Hindawi
Advances in Civil Engineering
Volume 2022, Article ID 9221211, 13 pages
https://doi.org/10.1155/2022/9221211

Research Article

Pothole Detection Using Deep Learning: A Real-Time and AI-on-the-Edge Perspective

Muhammad Haroon Asad,[1] **Saran Khaliq**,[1] **Muhammad Haroon Yousaf**,[1,2] **Muhammad Obaid Ullah**,[3] **and Afaq Ahmad**[4]

[1]*Swarm Robotics Lab, National Centre of Robotics and Automation, Rawalpindi, Pakistan*
[2]*Department of Computer Engineering, University of Engineering and Technology Taxila, Taxila, Pakistan*
[3]*Department of Electrical Engineering, University of Engineering and Technology Taxila, Taxila, Pakistan*
[4]*Department of Civil Engineering, University of Engineering and Technology Taxila, Taxila, Pakistan*

Correspondence should be addressed to Afaq Ahmad; afaq.ahmad@uettaxila.edu.pk

Received 10 January 2022; Revised 19 February 2022; Accepted 3 March 2022; Published 20 April 2022

Academic Editor: Qian Chen

Asphalt pavement distresses are the major concern of underdeveloped and developed nations for the smooth running of daily life commute. Among various pavement failures, numerous research can be found on pothole detection as they are injurious to automobiles and passengers that may turn into an accident. This work is intended to explore the potential of deep learning models and deploy three superlative deep learning models on edge devices for pothole detection. In this work, we have exploited the AI kit (OAK-D) on a single-board computer (Raspberry Pi) as an edge platform for pothole detection. Detailed real-time performance comparison of state-of-the-art deep learning models and object detection frameworks (YOLOv1, YOLOv2, YOLOv3, YOLOv4, Tiny-YOLOv4, YOLOv5, and SSD-mobilenetv2) for pothole detection is presented. The experimentation is performed on an image dataset with pothole in diverse road conditions and illumination variations as well as on real-time video captured through a moving vehicle. The Tiny-YOLOv4, YOLOv4, and YOLOv5 evince the highest mean average precision (mAP) of 80.04%, 85.48%, and 95%, respectively, on the image set, thus proving the strength of the proposed approach for pothole detection and deployed on OAK-D for real-time detection. The study corroborated Tiny-YOLOv4 as the befitted model for real-time pothole detection with 90% detection accuracy and 31.76 FPS.

1. Introduction

Roads are the essential means of transportation for a country to provide commutation facilities nationwide. Road infrastructure enables opportunities to connect people and transport goods to enhance business opportunities, access to jobs, economic growth, and health care system across the country. As first-rated roads contribute to the country's GDP [1], the calamitous infrastructure of roads can become fatal for passengers' safety and vehicles' condition. The roads are usually made up of asphalt pavement and are prone to different structural damages with the passage of time.

The asphalt pavement distresses have been a concern of authorities to avoid unwanted circumstances. These pavements are vulnerable to scenarios such as traffic load, weather conditions, age, poor material used for construction, and miserable drainage system, exhibiting two major pavement failures such as cracks and potholes. Potholes are essentially the concave-shaped depressions in the road surface that requires attention as they induce awful circumstances such as accidents, unpleasant driving experiences, and malfunctioning of vehicles, as shown in Figure 1. Potholes should be dealt with on a priority basis to minimize their contribution towards unfortunate scenarios. According to the prediction made by WHO (World Health Organization), road accidents will become the fifth leading cause of death in 2030 [3]. The significance of potholes created conspicuous interest for the researchers of the civil

Figure 1: Road image having potholes [2].

community. The developing nations use manual inspection methods to recognize the potholes leading to inaccurate estimation as it is highly dependent on individual experience. These manual inspection methods require human interventions that are time consuming and costly. Many technical solutions exist for pothole detection such as scanning based with 3D reconstruction [4–6], vibration sensor based [7–10], thermal imaging [11, 12], and computer vision based [13–15].

A sensor-based network "BusNet" [9] is proposed for the monitoring of road conditions whereas the camera was mounted on the public transport buses. Several sensors and GPS are used which are fast, sensitive, and cost efficient. However, this system is not ideal for every case due to weather conditions that may damage the sensor leading to the performance degradation of the BusNet. Over the last few years, the computer vision and image processing based techniques acquired fame due to the accessibility of cameras which are inexpensive and feasible, and have been proved to be the replacement of old fashioned manual inspection methods for pothole detection. However, the image processing-based pothole detection is still a demanding job because of the irregular pothole textures, pothole structures, road bumps, manholes and shadows, etc. For this problem, different computer-vision based approaches have been studied for pothole detection and classification. In [16], the authors proposed a cost-effective solution for pothole detection and severity estimation based on image processing techniques. As compared to the manual method, their study achieved 88.4% accuracy with the automated method. The proposed system is less time consuming and can be done without the committee; however, the lightweight camera can be used to overcome shadow effects. Zhou et al. [17] proposed a methodology to replace human intervention methods with the image processing techniques based on the discrete wavelet transform for pavement distress detection. Another work [18] proposed discolorations, where potholes are detected based on characteristics such as dark region, globe shape, and rough texture. However, the discolorations may not always be potholes as we have other reasons such as road markings, shadows, wet roads, and manholes. Wang et al. [19] proposed a method in which wavelet energy is assembled via morphological processing initially for pothole detection, and afterward, detected pothole images are segmented using the Markov random field model so that the pothole edge is extracted. This methodology tested and trained over 120 pavement images in MATLAB. The method has attained an accuracy of 86.7%, with 83.3% precision, and 87.5% recall. The overlap degree between the extracted pothole region and the original pothole region is approximately above 85%, which accounts for 88.6% of the total detected pavement pothole images.

Altogether, machine learning and deep learning techniques have reduced complexity and cost for pothole detection. Arbawa et al. [20] proposed a method for detecting road potholes using the gray-level co-occurrence matrix (GLCM) feature extractor and support vector machine (SVM) as a classifier. They analyzed three features such as contrast, correlation, and dissimilarity. The results have shown that a combination of contrast and dissimilarity features exhibits better results with an accuracy of 92.033% and computing time of 0.0704 seconds per frame. Oche et al. [21] used five binary classification models (Naïve Bayes, Logistic regression, SVM, K-Nearest Neighbors (KNN), and Random Forest Tree) and presented a comparison of various machine learning approaches on data collected through smartphones and car routes. The Random Forest Tree and KNN achieved the highest accuracy of 0.8889 on the test set. To improve the accuracy of the Random Forest Tree, they tuned hyperparameters and increased accuracy up to 0.9444. The model has shown promising results on different routes and out of sample data. Ping et al. [22] presented techniques based on the combination of machine learning and deep learning models to detect potholes. A pothole detection system uses YOLOv3, HOG, SSD, SVM, and Faster R-CNN, trained on their dataset collected by mounting smartphone on vehicle dashboard. After machine learning and deep learning models trained on each of the mentioned techniques, YOLOv3 outperformed others in detecting potholes and estimating the size of the pothole with an accuracy of 82% on 1,500 images. However, their performance on out-of-sample data is unsatisfactory as they have not tested real-world scenarios. Ye et al. [23] proposed a method for the inspection of road defects and potholes based on two convolutional neural networks, conventional CNN, and prepooling CNN. The model is trained on 96,000 digital pavement images and achieved higher recognition precision of 98.95% with optimized prepooling CNN. They concluded optimized prepooling CNN with higher stability and robustness for real-world scenarios against traditional image processing techniques. K-means and Sobel edge detection algorithms did not detect and localize potholes accurately as CNN did. In [24], three different datasets are used to classify three different road types such as paved, unpaved, and asphalt for the further classification of the pothole in each road type. Finally, the CNN model uses 7,000 images of datasets that are RTK, KITTI, and caRINA for training. Moreover, they proposed an application that notifies pedestrians and drivers of upcoming potholes on the route. YOLOv3 and CNN models were used to detect the potholes and classify road types. Zhang et al. [25] proposed an embedded system integrated with CNN for pavement distress detection using the Montreal Pavement dataset. The model exhibits a true-positive rate for a pothole, patch, marking, crack linear, and crack network as 75.7%, 84.1%, 76.3%, 79.4%, and 83.1%, respectively.

