Comparative Evaluation of YOLO Models on an African Road Obstacles Dataset for Real-Time Obstacle Detection

Pison Mutabarura
Department of Electrical & Electronics Engineering, Pan African University Institute for Basic Sciences, Technology, and Innovation (PAUSTI) Juja, Kenya
mutapiso@gmail.com (corresponding author)

Nicasio Maguu Muchuka
Department of Electrical & Control Engineering, Egerton University, Nakuru, Kenya
nmuchuka@egerton.ac.ke

Davies Rene Segera
Department of Electrical and Information Engineering, University of Nairobi, Nairobi, Kenya
Davies.segera@uonbi.ac.ke

Received: 29 September 2024 | Revised: 19 October 2024 and 31 October 2024 | Accepted: 9 November 2024

 DOI: https://doi.org/10.48084/etasr.9135

ABSTRACT

Public datasets are used to train road obstacle detection models, but they lack diverse and rare object classes found on African roads, negatively impacting the performance of models trained on them. Although attempts have been made to create custom datasets to train road obstacle detection models, they lack the unique challenges posed by African wildlife and livestock commonly encountered on African roads. This leads to poor performance of road obstacle detection systems in the African context. This study presents a custom dataset with rare African object classes and compares the performance of three YOLO models on it using mean Average Precision (mAP). The images were collected from multiple sources to ensure a wide range of scenarios. Offline data augmentation was applied to increase dataset diversity and simulate real-world road scenarios. The models were trained and evaluated, with YOLOv5 demonstrating superiority over the other two models, with an object detection accuracy of 94.68% mAP at an Intersection over Union (IoU) threshold of 0.5 with data augmentation. Offline data augmentation significantly improved all models' object detection accuracy, especially for YOLOv3. The results reveal the effectiveness of the custom dataset and highlight the importance of data augmentation in improving object detection.

Keywords-African road obstacles; object detection; data augmentation; road obstacles; YOLOv3

I. INTRODUCTION

In computer vision and object detection, dataset preparation is critical, as the performance of object detection models depends on the quality and diversity of the dataset used to train them. This fundamental step is crucial for all object detection models, which are classified into one-stage and two-stage object detectors. Two-stage detectors include Regional-based Convolutional Neural Networks (R-CNN) [1], Fast R-CNN [2], Faster R-CNN [3], and Deformable Part Model (DPM) [4]. Examples of one-stage detectors are You Only Look Once (YOLO) [5] and Single Shot Multibox Detectors (SSDs) [6]. The YOLO series of object detectors has gained a lot of attention due to its balanced accuracy and object detection speed [7]. In [5], the original YOLO algorithm outperformed two-stage object detectors, such as the Fast R-CNN algorithm and R-CNN, in object detection speed and distinguishing objects from background pixels. The YOLO algorithm has a simple pipeline and uses a regression mechanism, making it a real-time object detector. The YOLO algorithm surpasses two-stage object detectors such as RCNN, Fast RCNN, DPM, and Faster RCNN for applications requiring real-time object detection [1, 5].

The original YOLO algorithm struggled in object localization and recorded a lower recall than regional proposal-based object detectors [5]. Thus, it was modified to improve accuracy while maintaining its high object detection speed, leading to other YOLO versions, such as YOLOv2 [8], YOLOv3 [9], and YOLOv4 [10]. These versions modified the original YOLO architecture, for example, including the Spatial Pyramid Pooling (SPP) layer and PANet path aggregation in the neck to enhance the extraction of important features from

the backbone [10, 11]. Since 2020, other versions of YOLO with further advances have been proposed to improve object detection accuracy. These include YOLOv5 [11] with advanced features, such as automatic adaptation of anchors and mosaic data enhancement [12, 13], YOLOv6, YOLOv7, YOLOv8, YOLOv9, and YOLOv10.

These advances in the YOLO algorithm have significant implications in various applications, including obstacle detection. In road obstacle detection for collision avoidance systems, publicly available datasets such as the PASCAL Visual Object Classes Challenge (PASCAL VOC) [14], Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [15], Common Objects in Context (COCO) [16], and ImageNet [17] have been used to train models [18, 19]. This is because creating a large and diverse dataset is costly and time-consuming [20]. Of these datasets, KITTI has been widely used to train object detection models used in autonomous driving research [21-23]. The images in the KITTI dataset were captured using a camera and LiDAR for object detection, tracking, odometry, and semantic segmentation applications. This dataset comprises a large number of images for benchmarking these applications. However, the data in the KITTI dataset were captured in good weather, which limits its effectiveness in adverse weather conditions. As a result, data augmentation techniques such as rotation, flipping, the addition of random noise, etc., have been proposed to artificially transform data, simulate these weather conditions, and enhance the robustness and generalizability of trained models [24].

