Region-Specific Road Object Detection in African Environments: Dataset Curation, Benchmarking, and Edge Deployment

Rumbidzai Muzata
Department of Mechatronic Engineering, Pan African University Institute for Basic Sciences, Technology, and Innovation (PAUSTI), Juja, Kenya
rumbidzaimuzata@gmail.com (corresponding author)

Celestin Nkundineza
College of Science and Technology, University of Rwanda, Kigali, Rwanda
c.nkundineza@ur.ac.rw

James Kuria Kimotho
Department of Mechanical Engineering, Jomo Kenyatta University of Agriculture and Technology, Kenya
jkuria@eng.jkuat.ac.ke

Received: 30 December 2025 | Revised: 4 February 2026 and 16 February 2026 | Accepted: 25 February 2026

 DOI: https://doi.org/10.48084/etasr.17251

ABSTRACT

Road safety in Africa faces substantial challenges that current public and custom datasets for autonomous driving do not adequately address. Public datasets frequently underrepresent object classes crucial to road safety in the African region, which features unique obstacles such as wild and domestic animals, informal public vehicles, and infrastructure hazards. Without these classes, detection models often miss road obstacles, thereby reducing safety on African roads. This study introduces a custom dataset of 3,236 original images with annotations for 11 object classes. It includes wild animals (kudus, elephants), domestic animals (cattle, goats), informal public vehicles (boda bodas, tuk-tuks, minibus taxis), and infrastructure hazards (potholes, unmarked speed bumps). Three of the most recent You Only Look Once (YOLO) object detection models (versions 8, 9, and 10) were trained on our custom dataset and evaluated on an NVIDIA RTX 3050 GPU. Of all the models, YOLOv8 achieved the highest mean Average Precision (mAP)@0.5 of 94.2%, followed closely by YOLOv10 and YOLOv9 at 92.5% and 90.6%, respectively, demonstrating strong detection performance on challenging, geographically relevant data. To assess the models in real-world deployment, they were optimized with TensorRT and deployed on an NVIDIA Jetson Nano embedded platform. This optimization achieved a 77% reduction in inference time with a very small accuracy drop of 0.64%, proving that the models are capable of fast, accurate, real-time execution on inexpensive edge devices. This study addresses geographic bias in autonomous-driving datasets in Africa and offers deployment-ready solutions for inexpensive edge hardware.

Keywords-edge deployment; geographic bias; road obstacle detection;YOLO

I. INTRODUCTION

Computer vision and object detection enable the identification of objects in images and videos using bounding boxes [1] and are now widely applied across various fields, particularly in autonomous driving and Advanced Driver Assistance Systems (ADAS). Object detection plays a crucial role in identifying road obstacles, enabling ADAS and autonomous driving systems to enhance road safety. Recent advances in high-accuracy object detection models have further enhanced the capabilities of autonomous driving and ADAS, contributing to the development of Intelligent Transport Systems (ITS) and improved road safety [2].

Object detection frameworks are typically categorized into two groups: two-stage detectors and single-stage detectors [3]. Region-based Convolutional Neural Networks (R-CNNs) [4], Fast R-CNNs [3], Faster R-CNNs [5], and the Deformable Part Model (DPM) generate regional proposals first, followed by a second stage object classification; hence, they are two-stage detectors. This approach produces high accuracy at the expense of inference speed, limiting the deployment of two-stage

detectors for real-time applications such as road obstacle detection, where speed is a critical factor.

Single-stage detectors, such as the You Only Look Once (YOLO) [1] and Single Shot Detector (SSD) [6], perform localization and classification in a single pass, making them faster and more efficient than two-stage detectors [7]. By treating object detection as a single regression problem, the initial YOLO model transformed the field and paved the way for major advancements in subsequent iterations [1, 8]. Early improvements emphasized scale-aware detection and localization accuracy via multi-scale prediction [9] and anchor boxes [10], alongside the integration of Cross Stage Partial Darknet (CSPDarknet) [11] for improved feature extraction. More recently, PyTorch-based Cross Stage Partial Network (CSPNet) and Path Aggregation Network (PANet) [12] were adopted to further improve efficiency and adaptability in dynamic environments.

Recent YOLO family iterations, including YOLOv8 [13], YOLOv9 [14], and YOLOv10 [15], have further improved road obstacle detection. YOLOv8 introduces an anchor-free, decoupled detection head in conjunction with the C2f module, enhancing feature fusion and detection performance. YOLOv9 incorporates the Generalized Efficient Layer Aggregation Network (GELAN) and Programmable Gradient Information (PGI) to improve training stability and model resource efficiency. YOLOv10 applies an upgraded CSPNet and one-to-many heads to further improve multi-scale detection. With each version, YOLO-based models have improved their ability to recognize diverse road components, from static obstacles [16] and surface hazards like potholes [17] to moving road users such as pedestrians [18].

However, detection accuracy remains susceptible to limitations in the quality and balance of the training data, especially in unstructured driving scenarios. Authors in [19] reported that training strategies and offline data augmentation can significantly improve the detection and inference performance of YOLO models. This enables the YOLO family to meet the stringent computational and latency requirements for real-time deployment in complex, dynamic road environments typical of African road scenarios. Typical African road environments are often characterized by mixed traffic, where animals, informal public transport vehicles, and unregulated hazards are frequently present [20]. In these scenarios, fast and adaptable single-stage detectors like YOLO are the preferred choice.

