Indonesian Journal of Electrical Engineering and Computer Science
Vol. 37, No. 2, February 2025, pp. 995~1004
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v37.i2.pp995-1004

Comparative study of pothole detection using deep learning on smartphone

Achyar Ulul Amri, Gede Putra Kusuma
Department of Computer Science, BINUS Graduate Program-Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Article Info

Article history:

Received May 15, 2024
Revised Sep 16, 2024
Accepted Sep 29, 2024

Keywords:

Bayesian search
Deep learning
Hyperparameter tuning
Pothole detection
Smartphone resource usage

ABSTRACT

Potholes present a significant problem in many countries, leading to vehicle damage and traffic accidents. These road imperfections pose safety risks and impose economic burdens. Despite existing detection methods using sensors and computer vision deep learning processed on PCs, a gap remains in deploying cost-effective, widely accessible solutions. This study aims to bridge this gap by developing deep learning models optimized for smartphones, reducing costs and enhancing deployment feasibility. We developed multiple models for pothole detection, utilizing transfer learning and Bayesian hyperparameter tuning to optimize detection accuracy and resource efficiency. Our evaluations focused on computationally light models such as YOLOv8 small, YOLOv8-nano, YOLOv7 tiny, and faster R-CNN MobileNetV3. In terms of detection accuracy, YOLOv8 small and YOLOv8 nano stood out, achieving average precisions (AP) of 83.5% and 82.5%, respectively. YOLOv8 nano proved the most efficient, offering high detection accuracy, a file size three times smaller than YOLOv8 small in TFLite format, and the fastest inference time of 0.72 seconds per image. This study highlights the potential of smartphones in urban pothole detection, contributing to improved road maintenance and urban policy.

Corresponding Author:

Achyar Ulul Amri
Department of Computer Science, BINUS Graduate Program-Master of Computer Science
Bina Nusantara University
Jakarta, Indonesia
Email: achyar.amri@binus.ac.id

1. INTRODUCTION

Road infrastructure conditions are a key safety factor that needs to be specially designed to reduce risks for all road users. However, many roads in low- to middle-income countries have not yet met these [1]. The deterioration of urban infrastructure, including roads, can have significant social implications, affecting the quality of life, public safety, and access to services [2]. Road potholes are one of most issue that present a significant problem in many countries, leading to vehicle damage and traffic accidents [3]. The presence of potholes can sometimes be difficult to identify manually, especially while driving [4]. The existance of fast pothole detection methods is anticipated to aid drivers in avoiding pothole road and can provide valuable data to the government in prioritizing road repairs, ultimately helping to reduce the likelihood of accidents [5].

There are many methods have been developed to detect potholes, including manual inspection, sensor utilization and computer vision [6], [7]. Early adoption of sensors such as accelerometers or GPS in vehicles to detect potholes based on vibrations or irregularities. LiDAR is a more advanced and accurate sensor technology that uses laser light to create high-resolution 3D maps of the road surface albeit at a higher

cost [8]. Computer vision deep learning-based computer vision became most popular method considering the accuracy of detection offered and cost efficiency [9].

There are several pothole detection research studies that utilize deep learning using algorithms from the R-CNN family such as R-FCN, sparse R-CNN, faster R-CNN, and the YOLO family with various configuration variations. Most of the pothole detection models developed are still on PCs, with variations in the datasets used, including adding personal data [10], [11], noisetreatment, brightness variaety [12], and combining resources from multiple image datasets [13]. In previous research has been developed pothole detection model using fine tuning deep learning faster R-CNN ResNet101, which acieved accuracy AP of 82% and inference speed 5-6 FPS (frame per second) when implemented on a computer Nvidia DrivePX2 [14]. Besides, a pothole detection model, SPFPPN-YOLOv4 has been developed and achieved a precision score of 89% with FPS value of 38, implemented on a PC with CPU Intel i7-10700k CPU@3.80GHZ [15].

Pothole detection on mobile devices is relatively rare, with one example using YOLOv3 on an IoT device, achieving 40% AP and 0.83 second speed [16]. While recent research has explored mobile implementation, most studies focus on PC-based model comparisons with future mobile deployment plans [15], [17], [18]. Comparative studies have examined the performance of YOLO and faster R-CNN for pothole detection [19], [20]. Comparative studies show YOLOv7 outperforms YOLOv6 and v5 in detection task [21]. YOLOv7 can also be combined with smartphone sensors such as accelerometers to improve detection performance [22]. New architecture YOLOv8 could achieves 92% detection accuracy on PC. YOLOv8 can be further enhanced with CNNs for improved detection performance [23]. Additionally, YOLOv8 can be combined with smartphone sensors like GPS to track pothole locations [24]. Recently, research has focused on improving the lightweight YOLOv8n for mobile deployment, achieving an AP value of 53.1% on the RDD2022 dataset [25].

On the other hand, more researched have been developed for general object detection on mobile device including smartphone. Object detection models have been applied to the diverse COCO dataset using Android smartphones and Iphone using tflite format of fp16 and int8 optimization [26], [27]. In addition, there are several object detections on smaller objects, including food [28], as well as fly imaging datasets [29] and pests also using smartphones [30]. Furthermore, there are applications on mobile devices such as Raspberry TPU for peach cultivar detection [31], tomato variety detection [32] and Jetson TX2 for crop seedling detection [33]. Some recent research related to object detection model on smartphone or mobile device are Tuna-YOLO Object detection model which has been developed to detect tuna varieties on mobile devices, this algorithm based on YOLOv3 using MobileNet and transfer learning method. Based on the test result it was found that the highes mAP value of 85.74% with performance of 15.32 FPS [34]. One of the latest research is creating a corn seedling detection model with mobile device using variations of the YOLOv7 and YOLOv8 models, which successfully achieved accuracy of around 93% and FPS values of 90 and 125 respectively on mobile device [35].