Common transformation approaches for data augmentation in object detection include flipping, cropping, rotation, blurring, shifting, contrast, and brightness adjustments [25]. These data augmentation techniques can be applied to the dataset, either in offline or online data augmentation, before or during the model training. Although online data augmentation techniques have been integrated into the training pipelines of the most recent libraries, such as Ultralytics [26], OpenMMLab [27], and OpenVINO [28], they slow the training process and consume significant computational and extensive memory resources for execution [29]. On the other hand, offline data augmentation techniques increase data diversity with consistent data transformations across the dataset before training. This reduces the model's computational and memory burden on the training hardware. Therefore, offline data augmentation is valuable in increasing the volume of the data for training object detection models, addressing the significant challenges of overfitting and the limited amount of data due to the time required to collect large amounts of data [30]. According to [31], rotation and the Wasserstein Generative Adversarial Network (WGAN) have yielded better results than other data augmentation techniques. Adding noise to the images used to train an object detection model in the form of salt and pepper with white and black dots in an image helps prevent overfitting [32]. With random noise added to the training data, the object detection models can learn faster and more accurately. Positional biases in the images can be eliminated by moving the image along the x and y directions, left and right, up and down, helping object detectors search the entire image [30]. Although data augmentation has proven valuable in improving model performance and generalizability, researchers have recognized the need for more comprehensive datasets that sufficiently capture various environmental conditions.

Recently, large-scale datasets such as Waymo [33], ACDC [34], Foggy Cityscape [35], and RADIATE [36] have been proposed to address the need for adverse weather conditions. These rely on the high resolution of advanced sensors, particularly radar sensors, and extend data collection in a wide range of challenging weather conditions for reliable performance in diverse driving scenarios under all weather conditions. In addition to these datasets, researchers have also created custom datasets for specific road obstacle detection applications. In [37], a custom dataset was presented for autonomous driving environments. The images were collected using a camera mounted on a vehicle. This dataset included only ten object classes: vehicle, car, SUV, minibus, truck, bus, motorcyclist, pedestrian, bicyclist, and tricyclist, with 150,208 annotations created. In [38], a custom dataset was presented with images captured by a Raspberry Pi camera mounted on a vehicle's dashboard to train YOLOv3 and YOLOv5 models for road obstacle detection. Images were extracted from videos captured at 24 fps with a Raspberry Pi camera at a sample rate of 1 fps. The images were annotated using seven object classes: pedestrians, stray animals, speed bumps, etc. These public and custom datasets offer valuable contributions in training object detection models for road obstacle detection. However, these datasets do not account for different weather conditions, such as rain, fog, or others, that could improve the robustness of object detection models [38].

Although existing public datasets have contributed significantly to object detection, custom datasets are needed to overcome emerging challenges, especially in road safety and driver assistant systems in sub-Saharan Africa. This is specifically due to the lack of some object classes from these countries in the public datasets. Moreover, custom datasets proposed in the literature to train object detection models lack African-specific road obstacles, such as wild animals and livestock [37-38], which presents a gap in their usage in an African context. The lack of animal classes, such as cows, buffalos, zebras, goats, elephants, sheep, etc., makes obstacle detection models trained on these datasets susceptible to collisions in an autonomous environment when encountering such objects. The absence of African-specific object classes in both public and custom datasets presents a significant shortcoming in training obstacle detection models for applications in the African context. This is due to the reduced generalizability of the object detection models trained on these datasets, which leads to underperformance on absent objects. This complicates the development of robust object detection models for real-world applications such as road obstacle detection, especially for African road scenarios.

The main contributions of this study are:

- Develops and validates a custom dataset explicitly tailored for African road scenarios with rare African wildlife and livestock road obstacles.
- Quantitatively assesses how offline data augmentation techniques affect the object detection accuracy of YOLOv3, YOLOv4, and YOLOv5 models.

- Compares the YOLOv3, YOLOv4, and YOLOv5 models using a custom African road obstacle dataset.