While these improvements highlight the suitability of YOLO-based models for real deployment, their performance and reliability remain constrained by a critical challenge: the geographic bias inherent in existing training datasets [21]. Widely adopted benchmarks such as Microsoft Common Objects in Context (MS COCO) [22], Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [23], Berkeley DeepDrive 100K (BDD100K) [24], PASCAL Visual Object Classes (PASCAL VOC) [25], and Cityscapes [26] have driven remarkable progress in object detection. However, they predominantly represent structured road environments and conventional obstacle types characteristic of developed regions in North America, Europe, and East Asia. Moreover, 53% of African roads are unpaved [27], and traffic is often chaotic, with people frequently sharing space with vehicles and animals [28].

Many studies have addressed this issue by developing custom datasets [19, 29, 30]. For example, authors in [31] created the Drive India dataset, which presents objects unique to Indian roads. It includes 24 object classes collected from city streets, rural roads, and construction areas, achieving a mean Average Precision (mAP) of 0.78 with YOLOv8. Similarly, authors in [32] developed a South Asia dataset capturing 13 vehicle classes annotated by wheel number, driving force, and size to reduce the number of classes. Their dataset was trained on YOLOv5 and YOLOv8, achieving an mAP of 0.6 and 0.7, respectively. Authors in [33] created a dataset of three object classes (stone, pothole, and fallen trees), trained on an improved YOLOv5, achieving an mAP of 0.78. In [17], a custom pothole detection dataset for South African roads was developed, covering diverse lighting and weather conditions across the Eastern and Western Cape provinces. This dataset comprises 1,910 images annotated with bounding boxes and segmentation masks, supporting real-time deployment on edge devices using SSD and YOLOv3. This resulted in a strong, context-aware detection system that can be used in regional road maintenance and driver-assistance systems.

However, most existing datasets, both public and custom, do not capture several critical object classes unique to African roads [19, 20, 34]. The absence of region-specific objects reduces the generalizability of object detection models, making it difficult to address the unique challenges of African road scenarios. Although authors in [19] focused on the detecting animals on African roads and achieved an mAP of 94.68%, there is still a lack of comprehensive datasets that include diverse vehicle types, animals, and infrastructure hazards.

This study addresses this gap by developing a region-specific custom dataset that integrates these object classes. It further compares the performance of three YOLO-based models: YOLOv8, YOLOv9, and YOLOv10. Our benchmarking uses an NVIDIA RTX 3050 GPU and an NVIDIA Jetson Nano to measure key metrics, such as precision, recall, and mAP. This analysis identifies the best-performing model, particularly for low-cost edge hardware, and provides insights into the practical deployment of these models in resource-constrained African contexts.

The key contributions of this study are:

- Development and validation of a custom dataset explicitly tailored for African road scenarios, including rare wildlife and livestock road obstacles.
- Comparison of YOLOv8, YOLOv9, and YOLOv10 using the custom African road obstacle dataset.
- Systematic optimization for edge deployment, achieving 4.4× speedup through TensorRT and demonstrating real-time feasibility on the NVIDIA Jetson Nano.

II. METHODOLOGY

A. Data Collection and Annotation

Images and videos related to common obstacles encountered on roads in the African region were collected from various trusted online sources, such as Pexels [35] and Unsplash [36]. Some images were manually collected using smartphone cameras. In total, 1,500 images were extracted from Pexels and Unsplash using ImageEye [19], and 316 images were extracted from videos using a Python script. Images relevant to the African road context were collected by downloading 700 samples from the Open Image Dataset v6 (OIDv6) [37]. Original image collection was conducted along urban roads in Kenya and both rural and urban roads in Zimbabwe using a Google Pixel 6 Pro and an iPhone 13 Pro, resulting in 720 images. The images from Kenya were captured along Gatundu and Thika Road in Juja, whereas those from Zimbabwe were captured in Mvurwi, Mashonaland Central Province, and Harare, Mashonaland East Province. The total number of original images in the dataset was 3,236. These include traffic scenes from urban and rural areas in the African region, including unstructured road conditions, dense traffic, unmarked road features, wild animals, jaywalkers, street vendors, unpaved roads, and damaged road surfaces. The images in Figure 1 represent examples of the collected images.

Fig. 1. Sample images: (a) mixed traffic scene, (b) minibus, (c) wild animal and a car, (d) pothole, (e) unmarked speed bump, (f) person jaywalking.

All images were resized to a resolution of 608 × 608 pixels. This resolution was selected because it is compatible with the YOLOv8, YOLOv9, and YOLOv10 models, which accept input image resolutions that are multiples of 32 [38]. Higher-resolution images are more suitable because they improve model accuracy; however, they require more computational resources. The 608 × 608 resolution provides a better balance between accuracy and available hardware.

The resized images were uploaded and annotated in the Label Studio annotation tool using the bounding box labeling technique. That is, regular bounding boxes were drawn to mark the boundaries of each object class in the images. During the annotation, 11 object classes were considered to reflect the common African road obstacles. These are: 'person' (including jaywalkers and street vendors), 'car,' 'wild animals' (kudus and elephants), '2-wheeled vehicles' (boda bodas and bicycles used for transportation), 'bus,' 'minibus (used as taxis),' 'domestic animals' (sheep, donkeys, horses, goats, cattle, dogs), 'speed bumps,' 'potholes,' 'tricycle (tuk tuk),' and 'truck' (both large and small).