It is crucial to assess the condition of road surfaces for public safety and usability. Hassan et al. [26] provided a summary and detailed insight of factors that affect the generalizability of any model for automated pavement assessment by investigating some common issues such as the distance of pothole from camera angle and lighting, and variations of images in terms of image size. They used the Kaggle pothole dataset and tested Faster RCNN with inception V2 as a backbone. The model achieved an accuracy of 90% as they performed the first experiment on the negative image. Later, the second experiment is performed on the Cranfield dataset, which results in 80% precision and 92% recall. In the third experiment, images are captured by stereocamera, the and model results in 95% precision and 84% recall. The fourth experiment is performed on images collected across Dublin city center in daylight and the model gives precision 78% and 68% recall in normal light, and 78% precision and 73% recall in low light. Kavitha et al. [27] proposed a method for self-driving vehicles and used the YOLO algorithm to detect objects and the training. PASCAL VOC dataset with XML format is collected and then implemented the method on Raspberry pi. Chen et al. [28] proposed a novel location-aware convolutional neural network and trained on a public pothole dataset that consists of 4,026 images as training samples and 1,650 images as test samples. The proposed model is based on 2D-vision techniques and location-aware convolutional networks. CNN networks consist of two main subnetworks; the first localization subnetwork (LCNN) finds as many candidate regions as possible by employing a high recall network model, and the second part-based subnetwork (PCNN) performs classification on the candidates on which the network is expected to focus. The proposed method achieved high precision 95.2% and recall 92.0%. Rani et al. [29] detected potholes and road bumps for an advance driver assistance system (ADAS) using SSD-MobileNet for detection which is trained on their self-made dataset collected from Malaysian roads. The model was able to detect potholes and road bumps with the limitations of accuracy and confidence however they suggested it for real-time detection as FPS (frame per second) was 22. Lately, deep convolutional neural networks (DCNNs) have become well known for problems like object classification [30], object detection, and recognition [31, 32] as they automatically extract the main features from images with basically no interventions. Furthermore, DCNNs have various applications in domains such as natural language processing (NLP) [33] and speech and audio processing (SAP) [34].

The literature shows that the sensor-based pothole detections are vulnerable to weather conditions. Furthermore, the pothole is not detected unless a vehicle or sensor is above the pothole. The sensor-based detection system is prone to failure if potholes contain pebbles or sand. A 3D reconstruction-based system detects potholes with size estimation. This system has a limitation of close-range detection with costly equipment. Sensor-based and 3D reconstruction-based systems are less feasible for real-time applications as they require complex hardware. Vision-based systems that incorporate intelligent and low-cost cameras have gained attention from researchers. These systems are proven robust and feasible but with limitations of false detection, illumination, and the texture of road potholes. Machine-learning-based systems are the superior version of the above-discussed techniques and require prior knowledge and time to develop a feature extractor for the dedicated problem. Therefore, deep learning techniques are more suitable for pothole detection with edge devices mounted on vehicles.

In this work, our contributions are as follows:

(i) We have presented a comprehensive experimental and comparative study of whole YOLO family and SSD-Mobilenetv2 on pothole detection in terms of accuracy and speed

(ii) We have introduced a thorough methodology to deploy custom-trained CNNs on edge devices for pothole detection.

(iii) We have proposed a real-time and AI-on-the-edge solution for pothole detection using an OAK-D camera on a single-board computer (Raspberry Pi) as host by deploying the top three models based on performance metrics

In further proceedings of this article, we shall discuss in Section 2 our proposed methodology for a real-time pothole detection system. Similarly, Section 3 discusses the experimental results thoroughly. We discussed the conclusion and a few comments on future work in the Section 4 of this article.

2. Proposed Methodology

For a real-time pothole detection system, the block diagram of the proposed methodology is shown in Figure 2. Annotation for each image is performed explicitly after the collection of the dataset. The annotated data are split into training and testing data before passing it to deep learning models such as the YOLO family and SSD for custom model training. The weights obtained after training contribute to model performance evaluation on testing data. The custom weights are then converted into the OpenVino IR format to perform real-time detection on OAK-D and Raspberry pi as host computer. The methodology is discussed in detail in the following sections.

2.1. Dataset Acquisition.

The performance and reliability of the models depend upon the dataset used for training. The dataset must contain realistic pothole images. Hence, the latest publicly available pothole image dataset is used [2] that consists of 665 images, with effects of shadows, moving vehicles, and illumination variations that incorporate real-world scenarios around potholes. The dataset images are collected from online sources making noisy and low-quality images. Each image of a dataset contains at least three potholes. So, 8,000 potholes are available in the whole dataset approximately. Some samples from the pothole image dataset are shown in Figure 3.

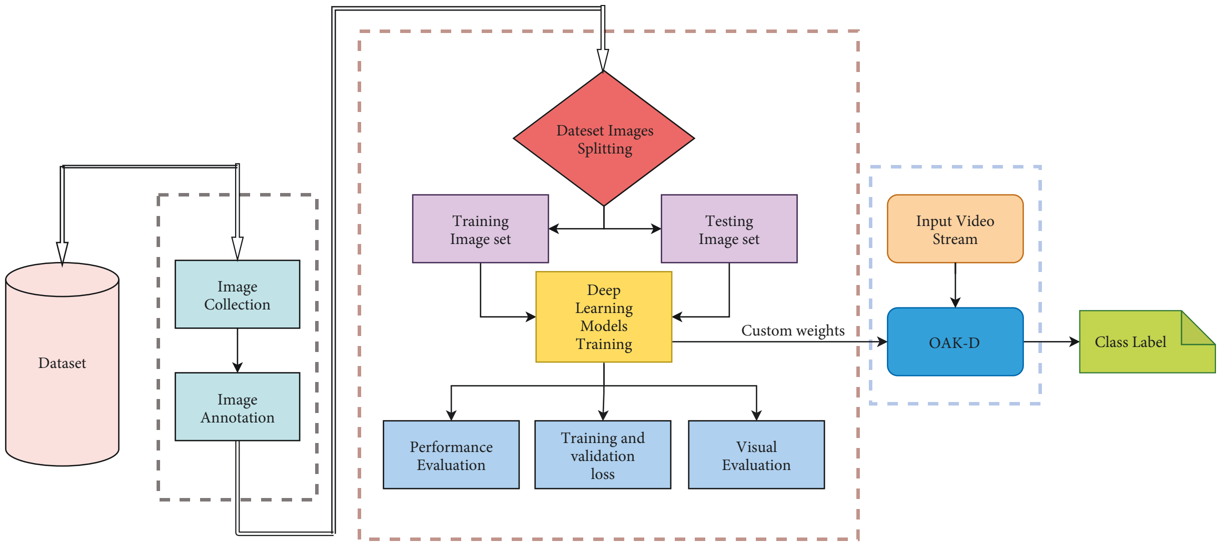


Figure 2: Proposed methodology block diagram of real-time pothole detection.

