, Pakistan
24100270@lums.edu.pk

Abstract

Potholes on roads are a significant safety hazard contributing to numerous road accidents, leading to fatalities and severe injuries. The current manual approach to identifying potholes is time-consuming and often inadequate, leading to delayed repairs and an increased risk of accidents. In this study, we propose a mobile application that employs machine learning algorithms to detect potholes on roads in real-time, providing a cost-effective and efficient solution to this safety hazard. The application uses the smartphone's camera to detect potholes allowing drivers and pedestrians to identify and avoid these hazardous spots. We present the design and implementation of the proposed system, including the machine learning algorithms used for pothole detection. We also evaluate the system's performance using accuracy metrics and conduct experiments to test its feasibility. This study aims to improve road safety by providing a reliable solution to accurately detect potholes in real-time, providing timely information to users to help them avoid accidents.

CCS Concepts: • **Networks → Application layer protocols; Mobile networks**; • **Computing methodologies → Object detection; Neural networks**.

Keywords: Vehicular Systems; Automotive Systems; Edge Computing; Crowd-sourcing; Neural Networks; Object Detection

ACM Reference Format:
Mustafa Bin Amir, Azmeer Faisal, and Muhammad Nameer Anjum. 2023. Real-time Pothole Detection. In *Proceedings of (Final Report Submission).* LUMS, Lahore, Pakistan, 5 pages.

1 Introduction

Potholes present a significant problem on roads globally, creating safety hazards for drivers, motorcyclists, and pedestrians. These hazards pose a serious threat to public safety, leading to numerous road accidents, fatalities, and vehicle damages. [6]Poor road conditions are responsible for one-third of approximately 33,000 traffic fatalities annually. The problem is even more severe in developing countries, with a higher percentage of fatalities and injuries caused by road accidents due to inadequate infrastructure and maintenance.

Potholes significantly contribute to the high number of road accidents worldwide, necessitating immediate action. A range of factors, such as weather conditions, heavy traffic, and poor maintenance, form potholes. These factors lead to the repeated contraction and expansion of pavement due to temperature changes and the weight of vehicles, eventually leading to cracks and fissures that develop into potholes. [1]In million-plus cities during 2021, potholes accounted for nearly 0.8

Traditionally, the identification and repair of potholes have been labor-intensive and time-consuming, often resulting in delays in repairs and increased road safety hazards. Although several pothole detection systems that use sensors and other technologies to identify potholes are available in the market, they are expensive and not widely accessible, particularly in developing countries where the problem is most severe.

1.1 Related Works

Several methodologies based on different technologies have been proposed, each of which has limitations and inefficiencies.

An accelerometer-based pothole detection system has been proposed by P. Harikrishnan[7] and K. Zoysa[5]. The system accurately detects potholes and other road anomalies, but it cannot inform the user before encountering the pothole.

Yu and Yu [11] use an accelerometer coupled with an oscilloscope, which increases the overall cost of installing the system due to additional hardware requirements.

S.Silvester[9] uses a smartphone application that runs the SSD MobileNet object detection model, coupled with accelerometer and gyroscope sensors built into smartphones. The camera detects potholes ahead, and the in-built accelerometer and gyroscope help detect missed potholes when the vehicle passes over them. However, the system's accuracy and effectiveness have not been tested in real-world scenarios.

A similar smartphone application using the YOLOv3 SPP and the SSD MobileNetV2 models has been proposed by N. Camilleri[3].

In this study, we propose a cost-effective and practical solution to the problem of potholes through the development of a mobile application that employs machine learning algorithms to detect potholes in real-time. We believe that a mobile application[4] will be the most feasible and cost-effective solution for pothole detection, particularly in developing countries where the problem is most prevalent. This approach has the potential to reach a wide audience, as most people in developing countries have access to smartphones. In the following sections, we discuss the methodology and implementation of our system, evaluate its performance through a series of experiments, measuring its accuracy and effectiveness in real-world scenarios. Ultimately, our goal is to contribute to improving road safety by providing a reliable and cost-effective solution to the problem of pothole.