This level of granularity demonstrates the lack of comprehensive, region-specific object classes in the majority of publicly available datasets. The inclusion of object classes such as street vendors, wild animals, informal public vehicles, and typical road hazards enables the trained models to better generalize and navigate the complex dynamics of African roads when trained on this dataset. Following annotation of all images, the dataset was exported to YOLO format (TXT) using normalized coordinates. This format was selected because it is compatible with the YOLO-based object detection models used for training and validation in this study.

B. Dataset Statistics and Splitting

The dataset contains 3,236 original images, each annotated, resulting in a total of 8,191 labels. Figure 2 shows the distribution of labels per class across the 11 object classes. The dataset was split into three subsets: training, validation, and testing. To ensure a robust and unbiased evaluation of model performance, the dataset was split using a 14:3:3 ratio [39]. This distribution allocates 70% of the data to training, allowing the model to learn from a diverse set of examples and reducing the risk of overfitting. The remaining 30% of the data was divided equally between validation and testing, with 15% allocated to each subset, enabling unbiased evaluation of the trained models.

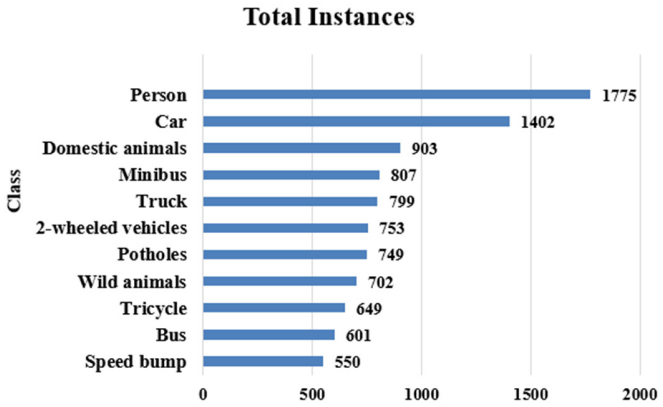


Fig. 2. Distribution of total annotated instances across all classes in the dataset.

Detecting or classifying road obstacles is difficult due to changing lighting conditions, partial or complete occlusions, and varying viewing angles [40]. To correct geometric distortion, we employed various techniques, including random scaling and cropping. To correct photometric distortion, the brightness and noise of the images were adjusted. This method not only improves the performance of the model, but also improves its ability to handle complex and unpredictable conditions in real-world scenarios [41].

1) Geometric Augmentation: Random Crop

Images were cropped with a minimum zoom of 0%, where the original image remains unchanged, and a maximum zoom of 27%, where approximately 27% of the image is removed from each side. This cropping approach is particularly useful in real-world scenarios, where road obstacles are unlikely to always appear in the same position, size, or shape. Cropping 27% from each side of the images helps the model handle such variations more effectively [42].

2) Photometric Augmentation

a) Brightness

We modified the brightness of the images in the dataset within a range of −15% to +15%. This allows the model to handle varying brightness levels that represent different weather conditions.

b) Exposure

In this study, the exposure of the dataset was adjusted within a range of −13% to +13%. This is important for enhancing the model's robustness when handling different exposure levels that occur under various lighting conditions.

c) Salt and Pepper Noise

We deliberately modified the pixel values of the images, producing black-and-white spots in random regions, affecting 1.5% of the pixels in each image.

d) Light Rain Overlay

The alpha blending technique was applied to simulate rainfall drops. Each image was transformed according to (1):

$$O = (1 - \alpha)I' + \alpha\text{Rain}, \quad \alpha = 0.25 \tag{1}$$

After applying the described geometric and photometric augmentation techniques, the dataset size increased from 3,236 original images to 15,548 images. This increase enhanced data diversity and reduced the risk of overfitting during the training process of the models.

C. Model Training

The YOLO models, namely YOLOv8n, YOLOv9t, and YOLOv10n, were selected based on their suitability for real-time detection under constrained hardware. The three models were originally developed using the PyTorch framework. Therefore, this framework was used to train the models on the custom dataset with an NVIDIA GeForce RTX 3050 GPU. Hyperparameters were carefully configured to strike a good balance between the models' performance and the available resources, given the limited training resources.

A batch size of 16 was chosen to meet the memory and computational requirements of the available training hardware while maintaining training stability. The AdamW optimizer was used during training, with a learning rate of 0.001 and a momentum of 0.9, which helps limit excessive oscillations. A weight decay of 0.0005 was selected because it helps prevent overfitting by adding L2 regularization to the loss function.

To make the models converge faster by providing a common set of visual features, pre-trained models on the COCO dataset were downloaded from Ultralytics [43]. The training was performed for 100 epochs on both the augmented and original custom datasets. Training the models on the same custom dataset and using the same framework, as well as the same hyperparameter configurations, ensured a fair comparison among the models.

D. Model Testing

Model testing was performed to ensure that the models YOLOv8, YOLOv9, and YOLOv10, trained on a custom dataset of African roads, are capable of detecting objects of interest accurately. The models were tested on both the dataset test set and a 5 min video filmed along Thika Road in Nairobi, Kenya.