The literature review highlights a need for pothole detection systems that operate efficiently on devices with limited computational resources, such as smartphones. Although previous research shows that the YOLOv8 algorithm could outperforms other state of the art (SoTA) models like YOLOv7 and faster R-CNN, their application in mobile pothole detection remains underexplored. This research aims to conduct incremental innovation by retraining and evaluating YOLOv8, YOLOv7, and faster R-CNN models using transfer learning and Bayesian hyperparameter tuning [36], [37]. The focus will be on small, tiny, or nano variants with less than 20 million parameters to ensure efficient deployment on smartphones. The goal is to assess and compare these models in terms of both detection performance and computational resource efficiency, thereby advancing intelligent transportation systems. The research is expected to enable real-time pothole detection on mobile devices and provide valuable data for infrastructure maintenance and planning.

2. METHOD

2.1. Proposed system

Automatic pothole detection is intended to help prevent and minimize traffic accident risks. This study introduces advanced models that can be implemented on devices with limited computational capacity, like the widely-used smartphones. The research involved training these models through transfer learning and hyperparameter tuning procedure on SoTA algorithms, including faster R-CNN MobileNetV3, YOLOv7 tiny, and YOLOv8 small, using Google Colab T4 and python programming language. Additionally, the implementation procedure was tested on an Android smartphone, specifically the Redmi Note 12. The evaluation assessed both detection accuracy and the computational resource usage, aiming to identify the model that offered the best balance of detection accuracy and efficiency for real-world pothole detection on mobile devices [26].

The steps involved in the proposed model, as illustrated in the Figure 1:

- Data input: collect raw image data for analysis.
- Data preprocessing: prepare images through cropping, resizing into 800×800, and normalization.
- Label re-formatting: convert labels to YOLO and Pascal VOC formats suitable for different models.
- Data splitting: divide the preprocessed data into training, validation, and testing subsets.
- Training and fine tuning: train and fine-tune models like YOLOv7 tiny, YOLOv8 small, and faster R-CNN MobileNetV3 using a Bayesian search method, with 10 iterations conducted for each model.
- Model conversion: convert each best tune models to TFLite formats (FP16, INT8) where applicable.
- Detection accuracy testing: test accuracy on mobile-optimized models.
- Model app deployment: deploy models within a android app which created using android studio.
- Sampling for test simulation: create 10 test susets with 50 images each for real-world simulations.
- Resource usage testing: measure each model's inference time and resource usage on smartphones using Android studio profiler.

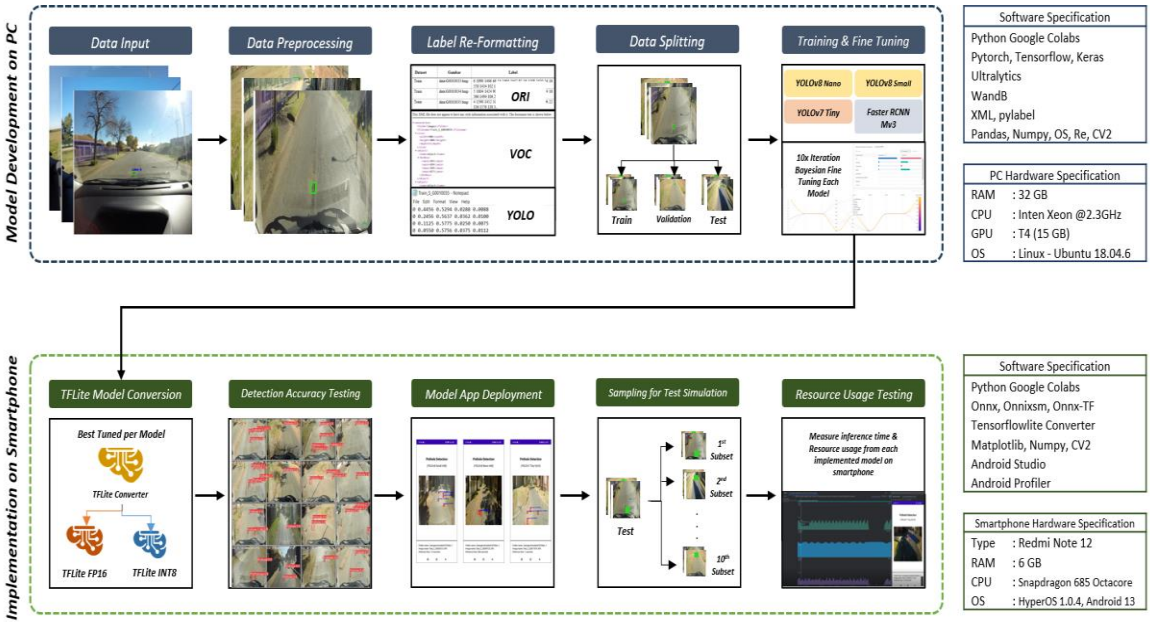


Figure 1. Flow diagram of the pothole detection system from development to smartphone deployment

2.2. Dataset

WHO notes that low- to middle-income countries often often failing to meet safety standards [1]. To reflect these conditions, this study uses South Africa as a representative country, drawing on a dataset from Nianaber et al. focused on pothole detection [6]. The data was collected using a GoPro Hero 3+ camera mounted as a dashcam, capturing images at a resolution of 3680×2760 pixels in JPG format. The dataset consists of 13.4K images, but only 3.8K images containing labeled potholes were used, supplemented by 400 negative class images. This results in a total of 4.1K images analyzed for this study, with each image labeled by experts to identify potholes, ranging from 0 to more than 1 per image, as illustrated in Figure 2.

