Basketball Board Detection Using YOLO Algorithms

Juozas Širmenis[1], Mantas Lukoševičius[1]

[1]*Faculty of Informatics, Kaunas University of Technology, Kaunas, Lithuania*

Abstract

This research aims to detect basketball board, rim, net, and shooting box using different versions of the YOLO algorithm: YOLOv5, YOLOv7, and YOLOv9. The study found that the YOLOv9-C model provided the best performance with a precision of 0.996, recall of 1, mAP_0.5 of 0.995, and mAP_0.5-0.95 of 0.899. YOLOv7 demonstrated the fastest inference time for the detection task and the smallest model size. Meanwhile, YOLOv5 exhibited architectural flexibility and the potential to be the optimal model in terms of detection speed, model size, and precision.

Keywords

YOLO Algorithm, Basketball Board Detection, Rim Detection, Net Detection, Shooting Box Detection, Machine Learning, Computer Vision

1. Introduction

Object detection and recognition are getting more and more powerful due to the popularity of smartphones, public cameras, AI solutions, and other algorithms that our devices are more capable of computing than any time before.

People try different solutions to detect objects, for example, some of them use sensors to detect and recognize the habits of different living species [1], others recognize the activities of the people [2], and some people use Computer Vision to detect hands and other objects [3], and some try different AI solutions for it [4, 5].

Detection of objects in sports is ever widely employed. The analysis conducted by AI can highly impact the performance of the athlete or the team. The use of AI in the sports field can be very wide: from healthcare, and skill improvement, to recognition of the goal in soccer or automatically counting shots in basketball [6, 7].

The final goal of our project is to create a model that counts free throws made by basketball players from a single static camera. Some of the most important parts to achieve this goal are the recognition of the basketball board and the rim, and it may also be useful to detect the shooting box and the net.

In this research, we apply one of the most popular machine learning models for detection tasks - You Only Look Once (YOLO). We conduct experiments to find out which version of the model and with what hyper-parameters performs the best in terms of precision, detection speed, and size of the model.

IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania
juozas.sirmenis@[ktu.edu,gmail.com] (J. Širmenis); mantas.lukosevicius@ktu.edu (M. Lukoševičius)
https://mantas.info/ (M. Lukoševičius)
0000-0001-7963-285X (M. Lukoševičius)

2. Image Processing Analysis

In this section, we will discuss the different computer vision algorithms for simplicity of calculation, noise reduction, and edge detection, different object recognition. In addition, mentioned the different solutions from a machine learning perspective.

2.1. Computer Vision Algorithms

This research is a follow-up to a previous study that used computer vision algorithms to identify and detect a basketball ball and board [8]. Initially, images were converted to grayscale to simplify calculations [9]. Noise reduction was then applied using median filtering [10], Gaussian blur [11], and morphological operations [12]. Canny edge detection [13] followed by Hough Lines [14] was used to identify lines, which were then analyzed to detect the basketball board. However, the results were disappointing in terms of speed and accuracy. Consequently, the use of more advanced machine learning algorithms is being considered.

2.2. Machine Learning Algorithm - YOLO

The YOLO (You Only Look Once) algorithm takes a different approach to object detection compared to traditional methods. Instead of repurposing classifiers, YOLO treats object detection as a regression problem, allowing for accurate localization of bounding boxes and associated class probabilities. The key innovation lies in using a single neural network, which enables end-to-end optimization and directly improves detection performance [15].

YOLO stands out for its simplicity. It employs a single convolutional network that efficiently predicts multiple bounding boxes and their corresponding class probabilities. Unlike conventional approaches, YOLO trains on complete images and directly optimizes the detection performance. This unified model offers several advantages over traditional methods in object detection.

To process images using YOLO, a straightforward three-step procedure is followed: the input image is resized, a single convolutional network is applied to the image, and the resulting detections are filtered based on the model's confidence threshold [15].

