Enhancing object detection for humanoid robot soccer: comparative analysis of three models

Handaru Jati[1], Nur Alif Ilyasa[1], Yuniar Indrihapsari[1], Ariadhie Chandra[2], Dhanapal Durai Dominic[3]
[1]Department of Electronics and Informatics Engineering Education, Faculty of Engineering, University Negeri Yogyakarta, Special District of Yogyakarta, Indonesia
[2]Department of Electrical Engineering Education, Faculty of Engineering, University Negeri Yogyakarta, Special District of Yogyakarta, Indonesia
[3]Departement of Computer and Information Science, University Technology Petronas, Bandar Seri Begawan, Malaysia

Article Info

Article history:

Received Dec 6, 2023
Revised Mar 14, 2024
Accepted Mar 29, 2024

Keywords:

Humanoid robot soccer
Object detection
YOLO-NAS
YOLOv7
YOLOv8

ABSTRACT

The humanoid robot soccer system encounters a notable challenge in object detection, primarily concentrating on identifying the ball and often neglecting crucial elements like opposing robots and goals, resulting in on-field collisions and imprecise ball shooting. This study comparatively evaluates three you only look once (YOLO) real-time object detection system variants: YOLOv8, YOLOv7, and YOLO-NAS. A dataset of 2104 annotated images, covering classes such as ball, goalpost, and robot, was curated from Roboflow and robot-captured images. The dataset was partitioned into training, validation, and testing sets, and each YOLO model underwent extensive fine-tuning over 100 epochs on this custom dataset, leveraging the pre-trained common objects in context (COCO) model. Evaluation metrics, including mean average precision (mAP) and inference speed, assessed performance. YOLOv8 achieved the highest accuracy with a mAP of 0.92, while YOLOv7 showed the fastest inference speed of 24 ms on the Jetson Nano platform. Balancing accuracy and speed, YOLO-NAS emerged as the optimal choice. Thus, YOLO-NAS is recommended for object detection for humanoid soccer robots, regardless of team affiliation. Future research should focus on enhancing object detection through advanced training techniques, model architectures, and sensor fusion for improved performance in dynamic environments, potentially optimizing through scenario-specific fine-tuning.

Corresponding Author:

Handaru Jati
Department of Electronics and Informatics Engineering Education, Faculty of Engineering
University Negeri Yogyakarta
Depok, Sleman, Daerah Istimewa Yogyakarta 55281, Indonesia
Email: handaru@uny.ac.id

1. INTRODUCTION

Robot soccer, also known as the RoboCup humanoid league, presents an intriguing and demanding domain within robotics, aiming to advance the development of self-governing humanoid robots capable of engaging in soccer matches. This field not only serves as a proving ground for state-of-the-art artificial intelligence, computer vision, motion planning, and control algorithms but also propels progress in both robotics and artificial intelligence, all while offering an enjoyable and interactive platform to showcase the capabilities of autonomous humanoid machines.

In humanoid robot soccer, one pivotal challenge pertains to the object detection system. The system's capability is limited to identifying the soccer ball, failing to recognize other critical elements such as opponent robots and the goalposts. The system relies on outdated techniques like basic segmentation and Hough circle methods, which have become less effective since the ball’s color mark was replaced with white color and spot patterns under current RoboCup regulations. This modification has significantly increased the difficulty of ball detection, resulting in frequent false positives due to the prevalence of white objects in the soccer environment. Furthermore, the absence of awareness regarding other robots and the goal often leads to collisions with other robots and challenges in aligning shots toward the goal.

From 2020 to 2023, many studies in humanoid robot soccer have delved into object detection systems, as detailed in Table 1. A notable portion of these investigations has incorporated the you only look once (YOLO) algorithm, revolutionizing computer vision with its ability to swiftly and accurately identify multiple objects in images and videos in real-time [1]. Unlike conventional methods that rely on multi-stage processes, YOLO’s innovative design enables it to process the entire image simultaneously, rendering it exceptionally rapid and efficient [2]. The data in Table 1 highlights the widespread adoption of YOLO, bolstered by many researchers who attest to its robust performance.

Table 1. Publication papers related to humanoid robot soccer object detection publication paper method/model

Publication Paper	Method/Model
[3]-[5]	CNN
[6], [7]	YOLOv3
[8]-[10], [11]	YOLO
[12]	XNOR-YOLO
[11]	MobileNet
[13]	YOLOv4
[14]	ROBO
[15]	YOLOv2
[16]	YOLOv4-Tiny
[15]	Faster RCNN
[17]	YOLOv7
[18]	YOLOv8
[19]	NimbroNet
[20]	Unet
[10]	SSD

Object detection has seen a proliferation of YOLO variants, showcasing their adaptability and versatility [21]. However, subsequence versions of YOLO have developed [22]. A comprehensive evaluation of each YOLO variant’s strengths and limitations and robust benchmarking on a relevant dataset are crucial for making an informed decision. This ensures that the selected YOLO model significantly improves object detection capabilities in humanoid robot soccer, regardless of team affiliation or specific application.