Negative (0) Positive (1) Positive (>1)

Figure 2. Sample dataset of pothole road

2.3. Data preparation

The original dataset images are taken from relatively fixed position dashcam it is not only captured the raod but also car dashboard and sky. The previous research suggest to downscale and crop the image and focus on the road section [38]. Then it was transformed using a Python program. This process included re-labeling data into Pascal VOC and YOLO formats, cropping images to focus on the road, resizing to 800×800 [14], normalizing images, and splitting the data into 1,672 train, 1,254 validation, and 1,254 test images.

2.4. Faster R-CNN

Faster R-CNN is a two-step object detection method that first proposes object regions from an image before performing classification [39]. In the context of pothole detection, the model is trained using a dataset of images containing potholes, where each image is annotated to indicate the location of the potholes. MobileNetV3, a variation of faster R-CNN, has a relatively small number of parameters, around 20 million, and offers good detection accuracy, making it suitable for use on mobile devices [40].

2.5. YOLOv7

YOLOv7 is a single-step object detection method that includes a backbone, head, and neck, and utilizes extended efficient layer aggregation network (E-ELAN) blocks and scalable bag of freebies (BoF) techniques [41]. When developing a pothole detection model, YOLOv7 is trained on an annotated image dataset that marks the locations of potholes. The model efficiently learns to detect and localize potholes in one pass through the image. YOLOv7 tiny, with 6.2 million parameters, could offer good detection accuracy, and suitable for implementation on smartphones.

2.6. YOLOv8

YOLOv8 is a single-step object detection method comprising a backbone, head, and neck, utilizing ConvModule blocks, CSPLayer 2conv (c2f), and SPPF. This model is also anchor-free, enhancing its detection capabilities [42]. For developing a pothole detection model, YOLOv8 is trained on a dataset with annotated pothole locations, enabling the model to efficiently detect and localize potholes in a single pass. YOLOv8 small, with 11.2 million parameters, and YOLOv8 nano, with 3.2 million parameters, offer relatively good detection accuracy and well-suited for mobile devices implementation.

2.7. Transfer learning and hyperparameter tuning

In this study, transfer learning was employed by fine-tuning pre-trained models on the target dataset. This approach leverages the knowledge from models trained on large datasets, allowing for quicker convergence and improved performance on the specific task [43]. To further enhance the models, Bayesian optimization was used for hyperparameter tuning, chosen for its efficiency in identifying optimal configurations. The Wandb library facilitated monitoring and analyzing the impact of each hyperparameter [44]. In this study, hyperparameter tuning was performed for 10 iterations on each model with search space include optimizers tested being AdamW and SGD. The training ran for 150 epochs, with a patience of 20 for early stopping. Batch sizes of 8, 16, and 32, alongside a learning rate sampled uniformly between 0.001 and 0.01. Momentum within a uniform range of 0.8 to 0.98.

2.8. TensorFlow lite

To deploy object detection models like those built with PyTorch or YOLO on an Android app, the models need to be converted to TensorFlow lite format. The conversion process typically involves exporting the PyTorch or YOLO model to open neural network exchange (ONNX) format first, which serves as an intermediate representation. Then the model being converted to TensorFlow format, followed by a final conversion to TensorFlow lite using TensorFlow lite converter. During this process, quantization techniques like FP16 or INT8 are applied to reduce the model size and computation requirements, optimizing it for efficient on-device inference [26], [27].

2.9. Android studio

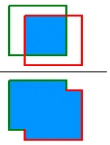
Android studio, the official integrated development environment (IDE) for Android app development, provides robust tools for coding, debugging, testing, and optimizing applications. The pothole detection app was developed using the Giraffe version of Android studio. Additionally, Android studio includes a feature called Android studio profiler, which allows developers to measure app performance in terms of CPU, RAM, and battery usage, ensuring efficient, and optimized app behavior [45].

2.10. Evaluation

Process to identify the most effective deep learning model for pothole detection on smartphone requires evaluating both detection performance and resource efficiency. Each model is analyzed based on these criteria, and results are compared. This process identifies most suitable model for deployment [26].

Detection accuracy is measured using the average precision (AP) metric, which is based on the concepts of IoU, precision, and recall [24]. Intersection over union (IoU) quantifies the overlap between the predicted bounding box and the ground truth, considering only predictions that exceed a set confidence threshold, typically 0.5, to identify true positives (TP), as seen in 1. False positives (FP) occur when the model incorrectly identifies a pothole where none exists, while false negatives (FN) arise when the model misses an actual pothole. With TP, FP, and FN defined, precision and recall are calculated-precision indicates the percentage of correct predictions, as seen in 2, and recall measures the percentage of actual positives that are accurately detected, as seen in 3. Finally, AP is calculated as the AP across all recall levels, represented by the area under the precision-recall curve, as seen in 4. This detection accuracy evaluation is performed on the test data to objectively assess the model's performance.