II. TOOLS AND METHODS

A. Tools

TABLE I. TOOLS USED IN THE STUDY

Tool	Table column subhead
Label-Studio	Data annotation
CUDA and CUDNN	GPU acceleration during training
Nvidia GeForce RTX 3050 laptop	Training models
Samsung Galaxy A13 phone	Image capture

B. Data Preparation

Images of commonly encountered African road obstacles were obtained from multiple sources: 1,655 were collected from online image repositories using Imageye [39], 1,000 were extracted from the Open Image Dataset v4 (OIDv4) [40], and 396 were captured with a phone camera on an urban road in Kenya. In searching the images from the online repositories, the object names on the highways and rural and urban roads were used for the objects. The object classes considered included car, van, truck, person, motorcycle, bicycle, motorcyclist, tricyclist, dog, cow, elephant, pig, signpost, goat, sheep, zebra, buffalo, lion, horse, donkey, and minibus. Images of interest from the online repositories were downloaded using Imageye [39] in JPG and PNG formats. The PNG images were converted to a JPG format for dataset consistency and renamed with the corresponding object classes. Duplicates in the formatted image were removed using a Python script, and formatted images were then manually analyzed to remove Artificial Intelligence (AI) generated and unnecessary images, ensuring the inclusion of various lighting and weather conditions. Images were extracted from the OIDv4 dataset using the OIDv4_Toolkit and specific commands to specify object classes.

The dataset was further refined by subdividing the vehicle classes to reflect the real-world scenarios on road networks. A vehicle is encountered in various positions, and the collision avoidance decision depends on the actual position in which it is encountered. Hence, for each vehicle class, three subdivisions of the body of the vehicle represent the sides of the vehicles that may be encountered as stationary vehicles parked on the road: the rear of the vehicle represents the vehicles being approached from behind, and the front of the vehicle represents vehicles being approached in the opposite direction. These subdivisions ensure elaborate and detailed class definitions, reflecting the different orientations of the vehicles encountered in the actual road environment. A total of 3,051 images were collected for all object classes of commonly encountered road obstacles, with each considered class having at least 100 representative images. The images were then loaded into Label Studio, an open-source tool that exports annotations in YOLO format. Rectangular boxes were created around the objects of interest for the 33 object classes considered in the dataset. A comprehensive set of annotation guidelines with a clear definition of each object class and instructions for bounding box placement was used to ensure consistency of the bounding boxes created. This was used along with a two-stage quality annotation process involving initial annotation and review and validation with the help of Label Studio's quality control feature of show overlap. Figure 1 shows examples of annotated images.

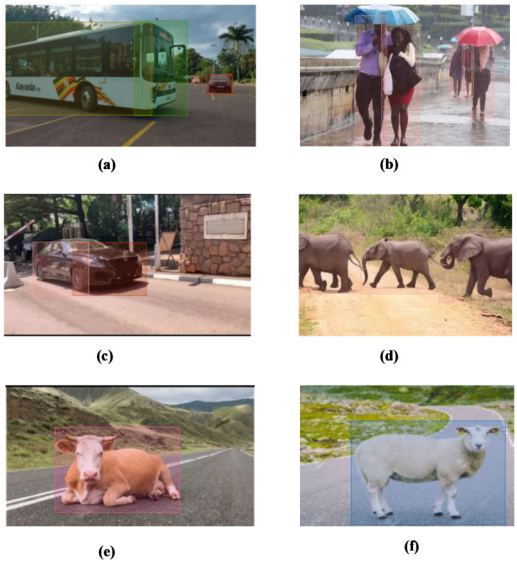

Fig. 1. Images from the dataset: (a) bus and car, (b) persons in rain, (c) car, (d) elephants, (e) cow, (f) sheep.

Using the Label Studio annotation tool, 8,133 labels were created for all the images. Figure 2 shows a histogram with a breakdown of annotations for each object class considered.

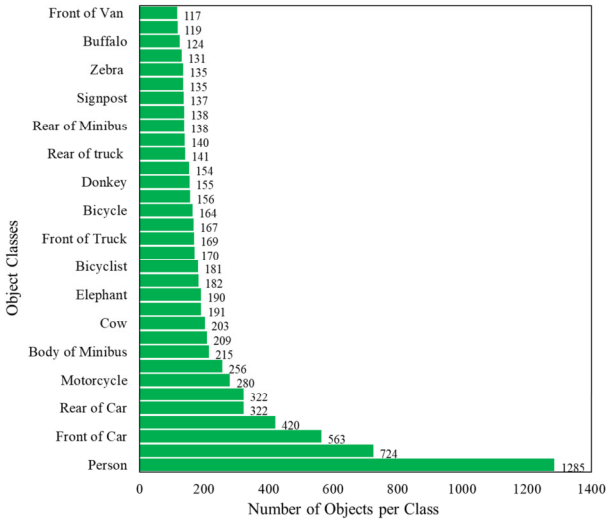

Fig. 2. Histogram showing the number of labels per class.