Since its introduction in 2016, the YOLO (You Only Look Once) algorithm has spawned several variations and improved versions. Notable mutations of YOLO include YOLOv3 [16] and YOLOv4 [17], which introduced enhancements such as multi-scale predictions, feature extraction improvements, and advanced architectural modifications.

These iterations of the YOLO algorithm have made significant advancements in real-time object detection, resulting in increased accuracy and improved speed. One notable variant is YOLOv5. To gain a deeper understanding of the capabilities of YOLOv5, further investigation and analysis will be conducted [18].

The evolution and mutation of YOLO demonstrate the continuous efforts to enhance object detection performance, adapt to different application scenarios, and optimize the trade-offs between speed and accuracy. Another notable mentions include YOLOv7 [19] and YOLOv8 [20], which further contributed to the advancements in the YOLO series.

Another approach of YOLO, a more optimized version, is Fast YOLO. It is designed to be faster and more efficient, and can be used in embedded systems and other devices with limited computational resources. It uses a faster CNN model, and other techniques to accelerate the object detection process.

YOLO were used to experiment with Pascal VOC dataset and the 17.85 FPS speed of object detection was achieved [21].

At the time of this study, a new YOLO algorithm method, YOLOv9, was developed [22]. It outperforms other models compared to the MS COCO dataset in terms of model size and accuracy [23]. However, it is not yet finalized because smaller versions of the model are not published, although they are mentioned, and users are not free to change the model architecture.

2.2.1. Description of YOLOv5

YOLOv5, an advanced YOLO variant, has refined the architecture and introduced efficient designs, advanced techniques, and a simplified implementation. This evolution has led to YOLOv5's popularity for its balance between accuracy and speed, making it suitable for real-time object detection tasks. Its ability to balance precise object detection and real-time processing positions YOLOv5 as a preferred algorithm for various real-world applications [24].

There are different variants of YOLOv5: small, medium, large, and extra-large are available. These variants differ in their model sizes and complexity, with larger versions offering potentially higher accuracy at the cost of increased computational demands [25].

For this work, the architecture of YOLOv5 created by Ultralytics was taken from the Roboflow template. It consists of three main components: anchors, backbone, and head [25]:

- **Anchors**: These are predefined bounding box sizes that are used for object detection. YOLOv5s has three sets of anchors, each associated with a specific feature map level (P3, P4, P5) in the model.
- **Backbone**: This part of the architecture extracts features from the input image. It consists of several convolutional layers, including Focus, Conv, and BottleneckCSP modules. These layers process the input image at different resolutions to capture hierarchical features.
- **Head**: The head of the model takes the features from the backbone and performs further processing. It includes additional Conv, Upsample, Concat, and BottleneckCSP modules. The head combines features from different levels (P3, P4) of the backbone using concatenation. It also performs upsampling and additional convolution operations. Finally, it utilizes the Detect module to generate the final detection output, which includes bounding box coordinates and class predictions.

2.2.2. Description of YOLOv7

YOLOv7 is a real-time object detection algorithm built on convolutional neural networks (CNNs). It uses convolutional layers to extract features from images, then applies bounding box regression and class prediction to localize and classify objects. YOLOv7's strength lies in its speed and real-time processing [19].

In terms of accuracy, YOLOv7 exhibits competitive performance. It achieves a high average precision (AP) score of 56.8% on the challenging MS COCO dataset, demonstrating its ability to accurately detect and classify objects across a wide range of categories. This dataset includes a diverse range of objects, making YOLOv7 adaptable to different scenarios and domains [19].

In addition to its architecture and performance, YOLOv7 introduces innovative techniques to improve accuracy during training. It incorporates a ”trainable bag-of-freebies” approach, which combines various state-of-the-art training strategies and modules to enhance the detection capabilities of the network. This approach addresses challenges in network training and object detection, contributing to the improved accuracy and robustness of YOLOv7 [19].