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\quad}{\quad} \tag{1}$$



$$Precision\ (p) = \frac{TP}{TP+FP} \tag{2}$$

$$Recall\ (r) = \frac{TP}{TP+FN} \tag{3}$$

$$Average\ Precision\ (AP) = \int_{r=0}^{1} p(r)\ dr \tag{4}$$

Smartphone resource usage for each model is evaluated by measuring inference speed, battery usage, and CPU/RAM consumption using Android profiler [45]. Inference speed, measured in seconds, reflects the time required for the model to generate predictions from each image as seen in 5. Battery usage is estimated by Android energy profiler, categorizing energy consumption into light, medium, and heavy intensity levels. The energy usage percentage within the light or less resource-intensive tasks category seen in 6. CPU usage is recorded by CPU profiler as a percentage of total available CPU time as seen in 7, while RAM usage is tracked by monitoring the private memory pages allocated to the app as seen in 8. The evaluation is conducted on 10 subsets of 50 images each, focusing on average inference time, average battery usage, and maximum CPU and RAM usage. These metrics determine overall performance, helping to identify the model with the best inference speed and lowest resource consumption [26].

$$Inference\ Time\ (seconds) = t_{prediction} - t_{input} \tag{5}$$

$$Energy\ Usage\ (\%) = \frac{Energy\ consumed\ by\ the\ app\ in\ light\ category}{Total\ energy\ allocated\ for\ light\ use} \times 100 \tag{6}$$

$$CPU\ Usage\ (\%) = \frac{CPU\ cycles\ used\ by\ the\ app}{Total\ CPU\ cycles\ available\ on\ smartphone} \times 100 \tag{7}$$

$$RAM\ Usage\ (MB) = Total\ prvate\ memory\ allocated\ to\ the\ app \tag{8}$$

3. RESULTS

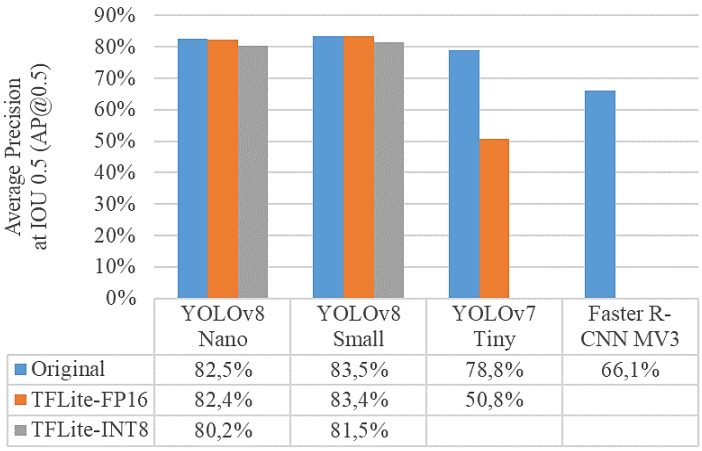
Pothole detection model has been developed on PC using several pretrained model of algorithm which are YOLOv8-small, YOLOv8-nano, YOLOv7-tiny and faster R-CNN MobileNetV3. Each model has experience 10 iteration of hyperparameter tuning such as batch size, learning rate, and momentum. Best hyperparameters for each model based on tuning result shown in Table 1.

The Table 1 indicates that on the validation dataset, YOLOv8 Small achieved the highest AP at IoU 0.5 (AP@0.5) score of 87.8%. YOLOv8 nano followed with a strong 84.9%, both benefiting from the AdamW optimizer. In contrast, YOLOv7 tiny and faster R-CNN MobileNetV3 showed lower validation performance, with AP@0.5 scores of 78.8% and 66.1% respectively, with the latter using the less effective SGD optimizer. This highlights the effectiveness of YOLOv8 models and the AdamW optimizer in achieving

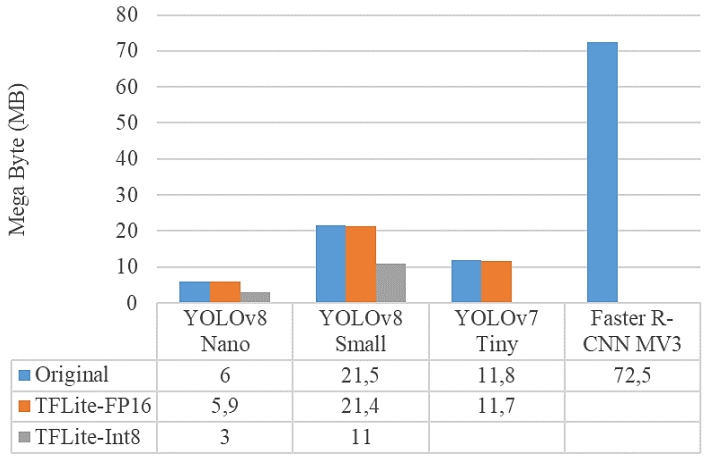
better validation results. Next, each of the models were try to be converted to TFLite format (float16 and int8) and evaluated on test dataset. The models detection performance on PC in Figure 3 its original format and after conversion to TFLite is shown in Figure 3(a), while the models file size comparison is illustrated in Figure 3(b).

Table 1. Best hyperparameter tuning result for each model

Model	Img size	Batch	Epochs	lr0	Momentum	Optimizer	AP@0.5 validation
YOLOv8 small	800×800	8	150	0.0014	0.908	AdamW	87.8%
YOLOv8 nano	800×800	32	150	0.0010	0.883	AdamW	84.9%
YOLOv7 tiny	800×800	16	150	0.0076	0.957	AdamW	78.8%
Faster R-CNN Mv3	800×800	16	150	0.0082	0.894	SGD	66.1%



(a)



(b)

Figure 3. Comparison of models detection performance on (a) test dataset and (b) file size

As shown in Figure 3, the YOLOv8 small and nano models achieved the highest detection accuracy on the test dataset and maintained performance after conversion which higher than previous research [25]. In contrast, YOLOv7 tiny model have a significant accuracy drop after FP16 conversion, so INT8 conversion was not pursued. The faster R-CNN MobileNetV3 model was excluded from TFLite conversion due to its or low accuracy and large file size. Notably, YOLOv8 TFLite-INT8 reduced its original file size by 50%.