After labeling, the dataset was exported in a YOLO format to train the YOLO models. A single line in YOLO format represents each object class and its bounding box details in a given image as: <$object\ class$> <x_{centre}> <y_{centre}> <$width$> <$height$>. Here, <$object\ class$> is a whole number denoting

the class of the object assigned using an auto-increment integer starting from 0, $<x_{centre}>$ and $<y_{centre}>$ are the normalized coordinates of the center of the bounding box ranging from 0 to 1, while $<height>$ and $<width>$ represent the normalized height and width of the image, respectively, ranging from 0 to 1. These values were obtained using:

$$x_{centre} = {}^{x}/_{W} \quad (1)$$

$$y_{centre} = {}^{y}/_{H} \quad (2)$$

$$width = {}^{w}/_{W} \quad (3)$$

$$weight = {}^{h}/_{H} \quad (4)$$

where x, y, w, and h are the bounding box's x-coordinate, y-coordinate, width, and height, while W and H are the respective width and height of the whole image [41]. The encoded labels were stored in a separate .txt file within a "labels" subfolder. Each text file corresponds to a respective image file in the "images" subfolder.

C. Dataset Splitting and Data Augmentation

For each class in the dataset, the images were divided into an 80% training set, a 10% validation set, and a 10% testing set. Training sets for all object classes were combined to form the final training set, and the individual class testing sets were combined to form the test set. Similarly, individual class validation sets were combined to form the dataset validation set to ensure equal representation of all object classes.

The training set was subjected to various data augmentation techniques, such as flipping, rotation, brightness adjustment, and random noise, in Robflow [42] to increase the model's generalization and robustness in detecting objects under various conditions and illuminations. Random noise simulated the rainy weather conditions to enhance the model's detection even in rainy conditions. The brightness adjustment facilitated object detection for dark conditions at night by introducing poorly visible images. Horizontal flipping and counterclockwise 90° rotation were also applied to simulate real-world scenarios of obstacles in varying orientations and ensure reduced overfitting [25]. This resulted in 8,769 training images with their respective labels generated to train the YOLO models. Figure 3 shows some of the augmented images.

Fig. 3. Examples of augmented images in the dataset.

D. Data Records

The augmented African road obstacles dataset was uploaded to the Figshare platform [43] as a zip file, containing an image folder with the 9,378 images, an annotations folder with the corresponding labels, and a classes.txt file with all the 33 object classes considered. The adopted naming convention for the images was c.jpg, where c is the class name. The label files in the annotations folder adopt the same naming convention but instead use a .txt extension. All images had a resolution of 608×608 pixels.

E. Model Training

To allow a meaningful comparison across the YOLO generations while maintaining the practical constraints of this research related to available hardware resources, the YOLOv3, YOLOv4, and YOLOv5 models were trained on the custom dataset. The three chosen YOLO models are the most widely used YOLO versions in obstacle detection because of their balance between performance and accessibility. The YOLOv3 model was trained first in the darknet framework using the dataset in its original and augmented forms using a laptop with a Nvidia GeForce RTX 3050 with 6 GB RAM GPU. The choice of the darknet framework to train the YOLOv3 model was inspired by the fact that the model was originally developed in this framework [44]. The training hyperparameters were carefully selected, following the developer's guide to balance the model's performance and computational efficiency given the available training hardware resources.

TABLE II. TRAINING HYPERPARAMETERS

Hyperparameter	***Value***
Batch size	16
Learning rate	0.001
Image size	608 x 608
Weight decay	0.0005

A batch size of 16 was selected to maintain a balance between the memory constraints of training hardware and training stability, as a relatively smaller batch size reduces memory usage and allows for more frequent weight updates. The chosen moderate learning rate allowed the model to learn progressively while avoiding excessive oscillations. An image size of 608×608 pixels was chosen to ensure the detection of small objects while maintaining the computational constraints of the training hardware. A weight decay of 0.0005 was introduced to avoid overfitting by introducing L2 regularization to the loss function. The number of filters was calculated using [45]:

$$filters = (number\ of\ classes + 5) \times 3$$

The YOLOv3 model was trained on the original and augmented datasets for 66,000 iterations, calculated from $max_{batches} = 2000 \times number\ of\ classes$ [45], and the weights were saved after every 10,000 iterations for model evaluation on the validation dataset. To train the model, the pre-trained YOLOv3 convolutional weights file darknet53.conv.74 from the official YOLOv3 website, was used [44].