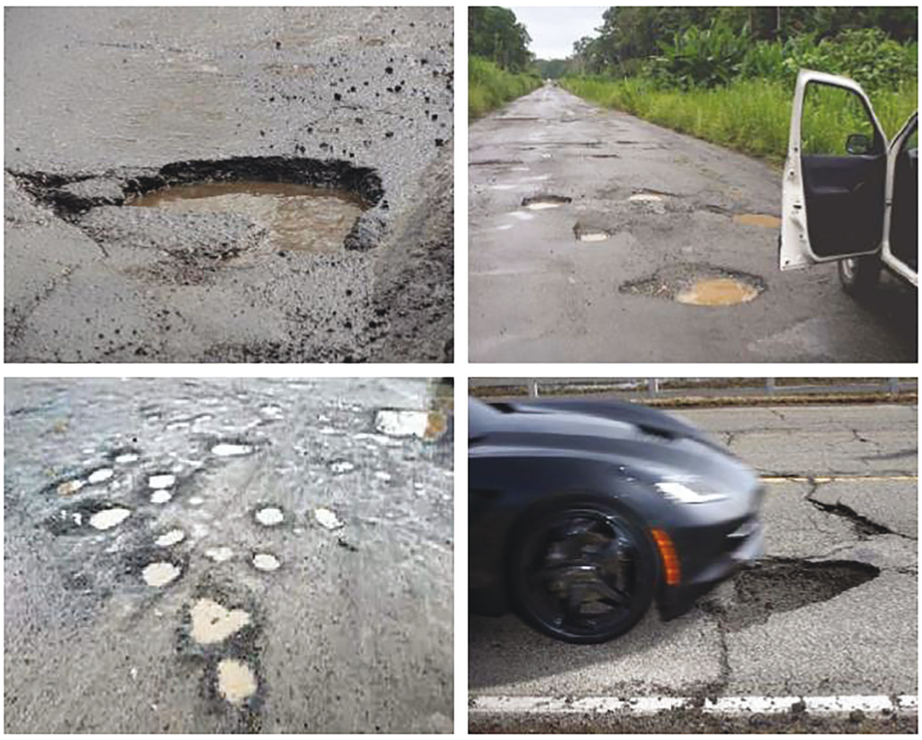
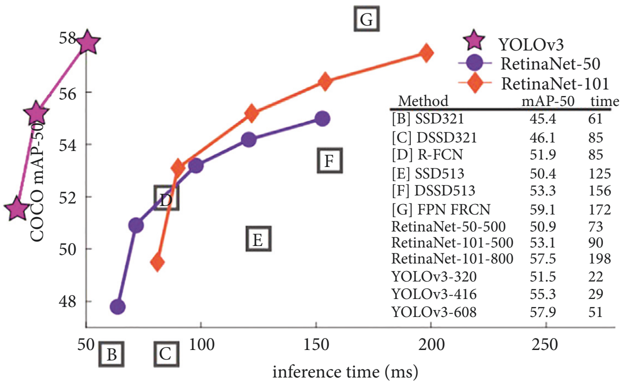


Figure 3: Sample of pothole images from the dataset.

2.2. Pothole Detection Using Deep Learning Models. The potholes are considered as the objects to be detected. Deep convolutional neural networks (DCNNs) have proven their abilities for many object detection tasks. These object detectors can be one-stage object detectors or two-state object detectors. Several object detection models such as region-based convolutional neural network family (R-CNN) [31], YOLO family [35], and SSD family [36], are available in deep learning for training. However, the R-CNN family is computationally expensive result in low latency. Conversely, YOLO and SSD are under study to supplement the responsiveness issues of the R-CNN family. Hence, we had focused on YOLO and SSD family for this problem.

2.2.1. YOLO Family. YOLO (you only look once) was first introduced in 2016 by Redmon [35]. It divides the input image into SxS grid cells where each cell is responsible for detecting an object and predicting its bounding box coordinates. Each object bounding box shows the *X*, Y coordinates, height (h), width (w), and confidence score with the class label. The confidence score is the matching percentage of the actual labeled object bounding box with the predicted bounding box and tells the accuracy of prediction of the bounding box. It can detect, classify, and localize multiple objects in one step, whereas other algorithms require multiple scanning of an input image. This algorithm was the milestone for real-time object detection known as YOLOv1. YOLOv1 exhibits some limitations when small and cluttered objects are detected. These drawbacks were removed in 2016 and presented as YOLO900 or YOLOv2 [37]. The improved version offered significant features such as better speed, performance, and accuracy. This version included advanced techniques such as batch normalization and anchor boxes. In 2018, "YOLOV3: an incremental improvement," another improved version was proposed [38] which was even better and stronger than previous versions. Figure 4 shows that YOLOv3 outperforms state-of-the-art detectors such as RetinaNet, SSD, and its variants. It made a significant improvement in terms of speed over other detection models, as shown in Figure 4.

YOLOv4 [39] was developed by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao on April 23, 2020, claiming up to 10% improvement in mean average precision as well as 12% in frames per second as compared to YOLOv3. In addition, Tiny YOLOv4 is a compressed version of the original YOLOv4 that is simpler and faster real-time object detector [40]. After the YOLOv4 release, a company named "Ultralytics" came up with the YOLOv5 by Glenn Jocher [41]. It is different from previous YOLO releases as its implementation is in the PyTorch framework keeping in mind the v5 is not a fork of the original, unlike Alexey's repository. They claim it to be extremely lightweight and speedy than the YOLOv4. However, the accuracy is comparatively equal to YOLOv4.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	59.1	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	22
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 4: Detection model performance as compared to YOLOv3 [38].

which helps in training and minimizing inference time. It takes an input image of resolution 608 × 608. On the other hand, Tiny-YOLOv4 takes input images of resolution 416 × 416. Experimentation proved YOLOv4 as the best real-time object detector without compromising accuracy and outperformed the efficient detection model, EfficientDET. YOLOv5 is the latest version with the same architecture (backbone, neck, and output block) as YOLOv4 but implemented in a PyTorch framework. YOLOv5 passes 640 × 640 resolution images to the backbone, and features are extracted using BottleneckCSP. Four models (YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x) are created by adjusting the height and the width of BottleneckCSP. YOLOv4 surpassed YOLOv5 in mAP when tested on COCO benchmark despite claiming an improvement of YOLOv4, as shown in Figure 5.

2.2.2. Architectures. YOLOv2 uses Darknet19 composed of 19 convolutional layers, five max-pooling layers, and a softmax layer at the end for assigning class labels. The overall mAP of YOLOv2 increased up to 4% by taking the input image of size 448 × 448 from 224 × 224. In YOLOv2, anchor boxes perform the prediction and localization and are responsible for predicting the bounding box. The fine-grained features of YOLOv2 enhance the quality to detect small objects. YOLOv3 uses logistic regression instead of the softmax layer for predicting class probabilities and objectiveness scores. YOLOv3 can detect objects at different scales using the feature pyramid network (FPN). It uses Darknet-53 deeper than Darknet-19 as it contains 53 convolutional layers of feature extractor comparatively. The standard input image size of YOLOv3 architecture is 416 × 416. The YOLOv4 architecture combines three main blocks starting with the CSPDarknet53 as the backbone, following the neck block that adds layers between the head and the backbone CSPDarknet53. A path aggregation network (PANet) is used for feature aggregation to improve the overall accuracy within YOLOv4. It also uses the spatial pyramid pooling (SPP) block to segregate essential features from the backbone. As in YOLOv3, the head performs the detection along with other techniques like bag of freebies and bag of specials

2.2.3. SSD-Mobilenetv2. Mobilenetv2 is a single-shot object detector created using the TensorFlow object detection. API was released by Google researchers in 2018 as an improved version of Mobilenetv1. It generates bounding boxes and class probabilities in a single step. MobileNet is integrated with SSD as designed for mobile and embedded applications. Mobilenetv2 contains two blocks with three layers. The basic building block of Mobilenetv2 is a bottleneck depth-separable convolution with residual block. The second block is for downsizing with the stride of two. Mobilenetv2 is suitable for real-time applications as its speed is high along with shorter inference time but with compromised accuracy. However, Mobilenetv2 is 35% faster than the Mobilenetv1.