Figure 4 showcases a sample comparison of detection results across different models under varying lighting conditions. In bright light, both YOLOv8 small and nano models accurately detect all potholes, while YOLOv7 misses one. However, in low-light conditions with shadows, YOLOv8 small remains precise, identifying all potholes, whereas YOLOv8 nano detects only four, and YOLOv7 captures just three.

This comparison highlights the models' strengths and limitations, particularly in detecting smaller or distant potholes under poor lighting conditions [4].

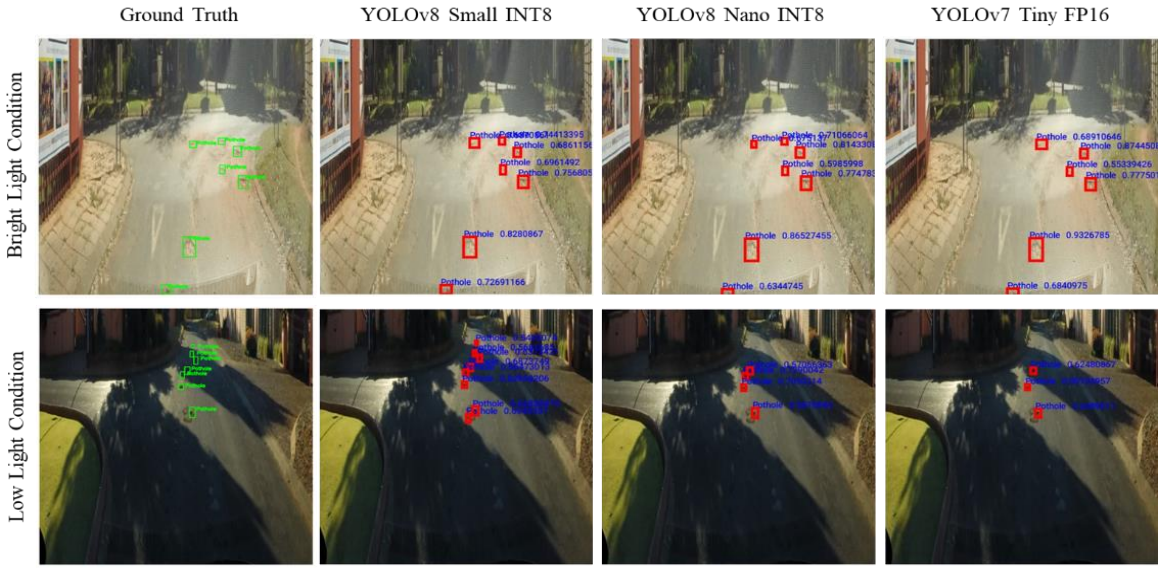
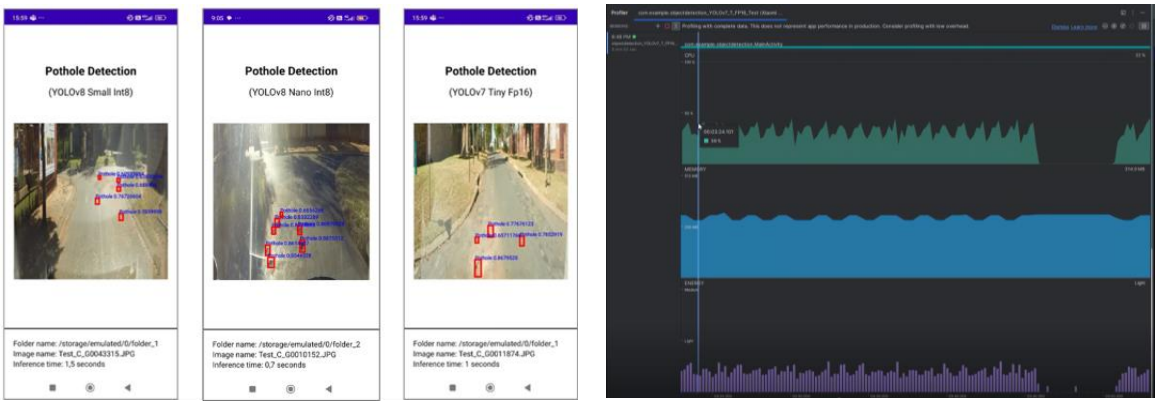


Figure 4. Sample comparison of pothole detection under varying lighting conditions

According to Figure 5 shows the sample of Figure 5(a) showcases the pothole detection models implementation on a smartphone. The models were tested across ten datasets to assess inference speed and detection accuracy. While Figure 5(b) provides insights into resource usage monitored via the Android Profiler, highlighting key metrics such as maximum CPU and RAM usage along with energy consumption over time. It was found that most of the energy usage fell into light energy consumption category. Therefore, the energy usage is shown as a percentage rate within the light energy category for more precise comparison. The monitoring results of inference speed and resource usage are shown in Table 2.



(a) (b)

Figure 5. Sample of (a) smartphone implementation result and (b) monitoring using android profiler

Table 2. Comparison of inference time and resource usage for each model

Metrices	YOLOv8	YOLOv8	YOLOv7
Model TFLite type	Small INT8	Nano INT8	Tiny FP16
Average inference time	1.33 seconds	0.72 seconds	1.04 seconds
Average baterai usage	48%	48%	54%
Maximum of CPU usage	39%	35%	44%
Maximum of RAM usage	166 MB	133 MB	263 MB

According to Table 2, it compares inference time and resource usage across models. YOLOv8 nano INT8 is the fastest and most efficient, with a 0.72 seconds inference time and the lowest CPU and RAM usage. YOLOv8 small INT8 is slightly slower but also resource-efficient. In contrast, YOLOv7 tiny FP16, though quicker than YOLOv8 small, has the highest battery and RAM usage, making it less efficient overall. This suggests YOLOv8 nano-INT8 is the best choice for balancing speed and resource efficiency.