The training procedure was repeated with the YOLOv4 in a darknet framework using the same training hardware and hyperparameter values on both the original and augmented datasets. The yolov4.conv.137 pre-trained weights file was used. In contrast, the YOLOv5 model was initially developed and trained in the PyTorch framework due to its key advantages in terms of superior flexibility, user-friendliness, and enhanced performance capabilities. In addition, the computation graph obtained in a PyTorch framework is dynamic, allowing for easier debugging and experimentation. This is key when implementing new features or modifying the model, which is crucial for adapting various datasets and tasks. Thus, YOLOv5 was trained in a PyTorch framework using the same hyperparameters and hardware resources.

F. Testing the Trained YOLO Models on the Dataset

For each model, the best weights were used to make an inference on the testing dataset. Bounding Boxes (BBs) marked the detected objects with the corresponding class labels and their confidence scores. The confidence score, ranging from 0 to 1, indicates the probability that the model detects an image accurately in a given BB. BBs with an IoU lower than the IoU threshold of 0.5 were filtered out, removing bounding boxes with minimal overlap with actual objects. Non-Maximum Suppression (NMS) was applied for all the BBs predicting the same object so that only one BB with the highest confidence score was responsible for predicting it. The detected images were saved locally to analyze the correct labels.

G. Testing the Models with Webcam and Live Videos

Using OpenCV's video capture function, a Python script was written to capture separate frames from either a webcam or live road traffic videos. The captured frame was passed through the trained models independently for real-time object detection. The trained models identified objects within the captured frame, drew the BBs around the predicted objects, and assigned a confidence score to each detection, indicating the predicted object's certainty. The predicted images with the corresponding bounding boxes, class labels, and confidence scores were saved locally on the laptop for further analysis of the predictions.

H. Evaluation Metrics

The trained models were evaluated for their performance on the validation dataset using the mean Average Precision (mAP) metric at IoU thresholds of 0.5 and 0.75. The mAP is the mean of the dataset's average precisions of all object classes, calculated by integrating the recall-precision curve using [46]:

$$mAP = \int_0^1 \rho(x)dx \quad (5)$$

where $\rho(x)$ is the precision-recall curve.

TABLE III. PERFORMANCE OF THE TRAINED MODELS

Model	Inference time (ms)	Without augmentation		With augmentation	
		$mAP_{0.5}$	$mAP_{0.75}$	$mAP_{0.5}$	$mAP_{0.75}$
YOLOv3	26.3	71.44	52.09	89.74	81.62
YOLOv4	25.6	77.95	72.15	90.42	84.58
YOLOv5	24.2	84.66	76.55	94.68	88.80

III. RESULTS AND DISCUSSION

Table III shows the models' performance regarding the best-achieved mAP values at different IoU threshold values and the inference times. The results demonstrate that the three YOLO models significantly benefited from data augmentation, reflected in the increases in the mAP scores. This is because data augmentation artificially generates variations of the original data in the custom dataset, enriching it and broadening the spectrum of object representation that the model encounters [24]. Therefore, the models learn invariant features from the different variations generated, leading to increased generalization and robustness in detecting objects under various conditions.

YOLOv5 outperformed YOLOv3 and YOLOv4, as demonstrated by the best mAP scores with and without data augmentation on the dataset. This superior object detection accuracy performance of YOLOv5 over YOLOv4 and YOLOv3 agrees with what has been recorded in recent studies comparing the three YOLO models [12, 47, 48]. The superiority of YOLOv5 on the custom dataset can be attributed to its ability to auto-learn the BB anchors [49], a specific feature that allows better adaptability to specific datasets. The auto-learning results in best-matched anchors, leading to better initial guesses and allowing faster convergence. While YOLOv3 exhibited the least mAP, it demonstrated a remarkable improvement when trained with augmented data on the custom dataset. Its mAP at an IoU threshold of 0.5 increased to 89.74% and 81.62% at an IoU threshold of 0.75, surpassing YOLOv4's performance without data augmentation. YOLOv4's performance at IoU thresholds of 0.5 and 0.75 was better than that of YOLOv3 but lagged behind YOLOv5. Compared to YOLOv3, which uses a darknet53 backbone that struggles to detect small objects, YOLOv4 and YOLOv5 use CSPdarknet53 with freebies and a bag of specials, significantly increasing object detection accuracy [12]. Interestingly, YOLOv3 achieved an inference time close to that of the other two models. Thus, YOLOv3 with offline data augmentation can achieve a high object detection accuracy with an inference speed closer to those of YOLOv4 and YOLOv5.