2.2.4. AI-on-the-Edge Implementation. We have mentioned that we have opted OAK-D camera on Raspberry Pi for real-time implementation of the proposed approach. To make our OAK-D computational, we need a host computer that has a USB port to plug in OAK-D; for this, we used both windows and raspbian. The next step is to install DepthAI that is a computer vision library provided by Luxonis to get our model running; and after installing DepthAI requirements, we were able to run over a custom model on OAK-D. OAK-D sensor has been chosen because it is a SpatialAI tool,

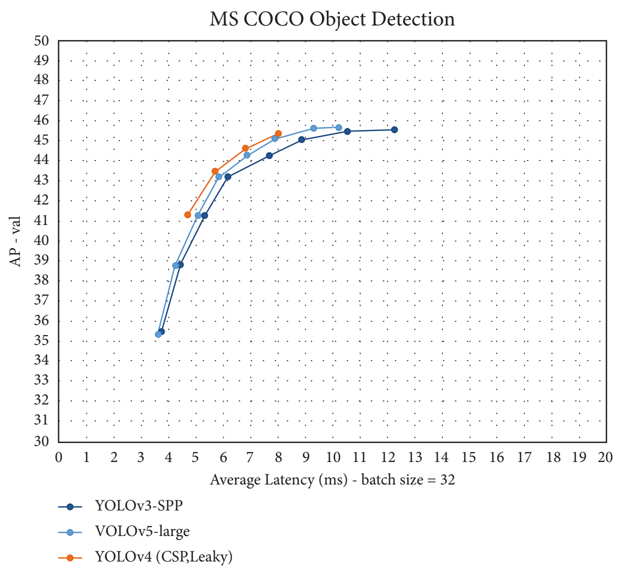


Figure 5: Performance of YOLOv5 as compared to other models [42].

capable of running complex neural networks while providing depth through its left and right stereo cameras and detection from the 4K RGB middle camera.

However, the darknet framework is not optimized to run on the Myriad *X* VPU hardware in Luxonis OAK-D. To run our custom model on OAK-D for real-time detection, we first need to translate our darknet YOLO weights to OpenVino format. We first need to convert to TensorFlow.pb weights and then to OpenVino as we do not have any direct conversion method for darknet weights. After the conversion, we get the.blob file to deploy on the OAK-D kit. For YOLOv5 to deploy on OAK-D, we have converted the YOLOv5 best.pt PyTorch weights to the ONNX framework,.xml and.bin file, and the.blob file, respectively. SSD-Mobilenetv2 weights are also converted to OpenVino.blob file by converting Tensorflow.pb to.xml and.bin file to OpenVino IR representation that is.blob file.

3. Experimentation and Results

3.1. Frameworks Used

3.1.1. Darknet Framework. We used YOLOv1, YOLOv2, YOLOv3, YOLOv4, and Tiny-YOLOv4 for training. YOLO is trained using an opensource neural network framework which is fast due to CUDA and C language providing the real-time attribute for our detection.

3.1.2. PyTorch Framework. An opensource deep learning framework reduces the gap between research and practical application. It is used for the training of YOLOv5 as the darknet framework does not support YOLOv5.

3.1.3. TensorFlow Framework. TensorFlow is an opensource deep learning and machine learning framework by Google that provides researchers with a wide range of tools and libraries for different machine learning and deep learning development and deploy applications. We used the TensorFlow framework for the training of SSD-Mobilenetv2.

3.2. Dataset Annotation. After the collection of a dataset, the next step was to manually annotate them; so, for the annotation, we have used labeling. It is a free graphical image annotation tool that generates labels in YOLO darknet format. For training the YOLO model, the annotations should be in YOLO format as < object-class> < *x*> < y> < width> < height > , where object class is an integer value starting from 0 up to the number of classes defined; in our case, the object class will be 0 as we have only one class, i.e., pothole and the remaining parameters are the coordinates, height, and width of the labeled object bounding box.

The YOLOv5 annotation format is a bit different from the YOLO darknet format, so the conversion is needed here. As YOLOv5 implementation is in PyTorch, its annotation format is < class_id > < center_x > < center_y > < width > < height > , where the class id is normalized to 1 from 0 and remaining parameters are same as YOLO darknet.txt annotation format. The other thing needed for dataset preparation is ‘data.ymal’ file which contains the number of classes, a path to train and validation folder, and lastly the class names. After annotation, the dataset is split into train and test folders with a ratio of 80% for training and 20% for testing. Each folder contained the images corresponding to its annotation.txt file having identical file names.

The SSD-Mobilenetv2 implementation is in the TensorFlow object detection, so the dataset annotations must be converted to the TensorflowTF record from Darknet txt for custom object training. The annotation format used for this model is a CSV file that contains the filename, width, and height of image, class name, and x_{min}, y_{min}, x_{max}, y_{max} coordinates of the labeled pothole. The dataset split ratio has been kept the same as YOLOv5 and YOLO Darknet. The train and test folder contains the images with the CSV annotation file each. The labelmap.pbtxt for train and test is written which contains the class name and id of the labeled objects in each folder.

3.3. Experimentation Protocols. The training of the YOLO and its variants are carried out on a system having Intel (R) Xeon (R) CPU at 3.0 GHz, RAM of 64 GB, and NVIDIA Titan Xp GPU. The dataset is split into 80% (1,066 images) training of the model and 20% (264 images) for the testing with labels of each image. The files needed for training are obj.names (names of the classes), obj.data (the number of classes), a path to train, test, and a backup folder. The backup folder saves the weights after every 100th iteration. The major file required for training is the configuration file which changes according to the model requirements. In our case, each model is trained for 20,000 max iterations with the batch size of 64 having subdivisions of 32 and a learning rate of 0.001 enclosed in the .cfg configuration file. The filters are set to 18 according to the formula filter size = (class+5) ∗ 3 where class = 1 in a .cfg file. For the training of YOLOv5, same parameters are used.