This study focuses on three prominent YOLO variants: YOLOv8, YOLOv7, and YOLO-NAS, chosen for their significance and relevance in object detection [23]. Each variant represents a distinct evolution of the YOLO algorithm, incorporating specific enhancements and optimizations [22]. YOLOv8 is known for its exceptional accuracy [24], [25], making it an excellent choice for applications where precision is crucial. Conversely, YOLOv7 is praised for its impressive speed, vital for real-time object detection scenarios [26]. Finally, YOLO-NAS is designed with an architecture that prioritizes efficiency, striking a balance between accuracy and speed, rendering it a notable contender across various applications [23].

However, a systematic comparison is currently lacking, impeding the ability to make an informed decision that could enhance object detection capabilities in humanoid robot soccer, regardless of team affiliation or specific application [27]. Therefore, this study aimed to comprehensively evaluate these three YOLO variants, considering the diverse requirements of object detection in humanoid robot soccer. The goal is to determine which YOLO variant is best suited for optimizing object detection in this context. This approach ensures that the selected YOLO model aligns seamlessly with the sport’s unique demands, accounting for computational resources, real-time performance, accuracy, and the types of objects commonly encountered in the soccer environment.

2. METHOD

Based on the literature study explained in the previous section, YOLO is proposed as a new detection approach. A YOLO was a convolutional neural network (CNN) model and an end-to-end machine learning workflow procedure was performed in this research. The procedure consists of the data engineering stage, model engineering stage, and code engineering stage as shown in Figure 1 [28].

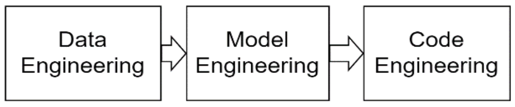

Figure 1. Machine learning model development procedure

2.1. Data engineering

Data in this context is data to train the model. Data engineering or data preparation stages involve a subsequence process to prepare a dataset to train the model. This involves data collection, cleaning, annotation, splitting, exploratory data analysis (EDA), and data augmentation [28]. Data preparation for YOLO training involves accumulating and refining a dataset of labeled images from humanoid robot soccer matches, supplemented by custom data from scripts on robots. Ensuring dataset quality requires cleansing to remove any irrelevant or redundant images. YOLO needs images and representative annotations to train the model [29]. Annotations, defined by bounding boxes, contain class, location, and size information. Labeling software is used for manual annotation, after which datasets are divided into training, validation, and test sets. Data augmentation techniques, informed by EDA, enhance dataset diversity.

2.2. Model engineering

The model engineering stage consists of modelling, model training, optimization, and evaluation [28]. A Python script notebook organizes training code, setting hyperparameters like batch size and learning rate for optimal training. Model optimization involves simplifying the model to reduce complexity and latency while maintaining detection performance. Upon completion of the model training, the next step is evaluation, which is conducted using a dedicated test dataset excluded from the training process. This dataset consists of unseen data, facilitating an unbiased evaluation of the model's generalization capabilities. Metrics such as mean average precision (mAP) and average precision (AP) are utilized to quantify the model's detection accuracy [30]. To compare the performance of trained object detection models, it is important to understand the metrics commonly used to analyze them. The evaluation metrics are calculated using the values of true positives (TP), false positives (FP), and false negatives (FN). In object detection, true negatives are not used, as the background does not need a bounding box prediction. To determine whether the model's bounding box prediction is correct, the intersection over the union (IoU) area is measured [31]. IoU gauges the accuracy of the predicted bounding box by assessing its overlap with the ground-truth bounding box for a particular object in the image. It quantifies the extent of the Intersection between the predicted and ground-truth bounding boxes [31].

The IoU is calculated by finding the area of the Intersection between the two bounding boxes and dividing it by the area of the union of the two bounding boxes [30]. The calculating IoU is as in (1):

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \tag{1}$$

In object detection, a threshold value for IoU is often defined to determine whether a predicted bounding box is considered a true positive detection. The detection is classified as a true positive if the IoU between the predicted bounding box and the ground-truth bounding box exceeds the threshold. Otherwise, it is considered a FP.