E. Model Evaluation

The model was evaluated using three metrics: precision, recall and mAP. Precision measures how many of the detected road obstacles were correctly identified. Recall indicates the model's capacity to identify all road obstacles in the dataset. Finally, the mAP metric assesses the trained model's performance based on classifying detected objects into three categories: false positives (FP), false negatives (FN), and true positives (TP) [44]. The formulas for these metrics are given in (2) through (5):

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{3}$$

$$\text{AP} = \int_0^1 \text{P(R)dR} \tag{4}$$

$$\text{mAP} = \frac{1}{\text{n}} \sum_{\text{i=1}}^{\text{n}} \text{AP(i)} \tag{5}$$

where n is the total number of categories, and AP(i) denotes the average precision of the i-th category.

F. Deployment on Jetson Nano

The NVIDIA Jetson Nano is a compact, powerful edge computer with parallel capabilities suitable for deep learning models. The Jetson Nano was configured using JetPack 4.6.1, which includes Ubuntu 20.04 as the supported operating system. Software libraries such as Ultralytics, TensorRT 8.6.0, CUDA 10.2, cuDNN 8.2.1, OpenCV 4.8.0 with CUDA, ONNX 1.17.0, and PyTorch 1.13 with GPU were installed to run deep learning models on the Jetson Nano.

The ONNX package was used to export the three trained PyTorch models to ONNX, while maintaining an input resolution of 608 × 608 pixels. The converted ONNX models

were individually evaluated for mAP@0.5 and inference speed, using the ONNX Runtime engine in PyTorch.

The ONNX models were carefully optimized with TensorRT engine formats while keeping the input resolution at 608 × 608 to balance hardware constraints and small-to-medium object detection. This optimization used TensorRT's tensor and fusion optimization capabilities, including pruning unnecessary nodes in the input-to-output path and fusing layers more prominently. To accelerate model inference speed on NVIDIA Jetson Nano, further optimization was performed using half-precision (FP16), a TensorRT precision calibration technique. The half-precision optimized models were reevaluated for mAP on the validation set of the custom dataset, and their results were recorded alongside inference speed.

III. RESULTS AND DISCUSSION

A. Test Results on NVIDIA RTX 3050 GPU

The models successfully detected objects in the images drawing bounding boxes around predicted objects and assigning a confidence score to each detection, as shown in Figure 3. The confidence score quantifies the model's certainty for every identified object. Higher confidence scores indicate that the model detected objects with greater certainty.

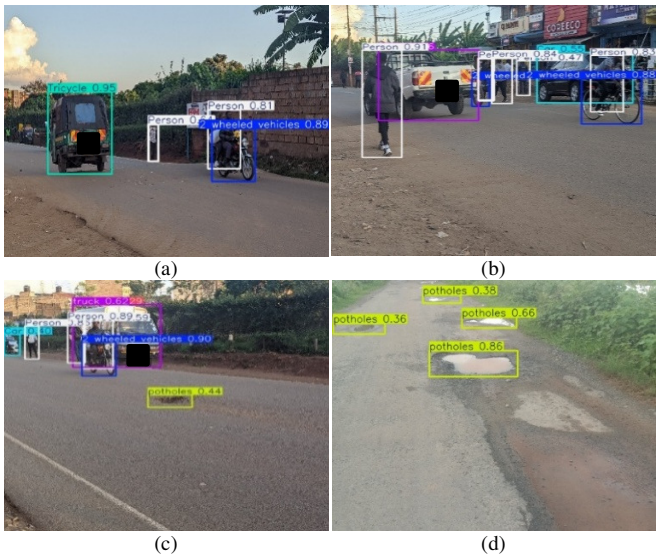


Fig. 3. Testing results on NVIDIA RTX 3050 GPU using YOLO models: (a) YOLOv8 detection from live traffic video, (b) YOLOv10 detection from live traffic video, (c) YOLOv8 detection from test set, (d) YOLOv10 detection from test set.

B. Evaluation Results on NVIDIA RTX 3050 GPU

Results in Table I show that YOLOv8 achieved the highest precision (0.823), resulting in the fewest false road obstacle detections. YOLOv10 and YOLOv9 followed closely, with scores of 0.822 and 0.787, respectively, indicating that they also produce relatively few false positives. YOLOv8 again leads in recall (0.718), meaning it misses the fewest objects. YOLOv10 and YOLOv9 follow, with recall values of 0.704 and 0.690, respectively. While all models perform equally, YOLOv9's somewhat lower recall implies it is more likely to miss some objects. YOLOv8 achieved an mAP of 0.942, demonstrating superior performance compared to YOLOv9 and YOLOv10. YOLOv10 and YOLOv9 also performed well, achieving an mAP of 0.925 and 0.906, respectively. Recent studies [30, 45, 46] on comparative analysis of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 on road obstacles align with these findings.

TABLE I. YOLO MODEL PERFORMANCE BEFORE AUGMENTATION

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Time (ms)	FPS
YOLOv8	0.823	0.718	0.798	0.593	10.8	93
YOLOv9	0.787	0.690	0.763	0.566	23.0	44
YOLOv10	0.822	0.704	0.783	0.589	12.8	78

When evaluated on inference speed, YOLOv8 outperformed YOLOv9 and YOLOv10, achieving an inference speed of 93 FPS. YOLOv10 closely followed, with an inference speed of 78 FPS. YOLOv9 exhibited the slowest inference speed of 45 FPS. Despite having the fewest parameters, YOLOv9's use of GELAN and PGI modules introduces dynamic computations that may impact both accuracy and speed, making it nearly half as fast as YOLOv8. In their study, authors in [18] compared YOLOv8 and YOLOv9 for detecting people in thermal images, and found that YOLOv9 is twice as slow, with YOLOv8 achieving approximately 29 FPS and YOLOv9 achieving 14 FPS. The experimental results suggest that despite the advancements made in the architecture of the latest models, YOLOv8 remains competitive, offering a superior balance in speed and accuracy.