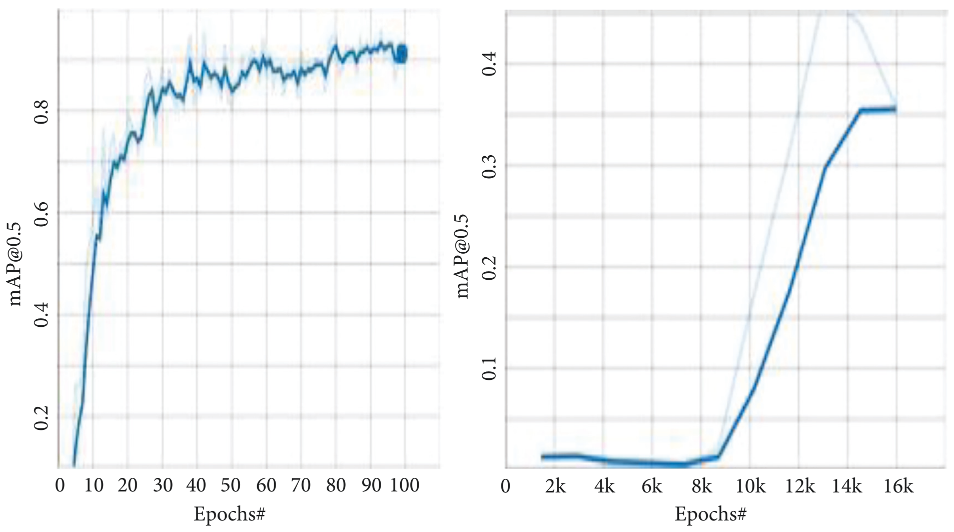


Figure 6: Accuracy graphs of (a) YOLOv5 when trained for 100 epochs and (b) SSD-Mobilenetv2 when trained for 16,000 batches.

According to the model and performance, we used pretrained weights from Google. Here, pretrained weights of YOLO v1, v2, v3, v4, v5, and Tiny-YOLOv4 were downloaded for training and used as transfer learning. YOLOv5 has less training time as compared to the rest of the YOLO models as 1 hour max, as shown in Figure 6(a). For the training of SSD-Mobilentv2, we used 20,000 training steps, 50 evaluation steps, and an initial learning rate of 0.004 for better accuracy in the configuration file. The TensorFlow object detection pretrained model was used for training. The overall training time for SSD-Mobilenetv2 taken was 3-hours 8-min approximately for 20,000 training steps, as shown in Figure 6(b).

3.4. Performance Criteria. The performance of each model has been evaluated with performance metrics (mAP, precision, recall, F1-score, and average inference time per image) for all the trained deep learning models, as shown in equations (1) to (5). By adjusting a threshold and observing the precision and recall values, the model is evaluated. N thresholds are assumed for the precision and recall calculations, with each threshold consisting of a pair of precision (P_n) and recall (R_n) ($n = 1, 2, \ldots,$ N). Average precision (AP) is defined by equation (4), and mean average precision (mAP) defined by (1) is the average of AP of each class. In our case, the AP and mAP will be same as we have only one class. The measure of overlapping area between the predicted bounding box and the ground truth bounding box is compared with the defined threshold called Intersection over Union (IOU). For this work, we have set the threshold to 0.3. Therefore, a prediction is correct if IOU score is greater than or equal to the threshold of 0.3 (30%). The precision, recall, and F1-score of YOLOv2 and YOLOv3 are nearly same, but the mAP@0.5 for YOLOv2 is 81.21% and 83.60%. However, the inference time of YOLOv2 is 33.7 ms which is quite good, as shown in Table 1. YOLOv4 and Tiny-YOLOv4 have achieved mAP@0.5 of 85.48% and 80.04%, respectively, whereas the inference time of Tiny-YOLOv4 is 4.8 ms which is very low as compared to the pure YOLOv4. YOLOv5 showed the highest mAP@0.5 of 95% with an inference time of 10 ms per image. The inference time of Tiny-YOLOv4 is the lowest to deploy on edge devices such as Raspberry Pi, Google Coral, and NVIDIA Jetson Nano. We found out SSD-Mobilentv2 can be run for real-time detection but did not perform for our problem as mAP is 47.4% which is not even close to the mAP of the YOLO family. Table 1 presents the overall evaluation parameters of other deep learning models.

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{2}$$

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{3}$$

$$AP = \sum_{n=1}^{N} (R_n - R_{n-1}) P_n, \tag{4}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i. \tag{5}$$

After looking into the pothole detection result using SSD-Mobilenetv2, we concluded that SSD-Mobilenetv2 does not detect objects that rely upon the appearance of environment like potholes because it does not consider its neighboring pixels, unlike YOLO. YOLO divides an image into grid cells of equal size. Each cell is to detect the object that lies in the center. Furthermore, SSD-Mobilenetv2 fails small pothole detection as our dataset contained small bounding boxes as well.

Figure 7 shows the detection results obtained from each trained model. The red circled pothole is not detected by YOLOv5 and SSD-Mobilentv2 in Figure 7. In contrast, the YOLOv4 even detected the small potholes, as well as long-distance potholes with a considerable confidence threshold of 0.67, encircled red in Figure 7.

Table 1: Performance evaluation of each model on test subset of pothole image dataset (PID)

Model	Precision	Recall	F1-score	mAP@0.5 (%)	Inference time (ms)
SSD-Mobilenetv2	0.42	0.56	0.479	47.4	7
YOLOv1	0.82	0.69	0.74	79.55	340
YOLOv2	0.81	0.76	0.78	81.21	33.7
YOLOv3	0.77	0.78	0.78	83.60	70.57
Tiny-YOLOv4	0.76	0.75	0.76	80.04	4.86
YOLOv4	0.81	0.83	0.82	85.48	52.51
YOLOv5	0.93	0.83	0.87	95.00	10

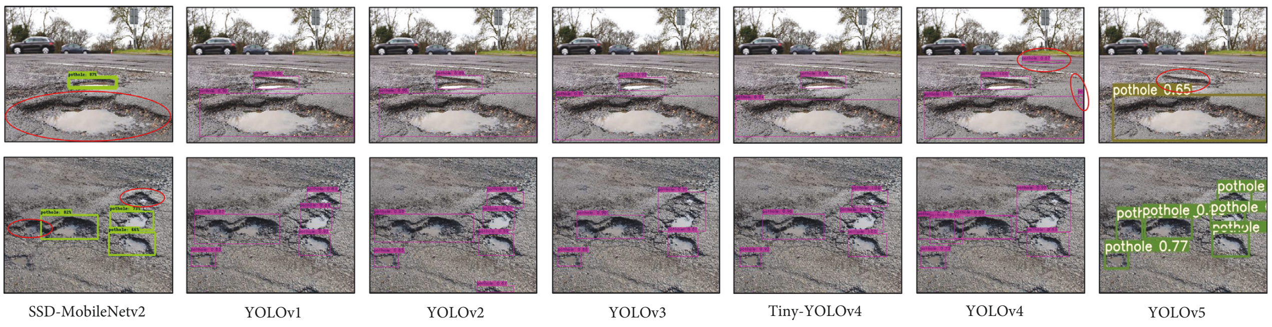


Figure 7: Prediction of potholes by YOLO family and SSD-Mobilnetv2.