AP measures how well the model localizes and identifies objects in an image, considering both precision and recall [18]. Precision quantifies the model's accuracy in identifying positive detections, with high values suggesting precise object detection and low values indicating frequent false positives [18]. To calculate the precision, the written as in (2). Recall assesses the model's ability to identify actual objects, with high recall reflecting the detection of most objects and low recall signaling many undetected objects [18]. Recall can be written as in (3).

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

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

First, the model's predictions are sorted based on their confidence scores to compute AP. Then, a precision-recall curve is generated by varying the confidence score threshold for considering a prediction as a positive detection. For each threshold, precision and recall values are computed. The precision values are interpolated to obtain a smooth precision-recall curve for increasing recall values. AP is computed as the area under the precision-recall curve. It summarizes the overall performance of the model across all confidence score thresholds. AP can be expressed as in (4), where $p(r)$ is the precision at a given recall value R, and dr is the differential notation representing an infinitesimally small change in the recall. In object detection tasks, multiple object classes are often present. To evaluate the model's performance across all classes, the AP for each class is calculated, and the mean of these AP values is computed to obtain mAP. MAP can be expressed as in (5), where Q is the number of queries in the set and $AP(q)$ is the AP for a given query, q.

$$AP = \int_0^1 p(r)dr \tag{4}$$

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{5}$$

A higher AP or mAP indicates better object detection performance, signifying more accurate localization and identification of objects in the images [32]. The evaluation of object detection models involves using Python notebooks, deep learning libraries, and custom scripts to calculate metrics like mAP on validation datasets, with visualization tools assisting in analysis.

2.3. Code engineering

In the code engineering phase, the trained model is integrated with the system, linking model inference to the detection workflow. Compatibility with the robot's hardware and software is crucial. Post-deployment testing evaluates the model's detection efficacy from 1 to 10 meters, using automated tools to identify performance bottlenecks, supported by embedded logging for data collection.

3. RESULT AND DISCUSSION

3.1. Data engineering

3.1.1. Dataset collection

A total of 5,910 raw images were gathered from Roboflow and robot-operated field recordings. These images featured varied lighting and scenarios, including balls, robots, and goalposts. Additionally, they included elements such as potential blurring, distance, and obstruction.

3.1.2. Dataset cleaning

The dataset cleaning identified duplicates and irrelevant images, which were subsequently removed to maintain model accuracy. As a result, 2,104 images were retained post-cleaning. This careful curation ensured the quality and relevance of the dataset for training and testing the model.

3.1.3. Dataset annotation

For dataset annotation, LabelImg was used to label the images. The annotations were saved as text files in YOLO format, including the class index and bounding box coordinates. Figure 2 shows the images with the drawn bounding box annotations.

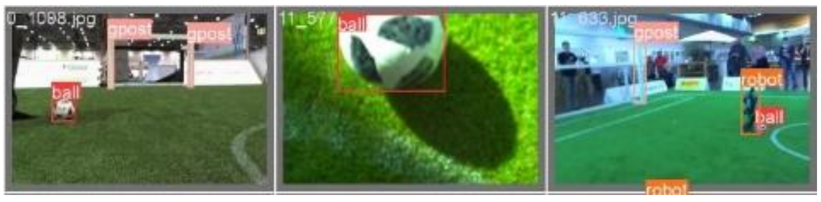

Figure 2. Samples of annotated images

3.1.4. Dataset splitting

The total dataset, consisting of 2054 images, is partitioned into training, validation, and test subsets using an 80:10:10% ratio. This results in 1644 images for training, 205 for validation, and 205 for testing. This distribution strategy aims to strike a balance between facilitating a comprehensive learning process and enabling a rigorous evaluation of the model, ensuring exposure to a diverse set of data while retaining a significant subset for performance assessment.

3.1.5. Exploratory data analysis

EDA is an analytical technique employed to evaluate the quality and suitability of datasets prior to training. In this phase, the focus is on examining the distribution of classes and generating heatmaps of the bounding boxes within each frame. These analyses confirm that the dataset is adequately prepared and ideal for subsequent model training. The complete dataset comprises 4106 bounding boxes, divided into 2010 for balls, 1293 for goalposts, and 803 for robots. The provided graph in Figure 3 visually represents the distribution of bounding boxes across these three classes.

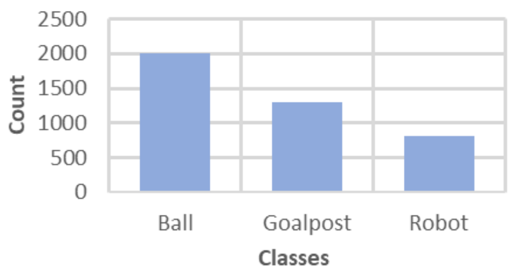

Figure 3. Each class bounding boxes count

From the graph presented above, it is evident that the dataset is imbalanced. The class 'Ball' has the highest number of bounding boxes, followed by 'Goalposts' and 'Robots.' This disparity in class representation could potentially lead to lower detection metrics for classes that appear less frequently in the dataset. Subsequently, the researcher analyzes the spatial distribution of bounding boxes for each class within the frame. The distribution map in Figure 4 illustrates that the placement of bounding boxes varies across different classes. Specifically, the distribution of bounding boxes for the 'Ball' class is not uniform across the frame.