2 RoadSense

RoadSense is an Android mobile application developed using TensorFlow's starter code. It identifies potholes in real time and notifies all users except the individual who identified the object. An object detection algorithm is utilised to detect potholes, manholes, and speed breakers on the road.

This article defines an *object* as potholes, manholes, or speed breakers. Figure 1 explains the working of the system. RoadSense detects objects in real-time using the device's native camera. The algorithm from a state-of-the-art object detection model EfficientDet-Lite is used on every individual frame to draw a bounding box around the detected object and the model's confidence level in detection. Object detection serves the purpose of a trigger that sends the device's current location, the user's id who detected the object and the metadata of the detected object to the cloud for persistent storage. When there is a new "entry" in the cloud, the application fetches the update, calculates the distance between the user and the object for every user, and displays an alert.

3 Methodology and Implementation

We can classify the project into two primary components. The first component involves implementing a machine-learning object detection model to identify potholes actively. The second component actively disseminates the pothole's location to other application users upon detecting it.

3.1 Object detection

3.1.1 Data Collection and Labeling.

Our dataset contains 1254 images, 87% of which have been extracted from Kaggle and the remaining 13% was collected manually in various contexts with varying lighting conditions. The data was collected manually in Keer Khud, a semi-urban location. The road conditions are identical to those found in Pakistan. LabelIMG software was used to label photos that were manually captured.

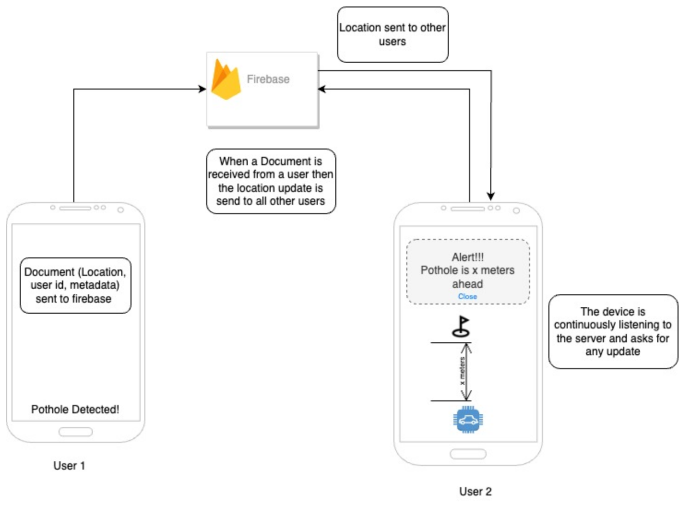
Location sent to other users
Firebase
When a Document is received from a user then the location update is send to all other users
Document (Location, user id, metadata) sent to firebase
Pothole Detected!
User 1
Alert!!! Pothole is x meters ahead
Close
x meters
The device is continuously listening to the server and asks for any update
User 2

Figure 1. This is a label for the image.

Multiple trials were conducted to identify the Train/Test split of the dataset. We began with the 80/20 split, but the 88/12 split allowed us to improve the Model's Mean Average Precision.

The dataset has certain limitations, which will be reflected in our trials, mainly because most of it came from Kaggle and contained out-of-context photos. The dataset has a class imbalance. Approximately 93% of the images are of potholes, with the remaining 7% being of manholes and speed breakers. This dataset is noisy since the photos were taken up close and at an extreme tilt.

3.1.2 Object Detection Model Selection.

Object detection is a computer vision task that aims to identify which of a known set of objects might be present and provide information about their positions within the given image or a video stream.

The state-of-the-art object detection models can be categorized into two main classes: Large models and mobile models.

Large models are complex models with a massive number of parameters. These are computationally expensive and require a relatively high inference time for object detection. These models include YOLO, SSD, Faster R-CNNs, Mask R-CNNs and Cascade R-CNNs.[10]

Mobile models are simpler models with relatively less number of parameters and are of smaller size. They are specifically designed for resource-limited devices such as mobile phones and Edge devices. These models include EfficientDet-lite, SSD MobileNet and YOLO tiny.[10]

The state-of-the-art mobile-optimized model does not only need to be more accurate, but it also needs to run faster and be smaller.