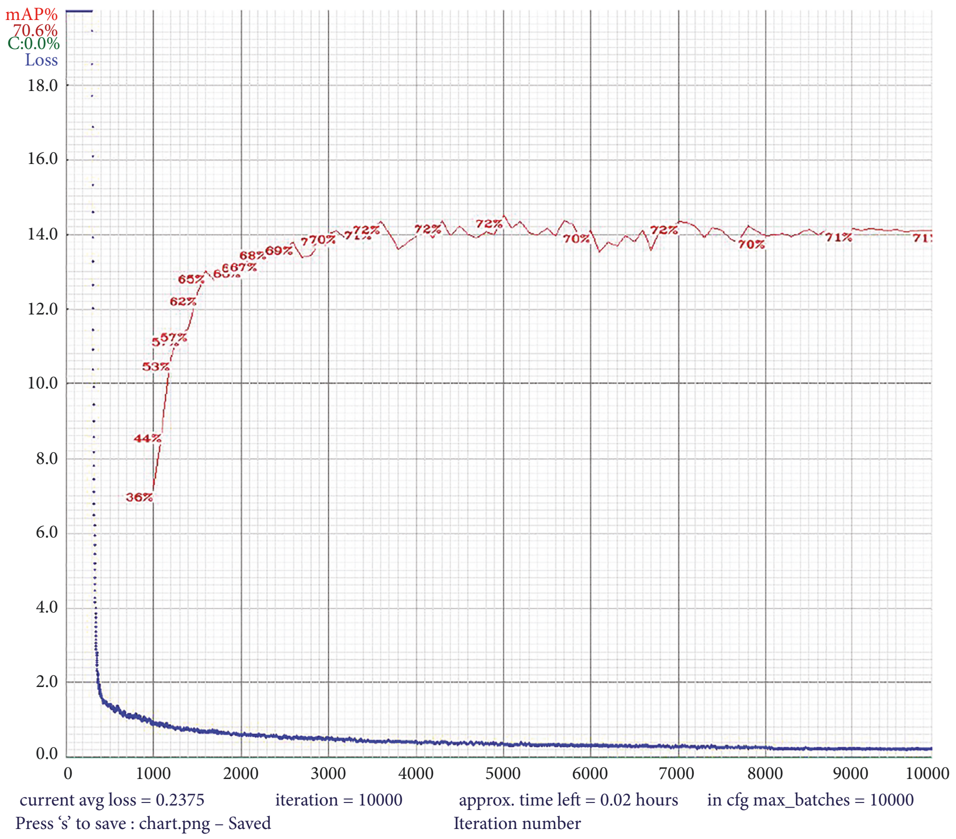
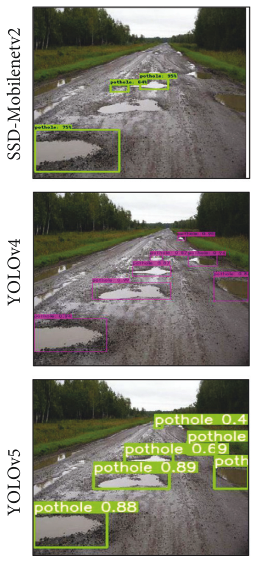


Figure 8: Average loss vs. iterations and mAP graph of YOLOv4 when trained for 10,000 iterations.

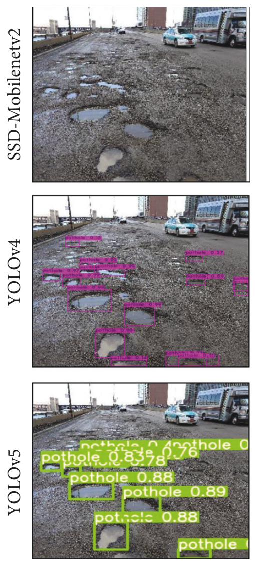
3.5. Accuracy Graphs. Figure 6(a) shows 95% as the best mean average precision for YOLOv5 for the present case. Similarly, the training loss is nearly equal to zero that is 0.02. Figure 6(b) shows the mAP and the total number of iteration set. Initially, iteration set to train SSD-Mobilenetv2 was 20000. After the 14,000th iteration, mAP started decreasing and became constant at the 16,000th iteration. Thus, the training is stopped at the 16,000th iteration, with an average

Table 2: Qualitative analysis on test images.

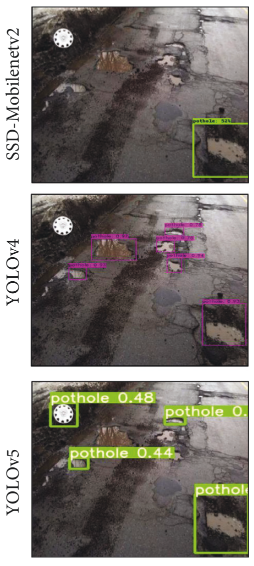
Image ID	No. of Labelled potholes	Detected by YOLOv4	Detected by SSD-Mobilenetv2	Detected by YOLOv5	False detection by YOLOv5
a	7	7	3	6	0
b	11	11	0	9	0
c	6	6	1	3	1
d	9	9	3	9	3
e	8	8	1	4	0
Average detection	—	100%	21%	73%	—



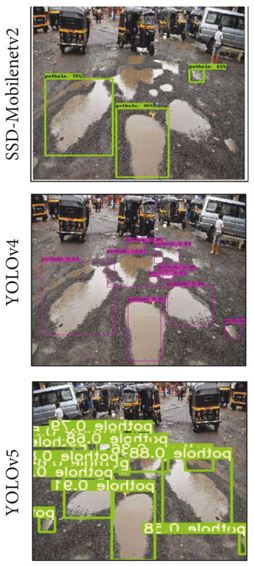
(a)



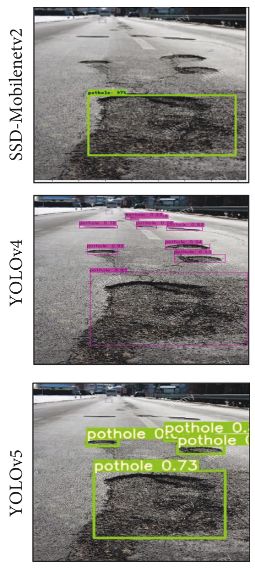
(b)



(c)



(d)



(e)

Figure 9: Qualitative analysis of SSD-Mobilenetv2, YOLOv5, and YOLOv4 on test images.

loss up to 4.5 at the end of training. Figure 8 shows average loss vs. no. of iterations as well as mAP of Tiny-YOLOv4. The loss and mAP became constant at the 4,000th iteration, so we continued training for 10,000 iterations. At last, 71% mAP@0.5 has been achieved with an average iteration loss of 0.237.

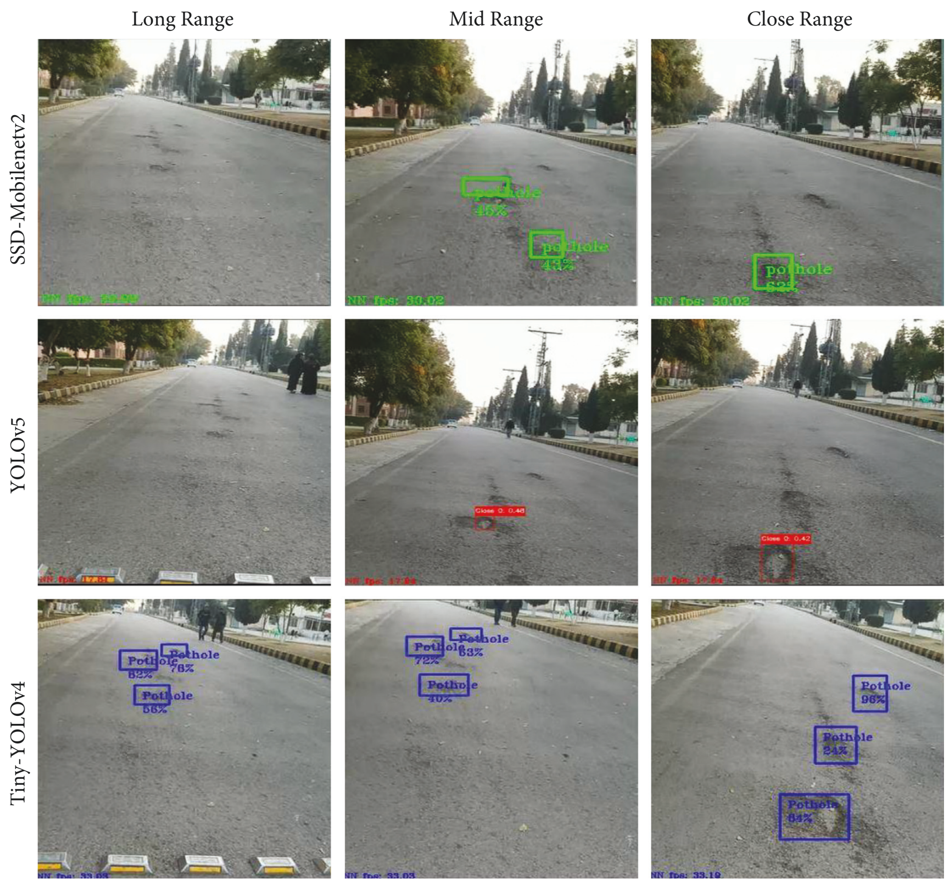


Figure 10: Real-time detection of potholes using OAK-D and Raspberry Pi on different distance ranges.