The architecture of YOLOv7 algorithm looks almost the same as YOLOv5 but has many more parts in backbone and head. The algorithm shares common components, including anchors, backbone, and head. However, it is intentionally designed to be wider, deeper, and overall more complex in order to enhance its performance [26].

2.2.3. Description of YOLOv9

YOLOv9 is a significant advancement in the field of object detection. It introduces two key concepts: Programmable Gradient Information (PGI) and a new lightweight network architecture, the Generalized Efficient Layer Aggregation Network (GELAN) [22].

PGI is designed to address the information bottleneck problem, a common issue in deep networks where input data loses information during the feedforward process. PGI generates reliable gradients through an auxiliary reversible branch, allowing deep features to maintain key characteristics for executing the target task. This ensures that the model can obtain complete input information for the target task, enabling reliable gradient information for network weight updates. GELAN, on the other hand, is a highly efficient and lightweight neural network. It uses conventional convolution to achieve higher parameter utilization than depth-wise convolution designs based on advanced technology. GELAN shows great advantages in terms of being light, fast, and accurate [22].

When combined, PGI and GELAN enhance the object detection performance of YOLOv9. The model outperforms existing real-time object detectors in all aspects. Specifically, compared to its predecessor, YOLOv8, YOLOv9 reduces the number of parameters by 49% and the amount of calculations by 43%, while still achieving a 0.6% AP improvement on the MS COCO dataset. This demonstrates that YOLOv9 is not only more efficient but also more accurate [22].

The architecture of the YOLOv9 is similar to older versions. Common elements of the algorithm are the anchors, backbone, and head. Although, the head, in the YOLOv9, is has more elements and is more complex [23].

3. Dataset

For the experiments, self-recorded free basketball shots were captured. These shots were taken with different levels of accuracy (0, 1, 2, 3 out of 3) in a single video using two different mobile devices: the Samsung Galaxy J5 and the Samsung Galaxy A52s. In total, 153 recordings were made with these specifications. The camera position is shown on the left side in Figure 1 (marked in red) was set at approximately 45 degrees, with a variation of approximately 10 degrees, relative to the basketball’s location. A different spot was chosen for each video, ensuring variations in the shooting perspective.

The dataset includes pictures with a clearly visible basketball. The labeling process involved marking the basketball board, rim, net, and shooting box. We used the Roboflow tool to label the

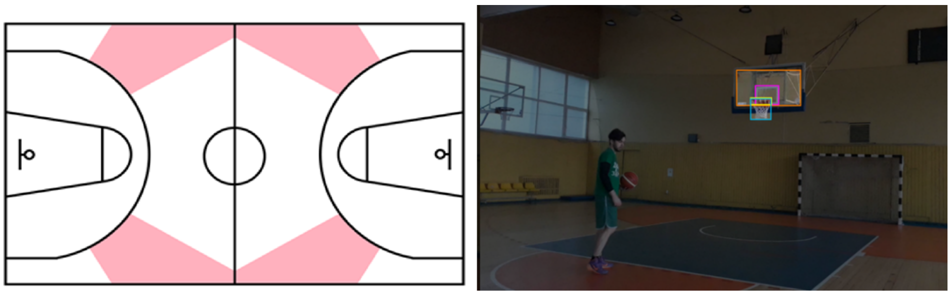
Figure 1: Positions of the camera in the basketball court (left) and Example of labeled data for board detection (right)

dataset, and the example of the labeled data on the website is shown on the right side in Figure 1. The dataset can be accessible publicly: https://app.roboflow.com/ktu-bo8mo/magister-yhpnv. Data is resized to 640x640.

4. Experiments

For the detection of basketball boards, net, rim, and shooting boxes, the three algorithms described in Section 2.2 were selected: YOLOv5, YOLOv7, and YOLOv9 (C and E).