We preferred the EfficientDet-lite model, which has the same accuracy as the YOLO-tiny but is much smaller. For example, YOLOv3-tiny has 65 million parameters compared to 4 million parameters in EfficientDet-lite0, despite both having comparable accuracy.

EfficientDet-Lite has multiple different versions: Lite0 to Lite4. The smaller version runs faster but is less accurate than, the larger version.

Our model selection is mainly focused on model inference time, and because EfficientDet-lite0 to lite2 versions provide the shortest inference time, we will focus on them in this study.

3.1.3 Transfer Training.

We utilize transfer learning on a pre-trained model to train the model on our customized dataset. A pre-trained model is a previously trained network on a large dataset, generally on a large-scale image classification task. Transfer learning involves reusing a pre-trained model (trained on a large dataset) to accomplish a new task on a different but related dataset. [2]Transfer learning is beneficial if the new dataset is insufficient or small to train a model from scratch. In such cases, fine-tuning the pre-trained model on the new dataset can achieve better performance on the new task. The rationale behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, it will effectively act as a generic model of the visual world. By training a large model on a large dataset, we can take advantage of these learned feature maps without starting from scratch.

3.2 Data Propagation

Data propagation refers to the process in which a user detects an object, and its coordinates are sent to the cloud. Another user then fetches this data, and an alert is displayed corresponding to the distance between that user and the object. Figure 1 depicts two users: user1 and user2. When user1 identifies an item, the mobile phone's current location's longitude and latitude coordinates, user1's id, and object metadata are transmitted to Firebase cloud storage. The object's metadata includes the bounding box coordinates, category details, and detection timestamps. This data tuple is saved in the Firebase Firestore as a document. When the Firestore is updated, the app retrieves all the documents. The most recent document is used to determine the distance between the device's precise position and the most recent object fetched using the formula below.

$$R1 = (\sin(lat1) * \sin(lat2))$$
$$R2 = \cos(lat1) * \cos(lat2) * \cos(long2 - long1)$$
$$D = 3963.0 * \arccos(R1 + R2)$$

The application divides live camera input into frames. Each frame is interpreted for object detection. Data is sent to the cloud if an object is spotted in each frame. Suppose an object remains in the camera grid for a slightly longer period, as it does in numerous frames. In that case, additional data propagation messages will be delivered for that same object. This soon fills up the cloud storage space. Our proposed solution included a timer of 30 seconds, which was tested through trial and error. This method allows the app to identify objects regularly while only invoking the database every 30 seconds. This method is still infeasible, and other approaches, such as uniquely identifying an object so that it is only detected for the first time.

4 Experiment and Analysis

4.1 Testbed Setup

In order to test our machine learning models we ran our application on a BlueStacks 5 emulator assigning it 3GB of RAM. Instead of realtime smartphone camera input we fed the machine learning model a stable downloaded video input of 25fps . The video was shot with a dashcam on a pothole-stricken road with the camera moving at a speed of approximately 20km/h. The video was played on OBS studio on a virtual camera connected to the emulator.

4.2 Object Detection Model Analysis

To compare models, the mean average precision metric is commonly used. Because our models were from the same EfficientDet-Lite family but had varying inference times and complexity, we plotted Mean average accuracy for each model trained over COCO dataset 2017 and Our dataset, as shown in figure 3. The model's mean average precision or accuracy rises as its complexity rises, but does this imply that the most complex model from our pool of models Lite0 to Lite2 suits our purpose?

One of our primary goals is to limit latency or inference time while increasing the number of detected items. We don't know how mean average accuracy relates to inference time, so we plotted an inference time graph for each model, as shown in Figure 4. It can be seen that as model complexity increases, so does inference time or letancy. Moreover it can also be inffered that as the complexity increases the error rises from 20% to 30%.

Inference times are highly influenced by the background application running on the mobile phone. We observe an increase in inference times along with the error when the number of background applications were increased.