C. Evaluation Results on NVIDIA RTX 3050 GPU After Augmentation

The results in Table II indicate that all three YOLO models benefit significantly from data augmentation, with consistent improvements in precision, recall, and mAP for different Intersection over Union (IoU) thresholds. Augmentation introduced artificial variations that enabled the models to learn more effectively, leading to better generalization and less overfitting. YOLOv8 achieved the best precision (0.938), implying it produces the fewest false positives, whereas YOLOv10 and YOLOv9 also showed good precision (0.926 and 0.931, respectively). Recall improved as well, with YOLOv8 leading at 0.868, followed by YOLOv10 (0.860) and YOLOv9 (0.846), indicating enhanced detection of all relevant obstacles in the dataset.

TABLE II. YOLO MODEL PERFORMANCE AFTER AUGMENTATION

Model	Precision	Recall	mAP@0.5	mAP@0.5:0.95	Time (ms)	FPS
YOLOv8	0.938	0.868	0.942	0.752	10.8	93
YOLOv9	0.931	0.846	0.906	0.746	23.0	44
YOLOv10	0.926	0.860	0.925	0.755	12.8	78

D. Per-Class Analysis

Table III presents the per-class performance of YOLOv8, YOLOv9, and YOLOv10. All models achieved strong

detection results, with AP@0.5 values above 0.78 for all classes. YOLOv8 demonstrated the most consistent and highest performance across most classes, particularly for critical and commonly occurring classes.

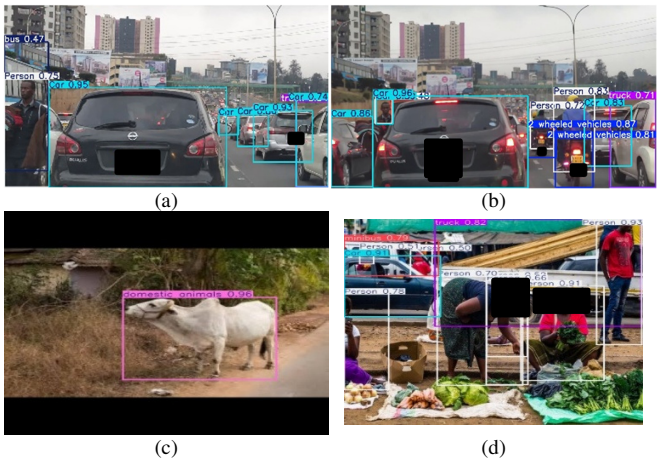
TABLE III. PER-CLASS PERFORMANCE COMPARISON (AP@0.5)

Class	YOLOv8	YOLOv9	YOLOv10	Best model
2-wheeled vehicles	0.927	0.926	0.927	v8 / v10
Car	0.918	0.861	0.864	v8
Person	0.873	0.789	0.795	v8
Tricycle	0.985	0.972	0.940	v8
Bus	0.988	0.937	0.979	v8
Minibus	0.970	0.922	0.946	v8
Domestic animals	0.906	0.871	0.895	v8
Speed bumps	0.995	0.995	0.995	All
Potholes	0.850	0.804	0.818	v8
Wild animals	0.987	0.988	0.976	v9
Truck	0.971	0.899	0.917	v8

All models showed slightly lower AP values for the pothole and person classes. This is likely due to high variability in appearance, size, and shape. Potholes vary in geometry, whereas people appear in diverse poses and scales. Additionally, both potholes and persons are often small in images, making it more difficult for the models to extract distinguishable features for accurate localization. These challenges are significant for lightweight models, which have limited capacity to capture such variations.

E. Evaluation Results on NVIDIA Jetson Nano

Figure 4 shows selected qualitative results of object detection on the NVIDIA Jetson Nano. The detections include frames from real-time traffic videos and the test set of the custom African road dataset. The models accurately localized objects using bounding boxes, assigned correct class labels, and achieved high confidence scores at an IoU threshold of 0.5.



(a) (b) (c) (d)

Fig. 4. Testing results on Jetson Nano edge device using YOLO models: (a) YOLOv8 detection from live traffic video, (b) YOLOv10 detection from live traffic video, (c) YOLOv8 detection from test set, (d) YOLOv10 detection from test set.

F. Performance Benchmarking of YOLOv8, YOLOv9, and YOLOv10 on Jetson Nano

The evaluation results in Table IV indicate that YOLOv8 provides the best detection accuracy in its PyTorch FP32 version, achieving 0.942 mAP@0.5, with only a small drop (0.6%) when optimizing it with ONNX and TensorRT FP16 formats. YOLOv9 shows similar behavior, achieving 0.906 mAP@0.5 in PyTorch, with a slight reduction to 0.899 after TensorRT half-precision quantization. YOLOv10 achieves 0.908 mAP@0.5 under TensorRT FP16.