4. DISCUSSION

This study investigated the effectiveness of deploying pothole detection models on devices with limited computational resources, such as smartphones. While earlier studies have explored the impact of deep learning models like faster R-CNN and YOLOv7 for pothole detection, they have not explicitly addressed the optimization and deployment of these models on mobile platforms with constrained resources, such as smartphones [21]. Although previous research shows that the YOLOv8 algorithm could outperforms other SoTA models like YOLOv7 and faster R-CNN, their application in mobile pothole detection remains underexplored [23]. This study aims to achieve incremental innovation by retraining and evaluating lightweight models (less than 20 million parameters) of YOLOv8, YOLOv7, and faster R-CNN using transfer learning and Bayesian hyperparameter tuning [37].

This study's key findings demonstrate that the YOLOv8-based algorithm is the most effective model for pothole detection, outperforming other state-of-the-art models such as YOLOv7 and faster R-CNN. With an AP@0.5 score of 80.2% and an inference time of just 0.72 seconds, YOLOv8 nano INT8 stands out for both its precision and speed, along with minimal resource consumption. The proposed method demonstrated a significantly higher proportion of accurate detections while maintaining low computational demands, indicating its potential for real-time applications on resource-constrained devices like smartphones.

Our study suggest that the YOLOv8 nano INT8 outperforms previous models used in pothole detection. For instance, Zeng and Zhong [25] which reported an AP@0.5 score of 53.1% on the RDD2022 dataset. Additionally, in terms of inference time, YOLOv8 nano INT8 is faster than the YOLOv3 model developed by Gajjar *et al.* [16], which had an AP@0.5 of 34.7% with a 0.84 seconds inference time. The proposed method benefits from higher accuracy and faster inference without adversely impacting computational efficiency, making it more suitable for deployment on devices with limited processing power.

This study explored the optimization of pothole detection models for deployment on smartphones under standard conditions. However, further and in-depth studies may be needed to confirm its effectiveness in varied environmental conditions, especially regarding performance on wet roads, extreme lighting, or different road textures. Such factors could potentially impact the model's detection accuracy and generalizability across diverse real-world scenarios.

Our study demonstrates that models optimized for mobile devices, like YOLOv8 nano INT8, are more efficient and practical for real-world applications than heavier models. Future studies may explore expanding the dataset to include diverse environmental scenarios and investigate feasible ways of enhancing model robustness. Additionally, exploring advanced optimization techniques and adaptive algorithms could further improve detection accuracy without compromising resource efficiency.

Recent observations suggest that deploying optimized pothole detection models on smartphones is both feasible and effective. Our findings provide conclusive evidence that the YOLOv8 nano INT8 model significantly improves detection accuracy and speed without requiring elevated computational resources. This advancement contributes to road safety and infrastructure maintenance by enabling real-time, accessible pothole detection through widely available mobile technology like smartphone.

5. CONCLUSION

This study has shown that YOLOv8 nano INT8 is the most effective model for pothole detection on smartphones, outperforming other models like YOLOv7 and faster R-CNN in both detection performance and resource efficiency. It has successfully demonstrates the feasibility of deploying pothole detection using deep learning on resource-constrained devices like smartphone, making it more accessible and practical.

Despite these achievements, the study has limitations, including performance variability under different environmental conditions and a primary focus on pothole detection. There is room for further improvement in smartphone processing, particularly in model optimization for faster inference speeds.

The implications of this research are significant for enhancing road safety and urban transportation systems. By integrating smartphone-based detection systems into existing infrastructure, cities can develop more intelligent transportation policies, helping to prevent accidents and protect both drivers and pedestrians.

Future work should focus on improving model robustness under diverse conditions and exploring additional applications such as smart and dashcam, and hazard detection. To further advance intelligent transportation systems and safer urban environments.

REFERENCES

[1] WHO, "Geneva: global status report on road safety 2023," 2023. https://www.who.int/teams/social-determinants-of-health/safety-and-mobility/global-status-report-on-road-safety-2023 (accessed Aug. 27, 2024).

[2] G. Ritzer, "Blackwell encyclopedia of sociology," in *The Blackwell Encyclopedia of Sociology*, G. Ritzer, Ed. Oxford, UK, Malden, USA and Carlton, Australia: Blackwell Publishing Ltd, 2007, pp. vii–vii.

[3] R. Bibi *et al.*, "Edge AI-based automated detection and classification of road anomalies in VANET using deep learning," *Computational Intelligence and Neuroscience*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/6262194.

[4] B. Bučko, E. Lieskovská, K. Zábovská, and M. Zábovský, "Computer vision based pothole detection under challenging conditions," *Sensors*, vol. 22, no. 22, p. 8878, Nov. 2022, doi: 10.3390/s22228878.

[5] R. Raj, B. K. Vadapu, R. Naresh, S. Gundu, and R. R. Lekshmi, "An artificial intelligence based assistant to detect, notify and update potholes," in *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation, IATMSI 2024*, Mar. 2024, pp. 1–6, doi: 10.1109/IATMSI60426.2024.10503123.

[6] S. Nienaber, M. J. (Thinus) Booysen, and R. Kroon, "Detecting potholes using simple image processing techniques and real-world footage," *Proceedings of the 34th Southern African Transport Conference (SATC 2015)*, pp. 153–164, 2015, doi: 10.13140/RG.2.1.3121.8408.