Table 3: Real-time detection performance evaluation of each model.

Model	No. of potholes	Detected Potholes	Accuracy (%)	FPS
SSD-Mobilenetv2	10	4	40	26.65
YOLOv5	10	5	50	18.25
YOLOv2	10	8	80	3.20
YOLOv3	10	9	90	2.39
YOLOv4	10	10	100	1.98
Tiny-YOLOv4	10	9	90	31.76

3.6. Qualitative Analysis. Table 2 presents the qualitative analysis of five test images. YOLOv4 performed well as compared to YOLOv5, while SSD-Mobilentv2 showed unsatisfactory results. As shown in Figure 9 Image ID c, YOLOv5 misclassified objects as potholes and does not detect potholes despite being visible. In Figure 9 Image ID b, SSD-Mobilenetv2 shows no detection of potholes provided many potholes are in the scene. YOLOv5 and SSD-Mobilenetv2 could not detect potholes with long distances from the camera. However, based on these analyses, YOLOv4 performed 100% accurately. The detections done by SSD-Mobilenetv2, YOLOv5, and YOLOv4 are present in Figure 9. Average detection is measured using (2).

$$\text{Average\_Detection} = \left(\frac{\text{Detected\_Potholes}}{\text{Total\_Potholes}}\right) * 100. \qquad (6)$$

3.7. Real-Time Detection of Potholes by Our System. We have conducted real-time pothole detection using OAK-D and Raspberry Pi on three different locations and distance ranges (Long-Range ≅ 10 m, Mid-Range ≅ 5 m, Close-Range ≅ 2 m). In Figure 10, YOLOv5 and SSD-Mobilentv2 did not detect the potholes located on the long-distance range and even missed the potholes in mid range and close range. However, Tiny-YOLOv4 detects all the potholes with the highest confidence score up to 96% in all defined distance ranges.

For real-time testing on vehicles, OpenCV AI Kit (OAK-D) has been mounted at the center of the vehicle dashboard to capture the maximum road area possible for better evaluation. However, the speed is constant at 65 km/h throughout the experiment. Raspberry Pi acted as a host computer for OAK-D. The Tiny-Yolov4 detects potholes at a distance of 10 meters from the dashboard with a high FPS of 31.76.

Table 4: Comparison with state-of-the-art vs. our results.

Contribution	Dataset	mAP@0.25	mAP@0.5	Inference time (ms)
Omar et al. [43]	Pothole image dataset (PID)	60%	—	—
Shaghouri et al. [44]	Pothole image dataset (PID)	75.53%	—	—
Gajjar et al. [45]	Self + Online collected	—	18.5%	481
Sung-Sik et al. [46]	Pothole image dataset (PID)	—	74.8%	—
Our trained SSD-Mobilnetv2	Pothole image dataset (PID)	—	47.4%	7
Our trained YOLOv3	Pothole image dataset (PID)	83.60%	—	70.57
Our trained YOLOv4	Pothole image dataset (PID)	85.48%	—	52.51
Our trained YOLOv5	Pothole image dataset (PID)	—	95%	10

Tiny-YOLOv4 is considered the best model to implement for real-time pothole detection systems as it has maximum FPS with the highest detection accuracy compared to YOLOv2, YOLOv3, and YOLOv4. SSD-Mobilenetv2 has shown low performance as it only detects when the confidence threshold is 30% or less having false and no detection. YOLOv5 has 18.25 FPS and misses a large number of potholes during real-time inference. However, it is fruitful for real-time pothole detection systems with high FPS but lower accuracy. The real-time detection results are present in Table 3. The testing is done in a completely unknown environment as we trained our models on the Pothole image Dataset.

3.8. Comparison. We have compared our results with other state-of-the-art techniques, showing that our YOLOv4 trained model has performed better in detection with minimum inference time. Shaghouri et al. [44] used a pothole image dataset with 75.63% mAP using YOLOv4; the trained YOLOv4 has achieved 85.48%, which is 9.85% more accurate; whereas [45] used self-collected dataset and achieved mAP@0.5 of 18.5% with higher inference time using SSD-Mobilentv2. Researchers in [46] used YOLOv5 on the PID and achieved the mAP@0.5 of 74.48% which is in difference of 20.52% as compared to our trained YOLOv5. Table 4 presents the comparison with other state-of-the-art techniques.

4. Conclusion

This work presented the state-of-the-art deep learning models (YOLO family and SSD-mobilenetv2) for real-time pothole detection leading towards the deployment on edge devices. Although, YOLOv5 showed the highest mAP@0.5 of 95% among other models but exhibits miss-classification and no detection potholes at long distances. Therefore, we concluded the YOLOv4 as the best-fit pothole detection model for accuracy and Tiny-YOLOv4 as the best-fit pothole detection model for real-time pothole detection with 90% detection accuracy and 31.76 FPS. The proposed approach can help road maintenance authorities to formulate rapid and optimized actions for road infrastructure repairs. A more sophisticated solution with the help of the global position system (GPS) can detect and point out the location of pavement failures. This work can contribute to self-driving applications and the automation industry. This work can further be extended to detect other pavement distresses, road depressions, classify roads as per quality, and depth estimation of potholes. The accuracy limitations can also be resolved in the future by further modification and extension in the real-time deployment.

Data Availability

The data are available upon request to the author.

Conflicts of Interest

The authors declare no conflicts of interest regarding authorship and publication of this research work.

Acknowledgments

This research work had been carried out at Swarm Robotics Lab under National Center of Robotics (NCRA) which was being funded by Higher Education Commission (HEC), Pakistan. The authors acknowledge the NCRA and HEC for technical and financial support.

References

[1] C. Ng, T. Law, F. Jakarni, and S. Kulanthayan, "Road infrastructure development and economic growth," *in IOP conference series: materials science and engineering*, vol. 512, no. 1, Article ID 012045, IOP Publishing, 2019.

[2] S. Rahman, *A; Patel, "Pothole Image Dataset. Kaggle*, 2020, https://www.kaggle.com/sachinpatel21/pothole-image-dataset/.

[3] W. health, *Statistics, "Injury Deaths Rise in Rank*, 2008, https://www.who.int/violence_injury_prevention/key_facts/VIP_key_fact_3.pdf/.

[4] K. C. Wang, "Challenges and Feasibility for Comprehensive Automated Survey of Pavement conditions," in *Proceedings of the Applications of Advanced Technologies in Transportation Engineering (2004)*, pp. 531–536, Beijing, China, May 2004.

[5] K. Chang, J. Chang, and J. Liu, "Detection of pavement distresses using 3d laser scanning technology," in *Proceedings of the Computing in Civil Engineering (2005)*, pp. 1–11, Cancun, Mexico, May 2005.