Figure 4 shows examples of the predicted objects using the trained model through a webcam, live videos, and images from the dataset's test set. It can be observed that the trained models correctly detected objects in images, webcam, and live road traffic videos with correct BBs, class labels, and higher confidence scores at an IoU threshold of 0.5. This shows that the models trained on the custom dataset can be effectively deployed in real-world applications for real-time road obstacle detection. Higher confidence scores in the detected objects indicate the models' confidence in reliably detecting the objects. The results of this study reveal the relevance of the developed dataset for object detection in African road scenarios, as demonstrated by the high mAP values obtained, particularly with the application of data augmentation. Table IV shows how the trained models compare with other studies using custom datasets.

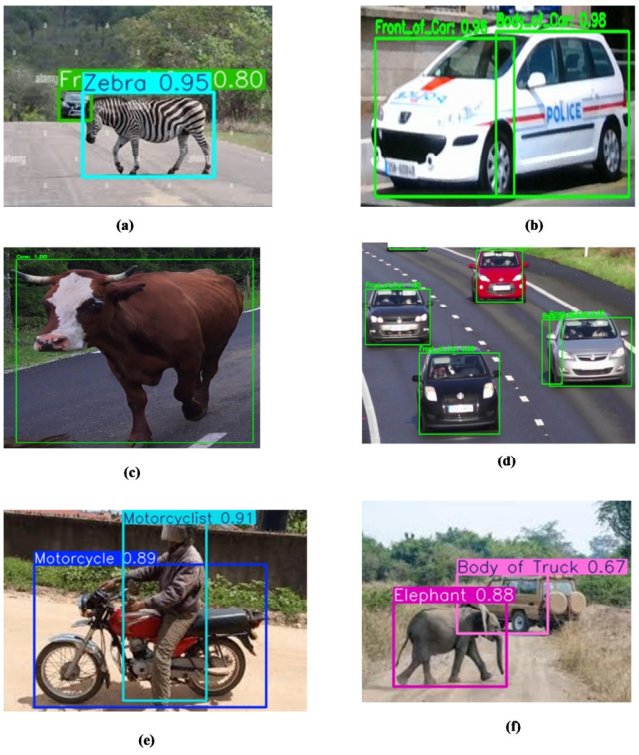

Fig. 4. Testing results: (a) and (b) from the webcam, (c) and (d) from live videos, and (e) and (f) from the testing set.

TABLE IV. COMPARISON WITH PREVIOUS STUDIES

Study	Model used	mAP (%)
[50]	YOLOv5	81.02
[38]	YOLOv3	76.26
	YOLOv5	73.78
[37]	YOLOv3	79.40
This study	YOLOv5	94.68
	YOLOv4	90.42
	YOLOv3	89.74

IV. CONCLUSION

This study compared the performance of three YOLO models on a custom dataset tailored for African road scenarios and evaluated the effectiveness of the dataset with rare object classes on African roads for road obstacle detection. The dataset comprised 33 object classes of commonly encountered road obstacles with rare animal species, addressing a critical gap in existing datasets. Offline data augmentation was applied to enhance the models' ability to generalize and detect objects reliably and accurately in real-world scenarios. The findings showed that YOLOv5 outperformed the object detection accuracy of the two other YOLO models, both with and without data augmentation, indicated by the highest achieved mAP. Data augmentation drastically improved the object detection accuracy of the YOLOv3 model, bringing it closer to the detection accuracy of YOLOv4 and YOLOv5. In particular, these results demonstrate that the trained models on the custom dataset achieve object detection accuracies higher than those recorded in the literature. Future work should focus on expanding the dataset to include additional object classes with additional images captured using advanced sensors, such as LiDAR and radar sensors, to enhance object detection accuracy. Furthermore, deploying the trained models for real-world obstacle detection could reveal more about the effectiveness of the dataset in road obstacle detection.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 580–587, https://doi.org/10.1109/CVPR.2014.81.

[2] R. Girshick, "Fast R-CNN." arXiv, Sep. 27, 2015, [Online]. Available: https://doi.org/10.48550/arXiv.1504.08083.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, https://doi.org/10.1109/TPAMI.2016.2577031.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010, https://doi.org/10.1109/TPAMI.2009.167.