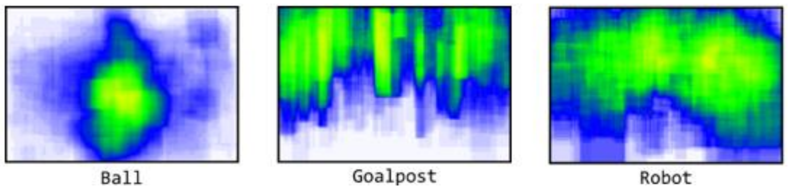

Figure 4. Each class bounding boxes distribution heatmap on a frame

3.1.6. Data augmentation

In response to the observed imbalance and uneven distribution of bounding boxes in the dataset, data augmentation, specifically mosaic techniques, is proposed as a solution. Figure 5 shows the results of applying mosaic augmentation to our dataset. Mosaic data augmentation, which combines multiple images into a single training example, increases the diversity and complexity of the data.

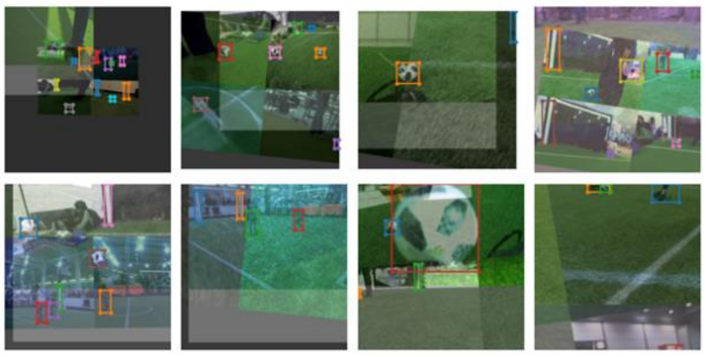

Figure 5. Mosaic augmented datasets

3.2. Modelling

3.2.1. Training YOLO model

Our object detection model was trained using fine-tuning on a pre-trained network to expedite convergence on our dataset rather than starting from scratch. Our framework's limitation necessitates this method only to support fine-tuning. Training began with a 100-epoch baseline, extendable if metrics were unstable, with batch sizes tailored to our hardware limitations. We trained three YOLO variants using their respective frameworks: YOLOv7 with Wongkinyiu's, YOLOv8 with Ultralytics, and YOLO-NAS with Supergradients, using Python notebooks for efficiency. Each model achieved convergence by 100 epochs, as further training showed negligible gains, supported by the graph of the mAP progression provided in Figure 6.

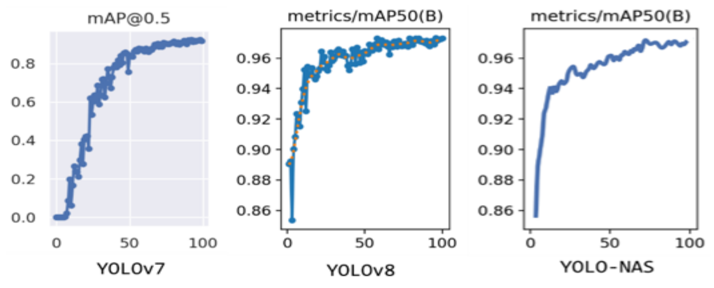

Figure 6. Each YOLO variants training performance

3.2.2. Model optimization

It is automatically simplified to reduce the complexity model using open neural network exchange (ONNX). Furthermore, the model's precision is decreased from floating point 32 (FP32) to floating point 16 (FP16), enhancing performance. Following these modifications, TensorRT is employed to compile the ONNX model. This compilation process incorporates techniques such as layer fusing and tensor fusing, all of which contribute to optimizing the model's inference performance. Nevertheless, it is essential to acknowledge that these optimization approaches may slightly compromise mAP metrics in exchange for enhanced inference speed.

3.2.3. Model evaluation

The performance metrics of the model were assessed on both the training device and the target deployment device, employing the test split dataset for evaluation. The primary metrics utilized to evaluate our trained YOLO models were mean mAP and inference speed. Comparisons were also conducted between the raw trained model and the TensorRT compiled model to provide comprehensive insight into the performance enhancements achieved through model optimization. Table 2 shows the results of the evaluation in terms of speed and mAP. The pipeline duration in milliseconds (ms) shown in Table 2 represents the total time taken from input preprocessing to postprocessing, including model inference.