[6] Z. Hou, K. C. Wang, and W. Gong, "Experimentation of 3d pavement imaging through stereovision," in *Proceedings of the International Conference on Transportation Engineering 2007*, pp. 376–381, Chengdu, China, July 2007.

[7] P. Harikrishnan and V. P. Gopi, "Vehicle vibration signal processing for road surface monitoring," *IEEE Sensors Journal*, vol. 17, no. 16, pp. 5192–5197, 2017.

[8] B. X. Yu and X. Yu, "Vibration-based system for pavement condition evaluation," in *Proceedings of the Applications of Advanced Technology in Transportation*, pp. 183–189, Chicago, IN, USA, August 2006.

[9] K. De Zoysa, C. Keppitiyagama, G. P. Seneviratne, and W. Shihan, "A public transport system based sensor network for road surface condition monitoring," in *Proceedings of the 2007 Workshop on Networked Systems for Developing Regions*, pp. 1–6, Colombo, Srilanka, August 2007.

[10] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pp. 29–39, Breckenridge, CO, USA, June 2008.

[11] A. Glowacz, "Ventilation diagnosis of angle grinder using thermal imaging," *Sensors*, vol. 21, no. 8, p. 2853, 2021.

[12] ——, "Thermographic fault diagnosis of ventilation in bldc motors," *Sensors*, vol. 21, no. 21, p. 7245, 2021.

[13] C. Koch and I. Brilakis, "Pothole detection in asphalt pavement images," *Advanced Engineering Informatics*, vol. 25, no. 3, pp. 507–515, 2011.

[14] G. Jog, C. Koch, M. Golparvar-Fard, and I. Brilakis, "Pothole properties measurement through visual 2d recognition and 3d reconstruction," in *Proceedings of the Computing in Civil Engineering (2012)*, pp. 553–560, Clearwater Beach, FL, USA, June 2012.

[15] L. Huidrom, L. K. Das, and S. Sud, "Method for automated assessment of potholes, cracks and patches from road surface video clips," *Procedia-Social and Behavioral Sciences*, vol. 104, pp. 312–321, 2013.

[16] I. H. Abbas and M. Q. Ismael, "Automated pavement distress detection using image processing techniques," *Engineering, Technology & Applied Science Research*, vol. 11, no. 5, pp. 7702–7708, 2021.

[17] J. Zhou, P. S. Huang, and F.-P. Chiang, "Wavelet-based pavement distress detection and evaluation," *Optical Engineering*, vol. 45, no. 2, p. 027007, 2006.

[18] S. Lee, S. Kim, K. E. An, S.-K. Ryu, and D. Seo, "Image processing-based pothole detecting system for driving environment," in *Proceedings of the 2018 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 1-2, IEEE, Las Vegas, NV, USA, January 2018.

[19] P. Wang, Y. Hu, Y. Dai, and M. Tian, "Asphalt pavement pothole detection and segmentation based on wavelet energy field," *Mathematical Problems in Engineering*, vol. 2017, 2017.

[20] Y. K. Arbawa, F. Utaminingrum, and E. Setiawan, "Three combination value of extraction features on glcm for detecting pothole and asphalt road," *Jurnal Teknologi dan Sistem Komputer*, vol. 9, no. 1, pp. 64–69, 2021.

[21] O. A. Egaji, G. Evans, M. G. Griffiths, and G. Islas, "Real-time machine learning-based approach for pothole detection," *Expert Systems with Applications*, vol. 184, p. 115562, 2021.

[22] P. Ping, X. Yang, and Z. Gao, "A deep learning approach for street pothole detection," in *Proceedings of the 2020 IEEE Sixth International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 198–204, IEEE, Oxford, UK, August 2020.

[23] W. Ye, W. Jiang, Z. Tong, D. Yuan, and J. Xiao, "Convolutional neural network for pothole detection in asphalt pavement," *Road Materials and Pavement Design*, vol. 22, no. 1, pp. 42–58, 2021.

[24] R. Agrawal, Y. Chhadva, S. Addagarla, and S. Chaudhari, "Road surface classification and subsequent pothole detection using deep learning," in *Proceedings of the 2021 2nd International Conference for Emerging Technology (INCET)*, pp. 1–6, IEEE, Belagavi, India, May 2021.

[25] C. Zhang, E. Nateghinia, L. F. Miranda-Moreno, and L. Sun, "Pavement distress detection using convolutional neural network (cnn): a case study in montreal, Canada," *International Journal of Transportation Science and Technology*, vol. 1, 2021.

[26] S. I. Hassan, D. O'Sullivan, and S. McKeever, "Pothole detection under diverse conditions using object detection models," *IMPROVE*, vol. 1, pp. 128–136, 2021.

[27] R. Kavitha and S. Nivetha, "Pothole and object detection for an autonomous vehicle using yolo," in *Proceedings of the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pp. 1585–1589, IEEE, Madurai, India, May 2021.

[28] H. Chen, M. Yao, and Q. Gu, "Pothole detection using location-aware convolutional neural networks," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 4, pp. 899–911, 2020.

[29] M. Rani, M. Mustafar, N. Ismail, M. Mansor, and Z. Zainuddin, "Road peculiarities detection using deep learning for vehicle vision system IOP Conference Series: materials Science and Engineering," *IOP Publishing*, vol. 1068, no. 1, p. 012001, 2021.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, San Juan, PR, USA, June 2014.

[32] S. Antol, A. Agrawal, J. Lu et al., "Vqa: visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433, Santiago, Chile, December 2015.

[33] C. Zhang, Z. Yang, X. He, and L. Deng, "Multimodal intelligence: representation learning, information fusion, and applications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 3, pp. 478–493, 2020.

[34] B. Martinez, P. Ma, S. Petridis, and M. Pantic, "Lipreading using temporal convolutional networks," in *Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6319–6323, IEEE, Beijing, China, January 2020.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788, Los Alamitos, CA, USA, June 2016.

[36] W. Liu, D. Anguelov, D. Erhan et al., "Ssd: single shot multibox detector," in *Proceedings of the European Conference on Computer Vision*, pp. 21–37, Springer, Cham, September 2016.

[37] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, Cham, June 2017.

[38] "Y.3: An Incremental Improvement," 2018, https://arxiv.org/abs/1804.02767.

[39] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal Speed and Accuracy of Object Detection*, 2020, https://arxiv.org/abs/2004.10934.

[40] Z. Jiang, L. Zhao, S. Li, and Y. Jia, *Real-time Object Detection Method Based on Improved Yolov4-Tiny*, 2020, https://arxiv.org/abs/2011.04244.

[41] A. Malta, M. Mendes, and T. Farinha, "Augmented reality maintenance assistant using yolov5," *Applied Sciences*, vol. 11, no. 11, p. 4758, 2021.

[42] Roboflow, "Y.5," 2020, https://models.roboflow.com/object-detection/yolov5.

[43] M. Omar and P. Kumar, "Detection of roads potholes using yolov4," in *Proceedings of the 2020 International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1–6, IEEE, Tashkent, Uzbekistan, November 2020.

[44] A. A. Shaghouri, R. Alkhatib, and S. Berjaoui, "Real-time pothole detection using deep learning," 2021, https://arxiv.org/abs/2107.06356.

[45] K. Gajjar, T. van Niekerk, T. Wilm, and P. Mercorelli, *Vision-based Deep Learning Algorithm for Detecting Potholes*, 2021.

[46] S.-S. Park, V.-T. Tran, and D.-E. Lee, "Application of various yolo models for computer vision-based real-time pothole detection," *Applied Sciences*, vol. 11, no. 23, p. 11229, 2021.