4.1. Experiment setup

The experiments of three models conducted in the Google Colab using Python and implementing models from libraries or from the source code directly. In each experiment, the virtual GPU was used. For the hyper-parameter tuning, the three main parameters were chosen:

- **Image Size**: It was changed to observe the influence of different dimensions of the input images on detection accuracy and speed.
- **Batch Size**: The number of images processed simultaneously was adjusted to assess its impact on training efficiency and model performance.
- **Epochs**: The model was trained for a specified number of iterations to evaluate how the YOLO algorithm evolves and improves over time.

The depth and width of the YOLOv5 model were adjusted, while for the other models, they were kept at the default (1) settings. During training, pre-trained weights of YOLOv7 and YOLOv9 were used.

4.2. Evaluation metrics

In order to obtain preliminary results for the evaluation of the board detection, four main indicators of the recognition problem were selected and computed on a validation set with an image size of 640x640 (in addition to these metrics, we will take a look at the detection inference time and the model size itself):

- **Precision**: the true positives divided by the sum of true positives and false positives.
- **Recall**: the true positives divided by the sum of true positives and false positives.
- **mAP_0.5**: Mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.5. This metric calculates the average precision at a specific IoU threshold. A prediction is considered correct if its overlap with the ground truth is more than 50%. It gives us an idea of the model's performance at a fairly lenient threshold.
- **mAP_0.5-0.95**: mAP at IoU thresholds ranging from 0.5 to 0.95. This metric calculates the average precision at multiple IoU thresholds. It gives us a more comprehensive view of the model's performance across various levels of strictness in the overlap between the predicted and actual bounding boxes.

5. Results

The YOLOv5 models were trained for 200 epochs, with the numbers in their names indicating the depth and width multipliers. The YOLOv7 models were trained for 125 epochs. For YOLOv9, both C and E versions were trained for 20 epochs, with the top three results and their hyper-parameter configurations presented in the Table 1.

Table 1
Results of models for validation set with hyper-parameters over the epochs

Model (YOLO)	Image Size	Batch Size	Size (MB)	Inference Time (ms)	Precision	Recall	mAP 0.5	mAP 0.5:0.95
v5-1/1	512	32	95.4	37.7	0.979	0.989	0.995	0.774
v5-0.4/0.9	544	32	49.4	21.4	0.992	0.998	0.995	0.823
v5-0.6/0.9	576	32	61.9	24.4	0.995	0.994	0.995	0.824
v5-0.6/0.9	608	32	61.9	24.6	0.991	1	0.995	0.84
v5-1/1	640	26	95.4	31.3	0.993	0.996	0.995	0.849
v7	512	32	74.8	15.4	0.989	0.996	0.995	0.775
v7	544	32	74.8	15.6	0.991	1	0.995	0.753
v7	576	26	74.8	15.3	0.990	1	0.995	0.765
v7	608	26	74.8	15.8	0.997	1	0.995	0.811
v7	640	24	74.8	15.8	0.993	1	0.995	0.793
v9-C	544	8	102.8	55.3	0.988	0.955	0.993	0.802
v9-C	640	8	102.8	59.2	0.964	0.998	0.994	0.863
v9-C	736	6	102.8	59.1	0.996	1	0.995	0.899
v9-E	640	8	140.0	73.5	0.991	0.993	0.995	0.874
v9-E	736	6	140.0	73.1	0.997	1	0.995	0.887
v9-E	832	5	140.0	70.3	0.997	1	0.995	0.893

The table presents an evaluation of various YOLO model variants on a validation set, highlighting key performance metrics and hyperparameters. YOLOv9-E models are the largest and slowest, with inference times up to 73.5 ms, but offer one of the highest mAP_0.5:0.95 values, reaching up to 0.893. YOLOv9-C models, slightly smaller and faster, excel in accuracy with high precision and recall, achieving the best mAP_0.5:0.95 value of 0.899. YOLOv7 models, the fastest with inference times around 15.6 ms, balance speed and performance, showing high precision and recall with notable mAP_0.5:0.95 values. YOLOv5 models, particularly the v5-1/1 at 640 image size, offer a good trade-off between speed and accuracy, achieving a mAP_0.5:0.95 of 0.849 with a moderate inference time.