Table 2. The comparative performance of YOLOv7, YOLOv8, and YOLO-NAS in terms of speed and mAP

Model variant	Device	Engine	Speed			mAP@0.5			
			Inference (ms)	Pipeline (ms)	FPS	Ball	Goalpost	Robot	Total
YOLOv7	RTX3060	Pytorch	1.8	5.2	192.31	0.986	0.966	0.896	0.949
		TensorRT	0.8	1.7	588.24	0.959	0.811	0.623	0.798
	Jetson Nano	TensorRT	24.8	34.6	28.90	0.959	0.811	0.623	0.798
YOLOv8	RTX3060	Pytorch	3.6	6.8	147.06	0.995	0.972	0.935	0.967
		TensorRT	1.4	2.7	370.37	0.990	0.922	0.866	0.926
	Jetson Nano	TensorRT	61.1	75.2	13.30	0.990	0.922	0.866	0.926
YOLO-NAS	RTX3060	Pytorch	3.2	5	200.00	0.995	0.972	0.935	0.965
		TensorRT	1.2	2	500.00	0.979	0.892	0.820	0.897
	Jetson Nano	TensorRT	48.2	57.1	17.51	0.979	0.892	0.820	0.897

To compare YOLO object detection metrics, researchers visualize the result in the graph in Figure 7. The square point represents the compiled model's performance. The circle point represents the raw trained model's performance. We can see the performance drop of the mAP as well as the increase of FPS between compiled and raw-trained model. The line between square and circle points represents the performance gap between the compiled and raw trained model. To compare the raw-trained model with the compiled model. These comparisons result from the measurement of the training device (Nvidia RTX 3060 12 GB).

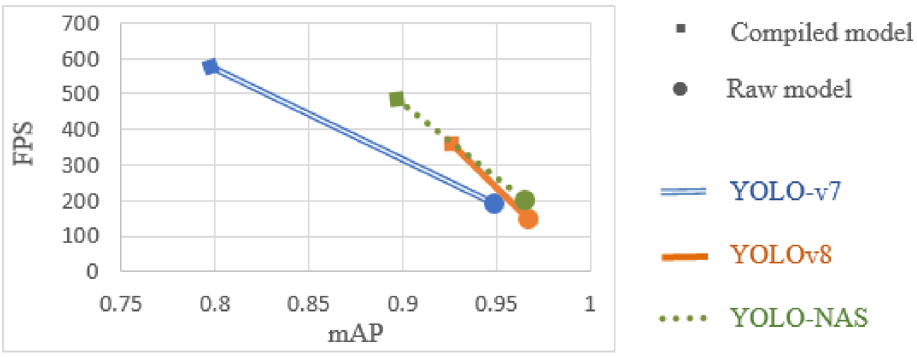

Figure 7. Comparison of each model performance on FPS and mAP trade-off

Blue represents YOLOv7, orange is YOLOv8, and green is YOLO-NAS. Circle points represent a raw-trained model. Square points represent the compiled model. YOLOv7 records the lowest mAP at 0.79 and performs least well in robot detection. YOLOv8 leads with an mAP of 0.92, with YOLO-NAS at 0.89. YOLOv7 is fastest with a 24 ms inference speed on Jetson Nano, YOLO-NAS follows at 48 ms, and YOLOv8 at 61 ms. YOLOv7 experiences the most significant performance drop after optimization, while YOLOv8 has the least. YOLO-NAS offers a balance between speed and precision, making it the most suitable compiled model overall.

3.3. Code engineering

The Python-based detection pipeline consists of streamlined preprocessing, inference, and post-processing, with image normalization in YOLO-NAS and integrated NMS in TensorRT enhancing efficiency. The pipeline, as shown in Figure 8, initially processes the image using OpenCV, converting it from RGB to the BGR color space and normalizing pixel values to a 0-1 range. These preprocessed inputs are then fed into the GPU for inference, producing outputs including detection count, bounding box coordinates, class indices, and confidence scores via EfficientNMS TRT. To validate model performance, a range test assesses object detection capabilities across distances. Success is determined by the accurate depiction of bounding boxes on live video feeds; failures are marked by their absence. The models consistently detected all object classes, with performance varying by distance, as detailed in Figure 9's depiction of detection success rates