[7] M. Rathee, B. Bačić, and M. Doborjeh, "Automated road defect and anomaly detection for traffic safety: a systematic review," *Sensors*, vol. 23, no. 12, p. 5656, Jun. 2023, doi: 10.3390/s23125656.

[8] S. A. Talha, D. Manasreh, and M. D. Nazzal, "The use of lidar and artificial intelligence algorithms for detection and size estimation of potholes," *Buildings*, vol. 14, no. 4, p. 1078, Apr. 2024, doi: 10.3390/buildings14041078.

[9] Y. M. Kim, Y. G. Kim, S. Y. Son, S. Y. Lim, B. Y. Choi, and D. H. Choi, "Review of recent automated pothole-detection methods," *Applied Sciences (Switzerland)*, vol. 12, no. 11, p. 5320, May 2022, doi: 10.3390/app12115320.

[10] J. J. Koh *et al.*, "Autonomous road potholes detection on video," in *Lecture Notes in Electrical Engineering*, vol. 481, 2019, pp. 137–143.

[11] P. A. Chitale, K. Y. Kekre, H. R. Shenai, R. Karani, and J. P. Gala, "Pothole detection and dimension estimation system using deep learning (YOLO) and image processing," in *International Conference Image and Vision Computing New Zealand*, Nov. 2020, vol. 2020-November, pp. 1–6, doi: 10.1109/IVCNZ51579.2020.9290547.

[12] R. Kannan, C. J. Jian, and X. N. Guo, "Adversarial evasion noise attacks against TensorFlow object detection API," in *2020 15th International Conference for Internet Technology and Secured Transactions, ICITST 2020*, Dec. 2020, pp. 1–4, doi: 10.23919/ICITST51030.2020.9351331.

[13] K. R. Ahmed, "Smart pothole detection using deep learning based on dilated convolution," *Sensors*, vol. 21, no. 24, p. 8406, Dec. 2021, doi: 10.3390/s21248406.

[14] J. J. Yebes, D. Montero, and I. Arriola, "Learning to automatically catch potholes in worldwide road scene images," *IEEE Intelligent Transportation Systems Magazine*, vol. 13, no. 3, pp. 192–205, 2021, doi: 10.1109/MITS.2019.2926370.

[15] D. H. Heo, J. Y. Choi, S. B. Kim, T. O. Tak, and S. P. Zhang, "Image-based pothole detection using multi-scale feature network and risk assessment," *Electronics (Switzerland)*, vol. 12, no. 4, p. 826, Feb. 2023, doi: 10.3390/electronics12040826.

[16] K. Gajjar, T. V. Niekerk, T. Wilm, and P. Mercorelli, "Vision-based deep learning algorithm for detecting potholes," *Journal of Physics: Conference Series*, vol. 2162, no. 1, p. 012019, Jan. 2022, doi: 10.1088/1742-6596/2162/1/012019.

[17] B. M. Prakash and K. C. Sriharipriya, "Enhanced pothole detection system using YOLOX algorithm," *Autonomous Intelligent Systems*, vol. 2, no. 1, p. 22, Aug. 2022, doi: 10.1007/s43684-022-00037-z.

[18] M. H. Asad, S. Khaliq, M. H. Yousaf, M. O. Ullah, and A. Ahmad, "Pothole detection using deep learning: a real-time and AI-on-the-edge perspective," *Advances in Civil Engineering*, vol. 2022, no. 1, Jan. 2022, doi: 10.1155/2022/9221211.

[19] K. Gayathri and S. Thangavelu, "Novel deep learning model for vehicle and pothole detection," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 23, no. 3, pp. 1576–1582, Sep. 2021, doi: 10.11591/ijeecs.v23.i3.pp1576-1582.

[20] C. K. Chiu, J. C. Liu, Y. W. Chan, and C. T. Yang, "Pavement distress detection using YOLO and faster RCNN on edge devices," in *Lecture Notes in Electrical Engineering*, vol. 1134, 2024, pp. 246–252.

[21] M. Omar and P. Kumar, "PD-ITS: pothole detection using YOLO variants for intelligent transport system," *SN Computer Science*, vol. 5, no. 5, p. 552, May 2024, doi: 10.1007/s42979-024-02887-1.

[22] C. C. Hsieh, H. W. Jia, W. H. Huang, and M. H. Hsih, "Deep learning-based road pavement inspection by integrating visual information and IMU," *Information (Switzerland)*, vol. 15, no. 4, p. 239, Apr. 2024, doi: 10.3390/info15040239.

[23] M. Divya, G. Divyashree, and B. U. Maheswari, "Pothole detection using YOLOv8 and CNN," in *2024 3rd International Conference for Innovation in Technology, INOCON 2024*, Mar. 2024, pp. 1–6, doi: 10.1109/INOCON60754.2024.10512004.

[24] Y. Matouq, D. Manasreh, and M. D. Nazzal, "AI-driven approach for automated real-time pothole detection, localization, and area estimation," *Transportation Research Record*, May 2024, doi: 10.1177/03611981241246993.

[25] J. Zeng and H. Zhong, "YOLOv8-PD: an improved road damage detection algorithm based on YOLOv8n model," *Scientific Reports*, vol. 14, no. 1, p. 12052, May 2024, doi: 10.1038/s41598-024-62933-z.

[26] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Smartphone-based real-time object recognition architecture for portable and constrained systems," *Journal of Real-Time Image Processing*, vol. 19, no. 1, pp. 103–115, Feb. 2022, doi: 10.1007/s11554-021-01164-1.