The results are as expected — larger training image sizes lead to higher precision and recall, although the difference is not substantial. Interestingly, increasing the image size, even beyond the original, improved all metrics. The other metrics remained similar, except the mAP_0.5:0.95 score, which was highest at 736x736 using YOLOv9-C.

The graph in Figure 2 illustrates the comparison of YOLO models based on training image size and mAP_0.5-0.95 score. The blue curve - YOLOv5 model, the red - YOLOv7, and the yellow/orange - two versions of YOLOv9.

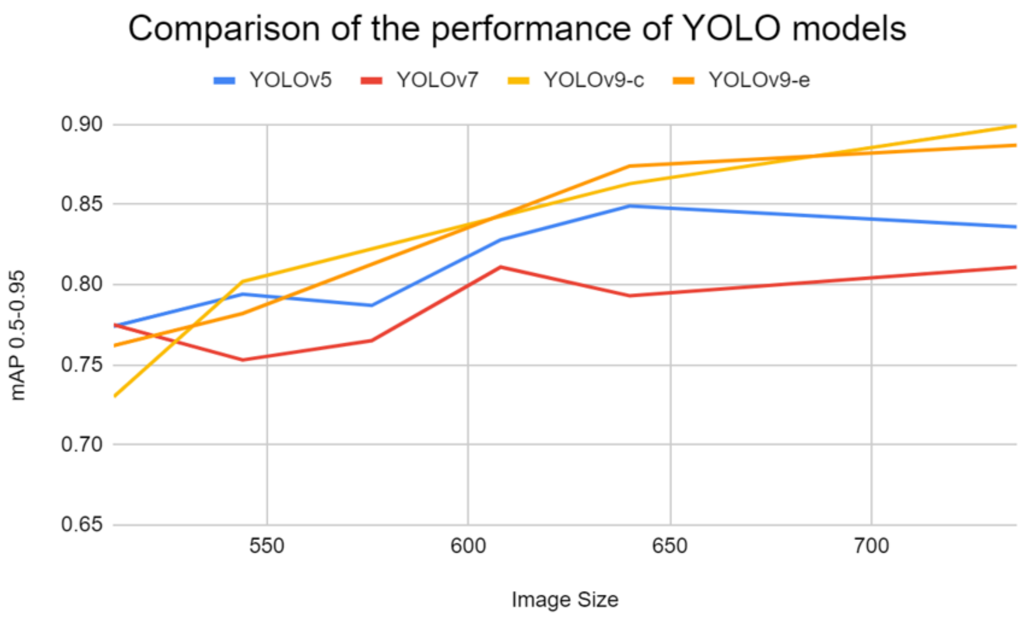

Figure 2: Comparison of the performance of YOLO models

The YOLOv9 model demonstrated superior performance compared to other models in terms of mAP_0.5-0.95 score. The difference between YOLOv9 models C and E versions is minimal. Meanwhile, YOLOv5 and YOLOv7 exhibited similar performance, with YOLOv5 slightly edging out the YOLOv7.

The Figure 3 shows the results of best performances of mAP_0.5-0.95 score for each model and different architecture.

Among the YOLO models, YOLOv9 stands out as the most precise, albeit the slowest and biggest. In contrast, YOLOv7 is the fastest but sacrifices precision when the default depth and width are set to 1. The results from YOLOv5 demonstrate that optimal configurations can be achieved through architectural experimentation.