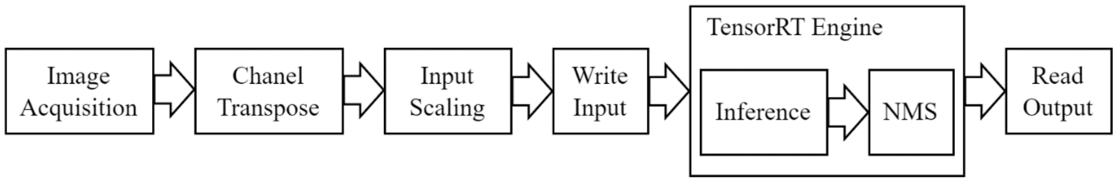

Figure 8. Object detection pipeline

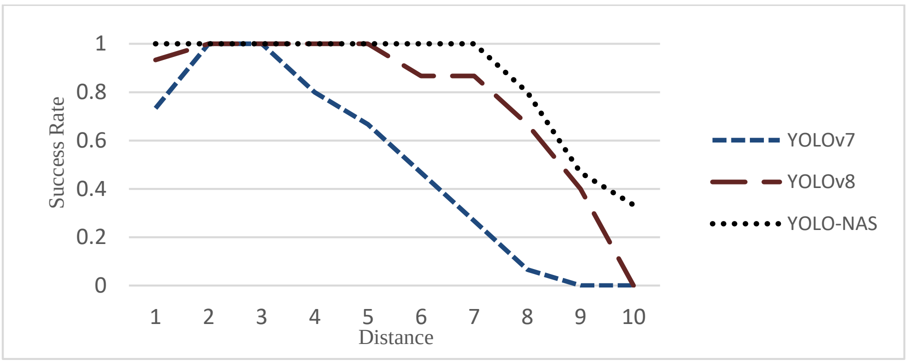

Figure 9. Result of deployment sanitation test against various distances

Compared to Szemenyei and Estivill-Castro [33] using the ROBO object detection system, which gained 84.5 mAP in accuracy and 13 FPS in speed. We can conclude that we have a similar result but a slightly better performance using YOLO-NAS. Szemenyei and Estivill-Castro [33] have a 200% increase in speed when introduced on pruning, while in our research, we have a 243% speed increase when introduced upon the implementation of quantization.

4. CONCLUSION

This study indicates that YOLO-NAS outperforms its counterparts by achieving a balance between inference speed (17.496 FPS on Nvidia Jetson Nano) and accuracy (mAP of 0.8968 at 0.5 IoU). Despite a 7% drop in mAP for a 243% gain in speed upon compilation, YOLO-NAS maintains robust detection from distances of 1 to 7 meters, although performance metrics are contingent on the specific test datasets used. Teams are advised to implement YOLO-NAS for their detection systems. The study highlights a notable gap in the form of a dedicated theoretical framework for YOLO application in this field, suggesting an opportunity for future research to develop a targeted approach. Recommendations for future research include considering hardware upgrades capable of int8 quantization for faster processing and exploring C++ implementation for more efficient GPU utilization, which could potentially increase frame rates over the current Python-based methods.

ACKNOWLEDGEMENTS

The completion of this research was made possible through the utilization of the Universitas Negeri Yogyakarta's Humanoid Robot Soccer.

REFERENCES

[1] S.-S. Park, V.-T. Tran, and D.-E. Lee, "Application of Various YOLO Models for Computer Vision-Based Real-Time Pothole Detection," *Applied Sciences*, vol. 11, no. 23, p. 11229, Nov. 2021, doi: 10.3390/app112311229.

[2] 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: IEEE, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.

[3] S. A. Irfan and N. S. Widodo, "Application of Deep Learning Convolution Neural Network Method on KRSBI Humanoid R-SCUAD Robot," *Buletin Ilmiah Sarjana Teknik Elektro*, vol. 2, no. 1, pp. 40-50, May 2020, doi: 10.12928/biste.v2i1.985.

[4] D. Zhou, G. Chen, and F. Xu, "Application of Deep Learning Technology in Strength Training of Football Players and Field Line Detection of Football Robots," *Frontiers in Neurorobotics*, vol. 16, p. 867028, Jun. 2022, doi: 10.3389/fnbot.2022.867028.

[5] F. Leiva, N. Cruz, I. Bugueño, and J. Ruiz-del-Solar, "Playing Soccer without Colors in the SPL: A Convolutional Neural Network Approach." *arXiv*, Nov. 29, 2018. doi: 10.48550/arXiv.1811.12493.

[6] A. C. Nugraha, M. L. Hakim, S. Yatmono, and M. Khairudin, "Development of Ball Detection System with YOLOv3 in a Humanoid Soccer Robot," *Journal of Physics: Conference Series*, vol. 2111, no. 1, p. 012055, Nov. 2021, doi: 10.1088/1742-6596/2111/1/012055.