TABLE IV. COMPREHENSIVE PERFORMANCE ANALYSIS ON JETSON NANO EDGE DEVICE

Model	Framework	mAP	Time (ms)	FPS	Loss (%)
YOLOv8	PyTorch	0.942	189.2	5.29	–
	ONNX	0.936	118.0	8.47	0.64
	TensorRT FP16	0.936	43.0	23.26	0.64
YOLOv9	PyTorch	0.906	262.0	3.82	–
	ONNX	0.900	148.9	6.72	0.66
	TensorRT FP16	0.899	62.0	16.13	0.77
YOLOv10	PyTorch	0.914	172.2	5.81	–
	ONNX	0.908	114.8	8.71	0.66
	TensorRT FP16	0.908	40.5	24.69	0.66

While YOLOv10's mAP is slightly lower than YOLOv8's, it offers substantially faster inference, reaching 24.69 FPS with a latency of 40.5 ms, making it the fastest and most efficient model in terms of speed under FP16. Across all models, the TensorRT inference engine provides a 4.4× speedup compared to standard PyTorch. This acceleration is achieved through optimizations such as half-precision support, kernel auto-tuning, and layer fusion, which enhance inference performance on the Jetson Nano [47]. Additionally, using the ONNX format also improves inference speed, making it a viable alternative for deployment.

G. Comparison with Previous Studies

Table V compares the models evaluated in this study with previously published studies on custom road obstacle datasets. The experimental results show that state-of-the-art YOLOv8–YOLOv10 models achieve accuracy comparable to recent YOLO-based road obstacle detection methods, while maintaining inference times within the real-time range. However, it is important to note that inference speed is highly dependent on the underlying hardware and optimization settings used during evaluation.

TABLE V. COMPARISON OF YOLO MODELS ON ROAD OBSTACLE DETECTION WITH PREVIOUS STUDIES

Study	Model	mAP@0.5	Inference time (ms)
[48]	YOLOv5	0.94	1.3
	YOLOv7	0.441	4.93
	YOLOv8	0.927	3.3
[32]	YOLOv5	0.60	Not indicated
	YOLOv8	0.70	Not indicated
[33]	YOLOv5	0.78	Not indicated
This study	YOLOv8	0.942	10.8
	YOLOv9	0.906	23.0
	YOLOv10	0.914	12.8

IV. CONCLUSION

This study presented a region-specific dataset for road obstacle detection that addresses the complexity of traffic scenes in African driving conditions. The custom dataset comprises 3,236 original images annotated across 11 object classes representing typical African road scenarios, including informal public vehicles, livestock, wildlife, potholes, and unmarked speed bumps. Emphasizing these underrepresented obstacle types helps mitigate geographic bias, which often limits the deployment of mainstream object detection models in Africa.

Experimental results demonstrated that the lightweight You Only Look Once v8 (YOLOv8) model provides the best trade-off between accuracy and speed on the NVIDIA RTX 3050 GPU, achieving 93 FPS with a mean Average Precision (mAP)@0.5 of 0.925. Per-class analysis showed that YOLOv8 consistently outperformed other models across most object classes, while YOLOv9 and YOLOv10 remained competitive, particularly for objects with complex or irregular shapes, such as potholes.

The feasibility of deploying these models on resource-constrained hardware was evaluated using an NVIDIA Jetson Nano edge device. Conversion of PyTorch models to ONNX and TensorRT FP16 formats resulted in a 77% reduction in inference time with less than 0.8% accuracy drop. The fastest inference speed was achieved by YOLOv10 (24.69 FPS), whereas YOLOv8 maintained the highest accuracy after optimization. These findings represent a substantial initial step toward robust, low-latency obstacle detection on cost-effective embedded hardware, making the proposed models suitable for Advanced Driver Assistance Systems (ADAS) and other safety-critical applications in African scenarios.

Future work will extend this study by incorporating video sequences to enable behavioral analysis and driving pattern recognition, and by generating behavioral annotations to improve model generalizability. Additionally, optimization techniques, such as knowledge distillation, will be investigated to further improve model efficiency and performance.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Figshare at https://doi.org/10.6084/m9.figshare.31241572.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 779–788, https://doi.org/10.1109/CVPR.2016.91.

[2] M. Katari, G. Krishnamoorthy, L. Shanmugam, and A. Tadimarri, "Driving Towards Safety: The Role of ECUs and IMUs in Advanced Driver-Assistance Systems (ADAS)," *IJFMR - International Journal For Multidisciplinary Research*, vol. 6, no. 2, Apr. 2024, Art. no. IJFMR240217022, https://doi.org/10.36948/ijfmr.2024.v06i02.17022.

[3] S. A. Alhashmi and A. Al-azawi, "A Review of the Single-Stage vs. Two-Stage Detectors Algorithm:Comprehensive Insights into Object Detection," *International Journal of Environmental Sciences*, vol. 11, no. 3s, pp. 775–787, May 2025.

[4] M. Othmani, "A vehicle detection and tracking method for traffic video based on faster R-CNN," *Multimedia Tools and Applications*, vol. 81, no. 20, pp. 28347–28365, Aug. 2022, https://doi.org/10.1007/s11042-022-12715-4.

[5] A. L. Dewi *et al.*, "Two-Stage Object Detection for Autonomous Vehicles With VGG-16 Based Faster R-CNN," *Jurnal Elektronika dan Telekomunikasi*, vol. 24, no. 1, pp. 25–30, Aug. 2024, https://doi.org/10.55981/jet.551.

[6] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *14th European Conference on Computer Vision*, Amsterdam, Netherlands, 2016, pp. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2.

[7] Y. Han, "Comparative Analysis of Two-Stage and One-Stage Object Detection Models," in *Proceedings of the 2nd International Conference on Data Analysis and Machine Learning*, Kuala Lumpur, Malaysia, 2024, pp. 289–294, https://doi.org/10.5220/0013515900004619.