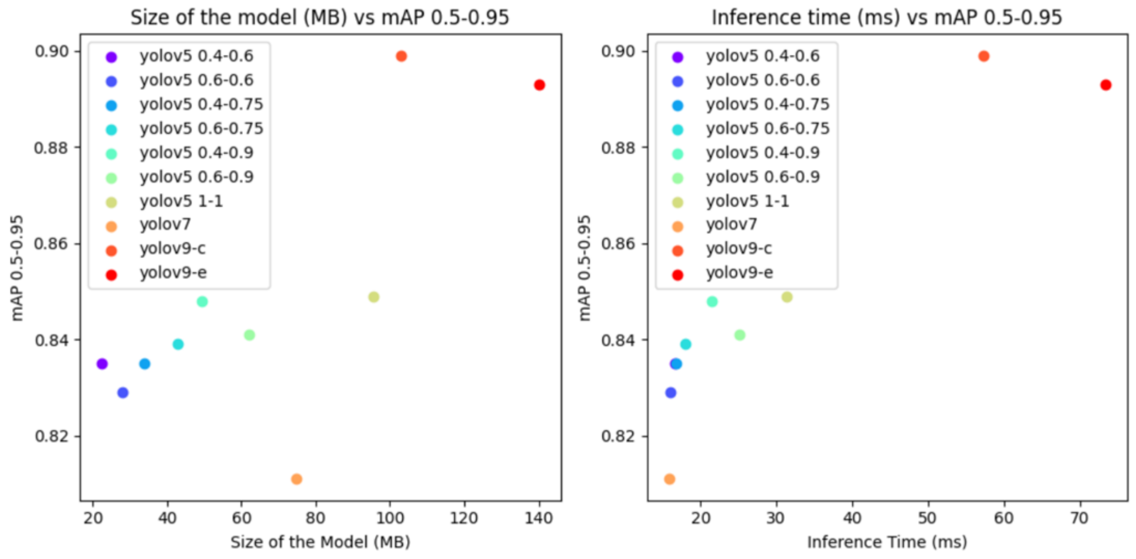

Figure 3: Relationship Between Model Size and Inference Time for mAP_0.5-0.95 Score of Board Detection

6. Conclusions

In this research, it was found that for the detection of basketball boards, rims, nets, and shooting boxes, the best performance was achieved with the YOLOv9-C model, which yielded a precision of 0.996, recall of 1, mAP_0.5 of 0.995, and mAP_0.5-0.95 of 0.899. Additionally, YOLOv9 models were noted for their rapid training with a low number of epochs when using pre-trained weights.

Other models, YOLOv5 and YOLOv7, also achieved satisfactory evaluation metrics. The most significant difference was observed in mAP_0.5-0.95, where YOLOv5 outperformed YOLOv7 at the same training image size. However, YOLOv7, with its smaller size (depth and width equal to 1 for both models), was the fastest.

Varying the depth and width of the YOLOv5 model showed that the results do not always decrease with smaller model size. In addition, the detection time of the model also decreases.

It was found that increasing the size of the training image improves model performance, leading to the conclusion that, if time and resources permit, training should be conducted with the largest possible image size, even if it exceeds the size of the source images.

Future work will incorporate more data from various publicly available sources and experiment with different models. A key focus will be optimizing YOLO models for this task by testing different architectures (varying depths and widths), as well as different optimizers, learning rates, and activation functions. Additionally, it would be beneficial to test on different basketball courts not included in the training and validation set and to analyze results from video files.

The ultimate goal is to develop a free-throw counting program for basketball enthusiasts, which will involve recognizing a moving ball in combination with the results of this study.

References

[1] J. T. Kerr, M. Ostrovsky, From space to species: ecological applications for remote sensing, Trends in ecology & evolution 18 (2003) 299–305.

[2] K. Li, W. Yang, M. Yi, Z. Shen, Graphene-based pressure sensor and strain sensor for detecting human activities, Smart Materials and Structures 30 (2021) 085027.