[7] S. Chatterjee, F. H. Zunjani, and G. C. Nandi, "Real-Time Object Detection and Recognition on Low-Compute Humanoid Robots using Deep Learning," in *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, Singapore, Apr. 2020, pp. 202–208. doi: 10.1109/ICCAR49639.2020.9108054.

[8] J. Hu, B. Yan, J. Wang, and P. Sun, "Humanoid Soccer Robot Target Detection and Localization," in *Proceedings of the 2023 International Joint Conference on Robotics and Artificial Intelligence*, Shanghai China: ACM, Jul. 2023, pp. 65–69. doi: 10.1145/3632971.3632992.

[9] Y. Hayashibara *et al.*, "RoboCup2022 KidSize League Winner CIT Brains: Open Platform Hardware SUSTAINA-OP and Software," in *RoboCup 2022: Robot World Cup XXV*, vol. 13561, pp. 215–227, 2023, doi: 10.1007/978-3-031-28469-4_18.

[10] S. K. Narayanaswami *et al.*, "Towards a Real-Time, Low-Resource, End-to-End Object Detection Pipeline for Robot Soccer," in *RoboCup 2022: Robot World Cup XXV*, Berlin, Heidelberg: Springer-Verlag, vol 13561, pp. 62–74, Mar. 2023, doi: 10.1007/978-3-031-28469-4_6.

[11] D. D. R. Meneghetti, T. P. D. Homem, J. H. R. De Oliveira, I. J. D. Silva, D. H. Perico, and R. A. D. C. Bianchi, "Detecting Soccer Balls with Reduced Neural Networks: A Comparison of Multiple Architectures Under Constrained Hardware Scenarios," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 3, p. 53, Mar. 2021, doi: 10.1007/s10846-021-01336-y.

[12] S. Susanto, F. A. Putra, and R. Analia, "XNOR-YOLO: The High Precision of The Ball and Goal Detecting on The Barelang-FC Robot Soccer," in *2020 3rd International Conference on Applied Engineering (ICAE)*, Batam, Indonesia, pp. 1–5, Oct. 2020, doi: 10.1109/ICAE50557.2020.9350386.

[13] J. Hagge, "Deep Active Learning for Object Detection in RoboCup Soccer," *University of Hamburg*, Hamburg, Jerman, 2021.

[14] M. Szemenyei and V. Estivill-Castro, "Fully neural object detection solutions for robot soccer," *Neural Computing and Applications*, vol. 34, no. 24, pp. 21419–21432, Dec. 2022, doi: 10.1007/s00521-021-05972-1.

[15] C. O. Yinka-Banjo, O. A. Ugot, and E. Ehiorobo, "Object detection for robot coordination in robotics soccer," *Nigerian Journal of Technological Development*, vol. 19, no. 2, pp. 136–142, Aug. 2022, doi: 10.4314/njtd.v19i2.5.

[16] M. Bestmann *et al.*, "TORSO-21 Dataset: Typical Objects in RoboCup Soccer 2021," in *RoboCup 2021: Robot World Cup XXIV*, vol. 13132, pp. 65–77, 2022, doi: 10.1007/978-3-030-98682-7_6.

[17] E. R. Jamzuri *et al.*, "Barelang FC - Team Description Paper Humanoid Kid-Size League RoboCup 2023 France," 2023, doi: 10.13140/RG.2.2.30653.23527.