[5] 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 (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788, https://doi.org/10.1109/CVPR.2016.91.

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

[7] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021." arXiv, Aug. 06, 2021, https://doi.org/10.48550/arXiv.2107.08430.

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

[9] K. Li, Y. Zhuang, J. Lai, and Y. Zeng, "PFYOLOv4: An Improved Small Object Pedestrian Detection Algorithm," *IEEE Access*, vol. 11, pp. 17197–17206, 2023, https://doi.org/10.1109/ACCESS.2023.3244981.

[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] U. Nepal and H. Eslamiat, "Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs," *Sensors*, vol. 22, no. 2, Jan. 2022, Art. no. 464, https://doi.org/10.3390/s22020464.

[12] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "Real-Time Vehicle Detection Based on Improved YOLO v5," *Sustainability*, vol. 14, no. 19, Jan. 2022, Art. no. 12274, https://doi.org/10.3390/su141912274.

[13] R. Rajamohanan and B. C. Latha, "An Optimized YOLO v5 Model for Tomato Leaf Disease Classification with Field Dataset," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12033–12038, Dec. 2023, https://doi.org/10.48084/etasr.6377.

[14] 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.

[15] 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, Jun. 2012, pp. 3354–3361, https://doi.org/10.1109/CVPR.2012.6248074.

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

[17] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, https://doi.org/10.1007/s11263-015-0816-y.

[18] O. Zendel, M. Schorghuber, B. Rainer, M. Murschitz, and C. Beleznai, "Unifying Panoptic Segmentation for Autonomous Driving," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 21319–21328, https://doi.org/10.1109/CVPR52688.2022.02066.

[19] D. Wu *et al.*, "YOLOP: You Only Look Once for Panoptic Driving Perception," *Machine Intelligence Research*, vol. 19, no. 6, pp. 550–562, Dec. 2022, https://doi.org/10.1007/s11633-022-1339-y.

[20] H. L. Nguyen, D. T. Le, and H. H. Hoang, "Application of Synthetic Data on Object Detection Tasks," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15695–15699, Aug. 2024, https://doi.org/10.48084/etasr.7929.

[21] W. Jiang, C. Song, H. Wang, M. Yu, and Y. Yan, "Obstacle Detection by Autonomous Vehicles: An Adaptive Neighborhood Search Radius Clustering Approach," *Machines*, vol. 11, no. 1, Jan. 2023, Art. no. 54, https://doi.org/10.3390/machines11010054.

[22] F. Gao, C. Li, and B. Zhang, "A Dynamic Clustering Algorithm for Lidar Obstacle Detection of Autonomous Driving System," *IEEE Sensors Journal*, vol. 21, no. 22, Aug. 2021, Art. no. 25922–25930, https://doi.org/10.1109/JSEN.2021.3118365.

[23] G. Al-Pefai and M. Al-Pefai, "Road object detection using Yolov3 and Kitti dataset," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 8, 2020.

[24] A. Mumuni and F. Mumuni, "Data augmentation: A comprehensive survey of modern approaches," *Array*, vol. 16, Dec. 2022, Art. no. 100258, https://doi.org/10.1016/j.array.2022.100258.

[25] P. Siripatthiti, "Data Augmentations for Improving Vision-Based Damage Detection : in Land Transport Infrastructure," M.S. Thesis, Kungliga Tekniska Högskolan (KTH), Stockholm, Sweden, 2023.

[26] "Ultralytics | Revolutionizing the World of Vision AI." https://www.ultralytics.com/.

[27] "OpenMMLab," *GitHub*. https://github.com/open-mmlab.

[28] "openvinotoolkit/openvino." OpenVINO™ Toolkit, Nov. 15, 2024, [Online]. Available: https://github.com/openvinotoolkit/openvino.

[29] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "RandAugment: Practical automated data augmentation with a reduced search space." arXiv, Nov. 14, 2019, https://doi.org/10.48550/arXiv.1909.13719.

[30] M. Patil, M. M. Patil, and S. Agrawal, "WGAN for Data Augmentation," in *GANs for Data Augmentation in Healthcare*, A. Solanki and M. Naved, Eds. Cham, Switzerland: Springer International Publishing, 2023, pp. 223–241.