[27] D. Dlužnevskij, P. Stefanovič, and S. Ramanauskaitė, "Investigation of YOLOv5 efficiency in iPhone supported systems," *Baltic Journal of Modern Computing*, vol. 9, no. 3, pp. 333–344, 2021, doi: 10.22364/bjmc.2021.9.3.07.

[28] A. Fakhrou, J. Kunhoth, and S. Al Maadeed, "Smartphone-based food recognition system using multiple deep CNN models," *Multimedia Tools and Applications*, vol. 80, no. 21–23, pp. 33011–33032, Sep. 2021, doi: 10.1007/s11042-021-11329-6.

[29] M. A. Genaev *et al.*, "Classification of fruit flies by gender in images using smartphones and the YOLOv4-tiny neural network," *Mathematics*, vol. 10, no. 3, p. 295, Jan. 2022, doi: 10.3390/math10030295.

[30] S. Khalid, H. M. Oqaibi, M. Aqib, and Y. Hafeez, "Small pests detection in field crops using deep learning object detection," *Sustainability (Switzerland)*, vol. 15, no. 8, p. 6815, Apr. 2023, doi: 10.3390/su15086815.

[31] E. Assunção *et al.*, "Real-time image detection for edge devices: a peach fruit detection application," *Future Internet*, vol. 14, no. 11, p. 323, Nov. 2022, doi: 10.3390/fi14110323.
[32] R. Li, Z. Ji, S. Hu, X. Huang, J. Yang, and W. Li, "Tomato maturity recognition model based on improved YOLOv5 in Greenhouse," *Agronomy*, vol. 13, no. 2, p. 603, Feb. 2023, doi: 10.3390/agronomy13020603.
[33] S. Kong, J. Li, Y. Zhai, Z. Gao, Y. Zhou, and Y. Xu, "Real-time detection of crops with dense planting using deep learning at seedling stage," *Agronomy*, vol. 13, no. 6, p. 1503, May 2023, doi: 10.3390/agronomy13061503.
[34] Y. Liu *et al.*, "An Improved Tuna-YOLO Model Based on YOLO v3 for real-time tuna detection considering lightweight deployment," *Journal of Marine Science and Engineering*, vol. 11, no. 3, p. 542, Mar. 2023, doi: 10.3390/jmse11030542.
[35] K. Zhao, L. Zhao, Y. Zhao, and H. Deng, "Study on lightweight model of maize seedling object detection based on YOLOv7," *Applied Sciences (Switzerland)*, vol. 13, no. 13, p. 7731, Jun. 2023, doi: 10.3390/app13137731.
[36] M. Coccia, "Sources of disruptive technologies for industrial change," *Industria*, vol. 38, no. 1, pp. 97–120, 2017.
[37] M. Coccia, "Sources of technological innovation: radical and incremental innovation problem-driven to support competitive advantage of firms," *Technology Analysis and Strategic Management*, vol. 29, no. 9, pp. 1048–1061, Oct. 2017, doi: 10.1080/09537325.2016.1268682.
[38] S. Nienaber, M. J. (Thinus) Booysen, and S. Kroon, "Dataset of images used for pothole detection," in *Proceedings of the 34th Southern African Transport Conference (SATC 2015)*, 2015, pp. 153–164.
[39] X. Jiang, A. Hadid, Y. Pang, E. Granger, and X. Feng, *Deep learning in object detection and recognition*. Singapore: Springer Singapore, 2019.
[40] M. Fromm, M. Schubert, G. Castilla, J. Linke, and G. McDermid, "Automated detection of conifer seedlings in drone imagery using convolutional neural networks," *Remote Sensing*, vol. 11, no. 21, p. 2585, Nov. 2019, doi: 10.3390/rs11212585.
[41] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 7464–7475, doi: 10.1109/cvpr52729.2023.00721.
[42] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," *github*, 2013. https://github.com/ultralytics/ultralytics.
[43] W. Y. Lee, S. M. Park, and K. B. Sim, "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm," *Optik*, vol. 172, pp. 359–367, Nov. 2018, doi: 10.1016/j.ijleo.2018.07.044.
[44] R. Sunkara and T. Luo, "No more strided convolutions or pooling: a new CNN building block for low-resolution images and small objects," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13715 LNAI, 2023, pp. 443–459.
[45] T. Hagos, "Android studio profiler," *Android Studio IDE Quick Reference*, pp. 73–82, 2019, doi: 10.1007/978-1-4842-4953-6_7.

BIOGRAPHIES OF AUTHORS

Achyar Ulul Amri is the advanced analytics lead at one of Indonesia's largest conglomerates. Holding a bachelor's degree in Mathematics from Universitas Gadjah Mada, he has extensive experience in retail and automotive analytics across multiple countries. Currently, he is pursuing a Master's degree in Computer Science at Bina Nusantara University, Indonesia, where he is part of the Binus Graduate Program under the guidance of Mr. Gede. His research interests focus on computer science, particularly in machine learning, deep learning, and computer vision. He can be contacted at email: achyar.amri@binus.ac.id.

Gede Putra Kusuma received Ph.D. degree in Electrical and Electronic Engineering from Nanyang Technological University (NTU), Singapore, in 2013. He is currently working as a lecturer and head of Department of Computer Science, BINUS Graduate Program, Bina Nusantara University, Indonesia. Before joining Bina Nusantara University, he was working as a research scientist in I2R–A*STAR, Singapore. His research interests include computer vision, pattern recognition, deep learning, face recognition, and appearance-based object recognition. He can be contacted at email: inegara@binus.edu.