[18] T. Werner, "Exploring the Effectiveness of Object Detection Training in Virtual Environments," *Bachelor Informatica University of Amsterdam*, Jun. 2023.
[19] D. Pavlichenko *et al.*, "RoboCup 2022 AdultSize Winner NimbRo: Upgraded Perception, Capture Steps Gait and Phase-Based In-Walk Kicks," in *RoboCup 2022: Robot World Cup XXV*, vol. 13561, pp. 240–252, 2023, doi: 10.1007/978-3-031-28469-4_20.
[20] D. Rodriguez *et al.*, "RoboCup 2019 AdultSize Winner NimbRo: Deep Learning Perception, In-Walk Kick, Push Recovery, and Team Play Capabilities," *RoboCup 2019, Robot World Cup XXIII*, vol. 11531, pp. 631-645, 2019, doi: 10.1007/978-3-030-35699-6.
[21] A. Tripathi, M. K. Gupta, C. Srivastava, P. Dixit, and S. K. Pandey, "Object Detection using YOLO: A Survey," in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*, Uttar Pradesh, India, pp. 747–752, 2022, doi: 10.1109/IC3I56241.2022.10073281.
[22] M. Hussain, "YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection," *Machines*, vol. 11, no. 7, p. 677, 2023, doi: 10.3390/machines11070677.
[23] 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, 2023, doi: 10.3390/make5040083.
[24] Z. Xiong, "A Design of Bare Printed Circuit Board Defect Detection System Based on YOLOv8," *Highlights in Science, Engineering and Technology*, vol. 57, pp. 203–209, 2023, doi: 10.54097/hset.v57i.10002.
[25] F. Xu, B. Li, and S. Xu, "Accurate and Rapid Localization of Tea Bud Leaf Picking Point Based on YOLOv8," in *Big Data and Social Computing*, vol. 1846, pp. 261–274, 2023, doi: 10.1007/978-981-99-3925-1_17.
[26] 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)*, Vancouver, BC, Canada, pp. 7464–7475, 2023, doi: 10.1109/CVPR52729.2023.00721.
[27] Susanto, E. Rudiawan, R. Analia, P. Daniel Sutopo, and H. Soebakti, "The deep learning development for real-time ball and goal detection of barelang-FC," in *2017 International Electronics Symposium on Engineering Technology and Applications (IES-ETA)*, Surabaya, pp. 146–151, 2017, doi: 10.1109/ELECSYM.2017.8240393.
[28] F. Zhengxin *et al.*, "MLOps Spanning Whole Machine Learning Life Cycle: A Survey." arXiv, 2023. doi: 10.48550/arXiv.2304.07296.
[29] Z. Xiao, Y. Zhu, Y. Chen, B. Y. Zhao, J. Jiang, and H. Zheng, "Addressing Training Bias via Automated Image Annotation." *arXiv*, 2018, doi: 10.48550/arXiv.1809.10242.
[30] J. Tian, Q. Jin, Y. Wang, J. Yang, S. Zhang, and D. Sun, "Performance analysis of deep learning-based object detection algorithms on COCO benchmark: a comparative study," *Journal of Engineering and Applied Science*, vol. 71, no. 1, p. 76, 2024, doi: 10.1186/s44147-024-00411-z.
[31] P. Števuliáková and P. Hurtik, "Intersection over Union with smoothing for bounding box regression." *arXiv*, 2023, doi: 10.48550/arXiv.2303.15067.
[32] T. Shehzadi, K. A. Hashmi, D. Stricker, M. Liwicki, and M. Z. Afzal, "Bridging the Performance Gap between DETR and R-CNN for Graphical Object Detection in Document Images," *arXiv*, 2023, doi: 10.48550/arXiv.2306.13526.
[33] M. Szemenyei and V. Estivill-Castro, "ROBO: Robust, Fully Neural Object Detection for Robot Soccer," in *RoboCup 2019: Robot World Cup XXIII*, vol. 11531, pp. 309–322, 2019, doi: 10.1007/978-3-030-35699-6_24.

BIOGRAPHIES OF AUTHORS

Handaru Jati received the Ph.D. degree in Departement of Computer and Information Science from Universiti Teknologi Petronas, Malaysia. He is currently a Assoc. Professor at Universitas negeri Yogyakarta, Department of Electronic and Informatics Engineering Education. His current research interests include machine learning, artificial intelligent, decision support system, data mining, software development, and vocational education. He can be contacted at email: handaru@uny.ac.id.

Nur Alif Ilyasa a graduate of the Bachelor of Information Technology program at Universitas Negeri Yogyakarta, distinguishes himself as a proficient software engineer with a fervent interest in artificial intelligence, computer vision, and robotics. A dedicated member of the Al-'Aadiyaat Humanoid Robot Soccer Team, he has contributed his skills and expertise to the UNY robotics team's humanoid robot soccer software for an impressive three years. He can be contacted at email: parasyst@gmail.com.

Yuniar Indrihapsari is currently working as a teacher at Universitas Negeri Yogyakarta, Indonesia and a Ph.D. student at National Taiwan University of Science and Technology, Taiwan. Her research interests include social network analysis, e-learning, and human-computer interaction. She can be contacted at email: yuniar@uny.ac.id.

Ariadie Chandra is a lecturer in the Department of Electrical Engineering Education at Universitas Negeri Yogyakarta. Renowned as an expert in learning media and robotics, he further extends his influence as a supervisor for the UNY robotics team. With an illustrious career spanning five years, he has been actively engaged in impactful research, particularly focusing on the humanoid robot soccer initiatives undertaken by the UNY robotics team. He can be contacted at email: ariadie@uny.ac.id.

Dhanapal Durai Dominic works as a Assoc. Prof. at Universiti Teknologi PETRONAS, Department of Computer and Information Sciences. His research interest is management information system, management information systems, decision support system, e-Business, and data analytics. He can be contacted at email: dhanapal_d@utp.edu.my.