4.3 Accuracy Metric

We define another metric called Detection Accuracy for our purpose. It it defined as:

$$DetectionAccuracy = \frac{no.of objects-detected}{Total-number-of-Objects}$$

Figure 5 illustrates that at a 50% confidence level, the simplest and least complex model with the shortest inference times detects the most objects. As a result, EfficcientDet-Lite0 enables us minimising latency as much as feasible. We know that increasing the number of threads reduces inference

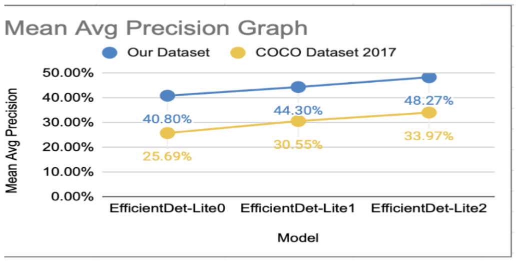

Figure 2

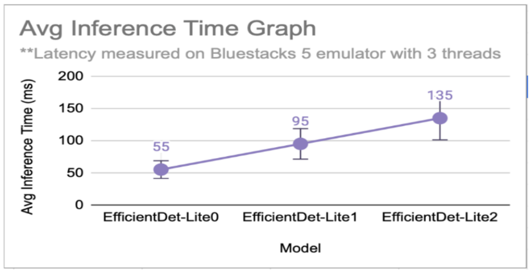

Figure 3

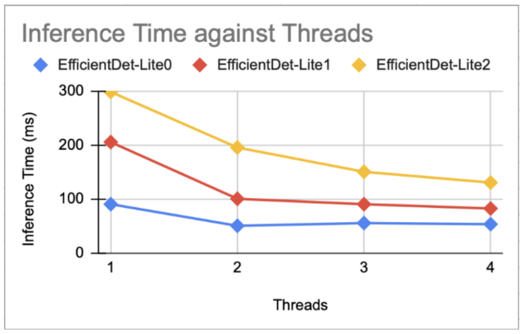

Figure 4

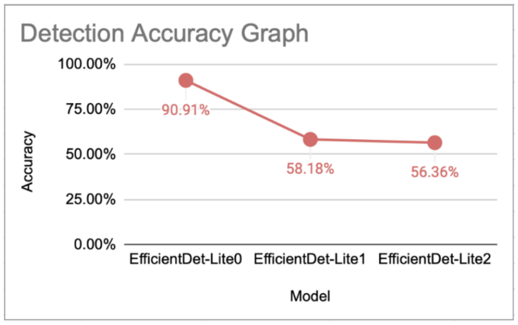

Figure 5

time, but how much? As a result, Figure 6 depicts the relationship between inference time and the number of threads for each model. It is evident that inference times can only be decreased to a certain point as we observe diminishing returns for higher number for threads.

4.4 Observations

Other observations which were made during the manual testing of the model are related to the dataset limitations.

4.4.1 Camera Angle. The images were a closeup of the object in the dataset hence the model could only detect objects when the camera was tilted at around a 25° angle of depression.

4.4.2 Camera Stability. The model recognises objects when the camera is stable.

4.4.3 Vehicle Speed. As vehicle speed increases the number of objects detected gets reduced. This limitation is because of the inherent inference time of the model.

4.4.4 Screen reflection. number of objects detected gets reduced when we tried the model to identify items through a glass.

5 Results

The model best suited to our purpose is EfficientDet-Lite0. It can recognize an object up to 4 meters at the speed of 20km/h. Figure 4 shows that increasing the number of threads decreased the inference time for each model; almost by a factor of 3 for EfficientDet-Lite2. The model with the least inference time was the smallest size model EfficientDet-Lite0. As inference time plays a major role in identifying a pothole under an appropriate amount of time Figure 5 shows that EffcientDetLite-0 also provided the highest accuracy of approximately 90% even though it had the lowest accuracy on the COCO datset.