[31] G. B. Rajendran, U. M. Kumarasamy, C. Zarro, P. B. Divakarachari, and S. L. Ullo, "Land-Use and Land-Cover Classification Using a Human Group-Based Particle Swarm Optimization Algorithm with an LSTM Classifier on Hybrid Pre-Processing Remote-Sensing Images," *Remote Sensing*, vol. 12, no. 24, Jan. 2020, Art. no. 4135, https://doi.org/10.3390/rs12244135.

[32] P. Sun *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 2443–2451, https://doi.org/10.1109/CVPR42600.2020.00252.

[33] P. Sun *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset." arXiv, May 12, 2020, https://doi.org/10.48550/arXiv.1912.04838.

[34] C. Sakaridis *et al.*, "ACDC: The Adverse Conditions Dataset with Correspondences for Robust Semantic Driving Scene Perception." arXiv, Jun. 07, 2024, https://doi.org/10.48550/arXiv.2104.13395.

[35] M. Meyer and G. Kuschk, "Automotive Radar Dataset for Deep Learning Based 3D Object Detection," in *2019 16th European Radar Conference (EuRAD)*, Paris, France, Jul. 2019, pp. 129–132.

[36] M. Sheeny, E. De Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "RADIATE: A Radar Dataset for Automotive Perception in Bad Weather," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, May 2021, pp. 1–7, https://doi.org/10.1109/ICRA48506.2021.9562089.

[37] J. Li, Y. Zhao, L. Gao, and F. Cui, "Compression of YOLOv3 via Block-Wise and Channel-Wise Pruning for Real-Time and Complicated Autonomous Driving Environment Sensing Applications," in *2020 25th International Conference on Pattern Recognition (ICPR)*, Milan, Italy, Jan. 2021, pp. 5107–5114, https://doi.org/10.1109/ICPR48806.2021.9412687.

[38] C. N. Jaikishore *et al.*, "Implementation of Deep Learning Algorithm on a Custom Dataset for Advanced Driver Assistance Systems Applications," *Applied Sciences*, vol. 12, no. 18, Jan. 2022, Art. no. 8927, https://doi.org/10.3390/app12188927.

[39] "Imageye - Image Downloader." https://www.imageye.net/.

[40] "Open Images V7." https://storage.googleapis.com/openimages/web/index.html.

[41] A. Menon, B. Omman, and A. S, "Pedestrian Counting Using Yolo V3," in *2021 International Conference on Innovative Trends in Information Technology (ICITIIT)*, Kottayam, India, Feb. 2021, pp. 1–9, https://doi.org/10.1109/ICITIIT51526.2021.9399607.

[42] "Roboflow: Computer vision tools for developers and enterprises." https://roboflow.com/.

[43] "figshare - credit for all your research." https://figshare.com/.

[44] "YOLO: Real-Time Object Detection," *Twitch*. https://pjreddie.com/darknet/yolo/.

[45] A. Bochkovskii, "AlexeyAB/darknet." Nov. 15, 2024, [Online]. Available: https://github.com/AlexeyAB/darknet.

[46] J. S. Walia and K. Seemakurthy, "Optimized Custom Dataset for Efficient Detection of Underwater Trash," in *Towards Autonomous Robotic Systems*, Cambridge, UK, 2023, pp. 292–303, https://doi.org/10.1007/978-3-031-43360-3_24.

[47] A. Kuznetsova, T. Maleva, and V. Soloviev, "YOLOv5 versus YOLOv3 for Apple Detection," in *Cyber-Physical Systems: Modelling and Intelligent Control*, A. G. Kravets, A. A. Bolshakov, and M. Shcherbakov, Eds. Cham, Switzerland: Springer International Publishing, 2021, pp. 349–358.

[48] O. Kıvrak and M. Z. Gürbüz, "Performance Comparison of YOLOv3,YOLOv4 and YOLOv5 algorithms : A Case Study for Poultry Recognition," *Avrupa Bilim ve Teknoloji Dergisi*, no. 38, pp. 392–397, Aug. 2022, https://doi.org/10.31590/ejosat.1111288.

[49] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, vol. 126, Jun. 2022, Art. no. 103514, https://doi.org/10.1016/j.dsp.2022.103514.

[50] A. Ben Atitallah, Y. Said, M. A. Ben Atitallah, M. Albekairi, K. Kaaniche, and S. Boubaker, "An effective obstacle detection system using deep learning advantages to aid blind and visually impaired navigation," *Ain Shams Engineering Journal*, vol. 15, no. 2, Feb. 2024, Art. no. 102387, https://doi.org/10.1016/j.asej.2023.102387.