[8] J. Wang *et al.*, "Research on Improved YOLOv5 for Low-Light Environment Object Detection," *Electronics*, vol. 12, no. 14, July 2023, Art. no. 3089, https://doi.org/10.3390/electronics12143089.

[9] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017, pp. 6517–6525, https://doi.org/10.1109/CVPR.2017.690.

[10] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv, Apr. 23, 2020, https://doi.org/10.48550/arXiv.2004.10934.

[11] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement." arXiv, Apr. 08, 2018, https://doi.org/10.48550/arXiv.1804.02767.

[12] M. Jani, J. Fayyad, Y. Al-Younes, and H. Najjaran, "Model Compression Methods for YOLOv5: A Review." arXiv, July 21, 2023, https://doi.org/10.48550/arXiv.2307.11904.

[13] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, https://doi.org/10.3390/make5040083.

[14] C.-Y. Wang, I.-H. Yeh, and H.-Y. Mark Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," in *18th European Conference on Computer Vision*, Milan, Italy, 2024, pp. 1–21, https://doi.org/10.1007/978-3-031-72751-1_1.

[15] A. Wang *et al.*, "YOLOv10: real-time end-to-end object detection," in *Proceedings of the 38th International Conference on Neural Information Processing Systems*, Vancouver, Canada, 2024, pp. 107984–108011.

[16] S. Rahman, J. H. Rony, J. Uddin, and M. A. Samad, "Real-Time Obstacle Detection with YOLOv8 in a WSN Using UAV Aerial Photography," *Journal of Imaging*, vol. 9, no. 10, Oct. 2023, Art. no. 216, https://doi.org/10.3390/jimaging9100216.

[17] K. Gajjar, T. van Niekerk, T. Wilm, and P. Mercorelli, "Vision-Based Deep Learning Algorithm for Detecting Potholes," *Journal of Physics: Conference Series*, vol. 2162, no. 1, Jan. 2022, Art. no. 012019, https://doi.org/10.1088/1742-6596/2162/1/012019.

[18] V. Breive and T. Sledevic, "Person Detection in Thermal Images: A Comparative Analysis of YOLOv8 and YOLOv9 Models," in *2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*, Vilnius, Lithuania, 2024, pp. 1–4, https://doi.org/10.1109/eStream61684.2024.10542600.

[19] P. Mutabarura, N. Muchuka, and D. Segera, "Comparative Evaluation of YOLO Models on an African Road Obstacles Dataset for Real-Time Obstacle Detection," *Engineering, Technology & Applied Science Research*, vol. 15, no. 1, pp. 19045–19051, Feb. 2025, https://doi.org/10.48084/etasr.9135.

[20] K. Jadaan, E. Al-Braizat, S. Al-Rafayah, H. Gammoh, and Y. Abukahlil, "Traffic Safety in Developed and Developing Countries: A Comparative Analysis," *Journal of Traffic and Logistics Engineering*, vol. 6, no. 1, pp. 1–5, June 2018, https://doi.org/10.18178/jtle.6.1.1-5.

[21] B. Wilson, J. Hoffman, and J. Morgenstern, "Predictive Inequity in Object Detection." arXiv, Feb. 21, 2019, https://doi.org/10.48550/arXiv.1902.11097.

[22] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context," in *13th European Conference on Computer Vision*, Zurich, Switzerland, 2014, pp. 740–755, https://doi.org/10.1007/978-3-319-10602-1_48.

[23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, 2012, pp. 3354–3361, https://doi.org/10.1109/CVPR.2012.6248074.

[24] Y. Jiang *et al.*, "EnlightenGAN: Deep Light Enhancement Without Paired Supervision," *IEEE Transactions on Image Processing*, vol. 30, pp. 2340–2349, 2021, https://doi.org/10.1109/TIP.2021.3051462.

[25] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015, https://doi.org/10.1007/s11263-014-0733-5.

[26] A. Song and Y. Kim, "Semantic Segmentation of Remote-Sensing Imagery Using Heterogeneous Big Data: International Society for Photogrammetry and Remote Sensing Potsdam and Cityscape Datasets," *ISPRS International Journal of Geo-Information*, vol. 9, no. 10, Oct. 2020, Art. no. 601, https://doi.org/10.3390/ijgi9100601.

[27] S. Randhawa, E. Aygün, G. Randhawa, B. Herfort, S. Lautenbach, and A. Zipf, "Paved or unpaved? A deep learning derived road surface global dataset from mapillary street-view imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 223, pp. 362–374, May 2025, https://doi.org/10.1016/j.isprsjprs.2025.02.020.

[28] A. Ziryawulawo, N. Mduma, M. Lyimo, A. Mbarebaki, R. Madanda, and A. Sam, "An Integrated Deep Learning-based Lane Departure Warning and Blind Spot Detection System: A Case Study for the Kayoola Buses," in *2023 First International Conference on the Advancements of Artificial Intelligence in African Context*, Arusha, Tanzania, 2023, pp. 1–8, https://doi.org/10.1109/AAIAC60008.2023.10465276.

[29] A. Musa, M. Hamada, and M. Hassan, "A Theoretical Framework Towards Building a Lightweight Model for Pothole Detection using Knowledge Distillation Approach," *SHS Web of Conferences*, vol. 139, May 2022, Art. no. 03002, https://doi.org/10.1051/shsconf/202213903002.