[3] T. Starner, B. Leibe, D. Minnen, T. Westyn, A. Hurst, J. Weeks, The perceptive workbench: Computer-vision-based gesture tracking, object tracking, and 3d reconstruction for augmented desks, Machine Vision and Applications 14 (2003) 59–71.

[4] P. Wong, Identifying table tennis balls from real match scenes using image processing and artificial intelligence techniques, International Journal of Simulation Systems, Science & Technology 10 (2009) 6–14.

[5] Y.-H. Lee, Y. Kim, Comparison of cnn and yolo for object detection, Journal of the semiconductor & display technology 19 (2020) 85–92.

[6] P. Liu, N. Yue, J. Chen, A machine-learning-based medical imaging fast recognition of injury mechanism for athletes of winter sports, Frontiers in public health 10 (2022) 842452.

[7] M. Barth, A. Güllich, C. Raschner, E. Emrich, The path to international medals: A supervised machine learning approach to explore the impact of coach-led sport-specific and non-specific practice, PLoS One 15 (2020) e0239378.

[8] J. Širmenis, M. Lukoševičius, Tracking basketball shots–preliminary results, in: CEUR workshop proceedings: IVUS 2021: Information society and university studies 2021: Proceedings of the 26th international conference on information society and university studies (IVUS 2021), Kaunas, Lithuania, April 23, 2021, volume 2915, CEUR-WS, 2021, pp. 181–190.

[9] C. Saravanan, Color image to grayscale image conversion, in: 2010 Second International Conference on Computer Engineering and Applications, volume 2, IEEE, 2010, pp. 196–199.

[10] Z. Pei, Q. Tong, L. Wang, J. Zhang, A median filter method for image noise variance estimation, in: 2010 Second International Conference on Information Technology and Computer Science, IEEE, 2010, pp. 13–16.

[11] B. Antoni, C. Bartomeu, J. Morel, On image denoising methods, CMLA Preprint, CMLA 15 (2004) 2004.

[12] B. Chakraborty, S. Meher, A real-time trajectory-based ball detection-and-tracking framework for basketball video, Journal of Optics 42 (2013) 156–170.

[13] Opencv: Canny edge detection, https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html, 2024. (Accessed on 02/05/2024).

[14] Opencv: Feature detection, https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d, 2024. (Accessed on 02/05/2024).

[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779–788.

[16] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).

[17] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, arXiv preprint arXiv:2004.10934 (2020).

[18] ultralytics/yolov5: Yolov5 in pytorch > onnx > coreml > tflite, https://github.com/ultralytics/yolov5, 2024. (Accessed on 02/05/2024).

[19] C.-Y. Wang, A. Bochkovskiy, H.-Y. M. Liao, Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 7464–7475.

[20] J. Terven, D. Cordova-Esparza, A comprehensive review of yolo: From yolov1 to yolov8 and beyond, arXiv preprint arXiv:2304.00501 (2023).

[21] M. J. Shafiee, B. Chywl, F. Li, A. Wong, Fast yolo: A fast you only look once system for real-time embedded object detection in video, arXiv preprint arXiv:1709.05943 (2017).

[22] C.-Y. Wang, I.-H. Yeh, H.-Y. M. Liao, Yolov9: Learning what you want to learn using programmable gradient information, arXiv preprint arXiv:2402.13616 (2024).

[23] W. Kin-Yiu, Wongkinyiu/yolov9: Implementation of paper - yolov9: Learning what you want to learn using programmable gradient information, https://github.com/WongKinYiu/yolov9, 2024. (Accessed on 03/25/2024).

[24] Yolov5 - ultralytics | revolutionizing the world of vision ai, https://ultralytics.com/yolov5, 2023. (Accessed on 05/28/2023).

[25] Yolov5 object detection model, https://roboflow.com/model/yolov5, 2023. (Accessed on 05/28/2023).

[26] Wongkinyiu/yolov7: Implementation of paper - yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, https://github.com/WongKinYiu/yolov7, 2024. (Accessed on 02/05/2024).