6 Limitations

Several limitations must be considered during the real-world testing of the mobile phone application. Firstly, limitations were observed due to the dataset used for training and validation. Specifically, lighting conditions varied significantly in the dataset, affecting the model's accuracy in detecting the potholes in real-world conditions. Moreover, the road conditions in the dataset only sometimes matched those encountered in real-world scenarios, which may limit the generalizability of the results. Another limitation was that the camera angle in the dataset often captured close-up images, which may not represent the actual scenarios where the camera pothole is at a distance greater than 4 meters.
Another limitation observed in our study was that the dataset used for training needed to be more evenly represented across the different classes. While the dataset contained three classes, namely potholes, manholes, and speed breakers, more than 90% of the data belonged to the potholes class. This imbalance in class representation may have affected the model's ability to accurately predict certain features, particularly in detecting manholes and speed breakers. Therefore, it is crucial to consider this limitation when interpreting the

results of this study and when using the model in real-world applications. The model's accuracy in detecting manholes and speed breakers may not be as reliable as that for potholes.

Additionally, there were limitations related to human errors. These limitations include variations in camera angle caused by tilting and the absence of any stabilization device, such as a gimbal, resulting in poor camera stabilization and blurry images. Moreover, the phone used to capture images was not firmly supported, further impacting image stability. Due to the high speed of the moving vehicle, motion blur was evident in the captured images. This also led to a reduction in the camera's exposure time, resulting in darker pictures.[8] Consequently, some frames were missed, and potholes appeared darker in the captured video footage. Due to this, detecting potholes, manholes, and speed breakers was challenging for the application, which caused a lower accuracy in the results. These factors collectively made it challenging to detect potholes accurately, leading to low accuracy of the pothole detection system.

Another limitation of this experiment is that a medium-end mobile device with a subpar camera and processing speed was used, which may have decreased the accuracy of the object detection application. Additionally, only one device was used for testing, limiting the ability to differentiate between the impacts of mobile device specifications, such as memory, camera quality, and camera optimizers, on the detection application's quality. Testing the application on multiple devices, including low-end and high-end specifications, would provide a better understanding of the application's performance across various mobile devices.

7 Future Work

The future work for our application involves addressing certain limitations of the current implementation. Firstly, the inadequacy and imbalance of the dataset, which mainly comprised close-up images of potholes, need to be overcome by gathering a more extensive dataset consisting of images captured from the dashboard view inside the car, as the smartphone would also be mounted there. Secondly, we plan to explore offloading the machine learning model to an edge server. We can run more complex models, such as non-lite versions of EffecientNets, YOLOS, and SSDs which provide higher accuracies. However, we must address the challenge of increased inference time associated with this approach and devise effective workarounds to overcome it.

8 References

References

[1] [n. d.]. ([n. d.]).

[2] [n. d.]. https://www.tensorflow.org/tutorials/images/transfer_learning

[3] Neil Camilleri and Thomas Gatt. 2020. Detecting road potholes using computer vision techniques. In *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE, 343–350.

[4] Kevin Christensen, Christoph Mertz, Padmanabhan Pillai, Martial Hebert, and Mahadev Satyanarayanan. 2019. Towards a distraction-free waze. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. 15–20.

[5] Kasun De Zoysa, Chamath Keppitiyagama, Gihan P Seneviratne, and WWAT Shihan. 2007. A public transport system based sensor network for road surface condition monitoring. In *Proceedings of the 2007 workshop on Networked systems for developing regions*. 1–6.

[6] Pothole Facts. 2020. The pothole facts. https://www.pothole.info/the-facts/

[7] PM Harikrishnan and Varun P Gopi. 2017. Vehicle vibration signal processing for road surface monitoring. *IEEE Sensors Journal* 17, 16 (2017), 5192–5197.

[8] Huang Hsiang, Kuan-Chung Chen, Po-Yi Li, and Yung-Yuan Chen. 2020. Analysis of the effect of automotive ethernet camera image quality on object detection models. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 021–026.

[9] Shebin Silvister, Dheeraj Komandur, Shubham Kokate, Aditya Khochare, Uday More, Vinayak Musale, and Avadhoot Joshi. 2019. Deep learning approach to detect potholes in real-time using smartphone. In *2019 IEEE Pune Section International Conference (PuneCon)*. IEEE, 1–4.

[10] Tensorflow. [n. d.]. Higher accuracy on vision models with EfficientNet-Lite. https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html

[11] Bill X Yu and Xinbao Yu. 2006. Vibration-based system for pavement condition evaluation. In *Applications of advanced technology in transportation*. 183–189.

Received 10 May 2023