[30] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, "YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions." arXiv, Mar. 17, 2025, https://doi.org/10.48550/arXiv.2411.00201.

[31] R. Kumar, D. S. Reddy, and P. Rajalakshmi, "DriveIndia: An Object Detection Dataset for Diverse Indian Traffic Scenes." arXiv, Aug. 26, 2025, https://doi.org/10.48550/arXiv.2507.19912.

[32] M. N. Baig, R. Hajong, M. M. Patwary, M. S. Rahman, and H. A. Chowdhury, "BadODD: Bangladeshi Autonomous Driving Object Detection Dataset." arXiv, Jan. 19, 2024, https://doi.org/10.48550/arXiv.2401.10659.

[33] P. Tan, Z. Wang, and X. Chang, "Road Obstacle Detection Method Based on Improved YOLOv5," *Algorithms*, vol. 18, no. 6, May 2025, Art. no. 300, https://doi.org/10.3390/a18060300.

[34] M. E. Irhebhude, M. T. Ahmed, I. A. Ibrahim, and I. Shuaibu, "Real-Time YOLOv5-Based Object Detection for Autonomous Vehicles on Nigerian Roads," *International Journal of Recent Advances in Multidisciplinary Topics*, vol. 6, no. 12, pp. 19–25, Dec. 2025, https://doi.org/10.65138/ijramt.2025.v6i12.3168.

[35] "Photo by Bobby Randu on Pexels." Pexels. https://www.pexels.com/photo/man-riding-bicycle-on-street-20818748/.

[36] "Photo by Henning Borgersen on Unsplash." Unsplash. https://unsplash.com/photos/brown-deer-crossing-on-pathway-dP5tzqn2Jjg.

[37] "Open Images Dataset V6." Googleapis, [Online]. Available: https://storage.googleapis.com/openimages/web/factsfigures.html.

[38] E. C. Nnadozie, P. Casaseca-de-la-Higuera, O. Iloanusi, O. Ani, and C. Alberola-López, "Simplifying YOLOv5 for deployment in a real crop monitoring setting," *Multimedia Tools and Applications*, vol. 83, no. 17, pp. 50197–50223, May 2024, https://doi.org/10.1007/s11042-023-17435-x.

[39] P. Gupta, N. Vadgaonkar, J. Nirmal, and N. Mehendale, "A hybrid CNN–ViT framework for skin disease classification via feature extraction and selection," *Neural Computing and Applications*, vol. 37, no. 32, pp. 27151–27177, Nov. 2025, https://doi.org/10.1007/s00521-025-11664-x.

[40] S. Sultana, B. Ahmed, M. Paul, M. R. Islam, and S. Ahmad, "Vision-Based Robust Lane Detection and Tracking in Challenging Conditions," *IEEE Access*, vol. 11, pp. 67938–67955, 2023, https://doi.org/10.1109/ACCESS.2023.3292128.

[41] N. Aboudeshish, D. Ignatov, and R. Timofte, "AugmentGest: Can Random Data Cropping Augmentation Boost Gesture Recognition Performance?" arXiv, June 08, 2025, https://doi.org/10.48550/arXiv.2506.07216.

[42] A. S. Geetha, M. A. R. Alif, M. Hussain, and P. Allen, "Comparative Analysis of YOLOv8 and YOLOv10 in Vehicle Detection: Performance Metrics and Model Efficacy," *Vehicles*, vol. 6, no. 3, pp. 1364–1382, Aug. 2024, https://doi.org/10.3390/vehicles6030065.

[43] G. Jocher, J. Qiu, and A. Chaurasia. "Ultralytics YOLO." GitHub. https://github.com/ultralytics/ultralytics.

[44] Z. Chen, Y. Fang, J. Yin, S. Lv, F. Sheikh Muhammad, and L. Liu, "A novel lightweight YOLOv8-PSS model for obstacle detection on the path of unmanned agricultural vehicles," *Frontiers in Plant Science*, vol. 15, Dec. 2024, Art. no. 1509746, https://doi.org/10.3389/fpls.2024.1509746.

[45] I. Katsamenis *et al.*, "DORIE: Dataset of Road Infrastructure Elements—A Benchmark of YOLO Architectures for Real-Time Patrol Vehicle Monitoring," *Sensors*, vol. 25, no. 21, Oct. 2025, Art. no. 6653, https://doi.org/10.3390/s25216653.

[46] S. A. Dhinakar Raj, A. Singh, T. S. Shekhawat, M. M. M, and K. Sivasankaran, "Collision Avoidance System using YOLO-based Object Detection and Distance Estimation," in *2025 3rd International Conference on Intelligent Data Communication Technologies and Internet of Things*, Bengaluru, India, 2025, pp. 178–184, https://doi.org/10.1109/IDCIOT64235.2025.10915167.

[47] I. J. Ratul, Y. Zhou, and K. Yang, "Accelerating Deep Learning Inference: A Comparative Analysis of Modern Acceleration Frameworks," *Electronics*, vol. 14, no. 15, July 2025, Art. no. 2977, https://doi.org/10.3390/electronics14152977.

[48] N. M. Alahdal, F. Abukhodair, L. H. Meftah, and A. Cherif, "Real-time Object Detection in Autonomous Vehicles with YOLO," *Procedia Computer Science*, vol. 246, pp. 2792–2801, Jan. 2024, https://doi.org/10.1016/j.procs.2024.09.392.