Lightweight YOLOX-Based Object Detection with Structural Pruning for Edge Device Deployment

Mikihisa Ishino[1], Lin Meng[2,*]

[1]Department of Electronic and Computer Engineering, Ritsumeikan University, 1-1-1 Noji-higashi, Kusatsu, Shiga, 525-8577, Japan

[2]College of Science and Engineering, Ritsumeikan University, 1-1-1 Noji-higashi, Kusatsu, Shiga, 525-8577, Japan

Abstract

Research on object detection using deep learning has achieved remarkable accuracy. However, real-time performance remains a significant challenge on edge devices with limited computational resources, posing a substantial hurdle for practical implementation. Despite significant advancements in the YOLO series, including improving accuracy, speed, and capabilities for detecting small objects and moving towards anchor-free designs, the resulting models remain computationally demanding for resource-constrained edge devices, particularly those limited to a CPU. Alternatively, while reducing model complexity, for instance, by decreasing the number of layers, can shorten computation time, this approach often compromises accuracy, rendering the model impractical for real-world applications. To address the limitations of object detection under constrained computational resources, our paper proposes an efficient object detection model designed for deployment on edge device. The proposed model is based on YOLOX and applies structural pruning to its backbone, which typically has a large number of parameters, to reduce the number of parameters per channel. The proposal is achieved by identifying and removing less important channels based on the scale factor (γ) of the batch normalization layers. Specifically, to improve inference speed under limited resources, we applied iterative structural pruning to the YOLOX backbone, reducing the number of parameters while maintaining inference accuracy. To demonstrate the effectiveness of the proposed model, we create a unique dataset consisting of plastic bottles, cans, and glass bottles for recycling purposes and conduct comprehensive experiments. The experimental results indicate that significant computational reduction is achieved by iterative structural pruning, demonstrating the effectiveness of our model. The work represents a significant step towards enabling high-performance object detection on edge devices with limited computational resources.

Keywords

Deep Learning, YOLOX, pruning, Object Detection, Edge Device

1. Introduction

The concept of object detection emerges with the advent of digital cameras, initially aiming to automatically detect objects and adjust brightness and focus. The idea gains significant societal demand because there is a possibility to solve challenges across diverse fields like autonomous driving, medicine, surveillance, retail, and agriculture, leading to a surge in research.

The 7th International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT 2025), August 10-11, 2025, Kusatsu, Shiga, Japan

*Corresponding author.

ri0135ve@ed.ritsumei.ac.jp (M. Ishino); menglin@fc.ritsumei.ac.jp (L. Meng)

0000-0003-4351-6923 (L. Meng)

Coupled with the rise of deep learning in 2012, object detection witnesses a remarkable leap in accuracy through the application of image recognition models like R-CNN. However, detection speed remains a bottleneck for applications requiring real-time performance. This challenge is addressed in 2015 with the introduction of YOLO, which boasts exceptional processing speed. YOLO's groundbreaking performance profoundly impacts the field of object detection, raising expectations for real-time applications and shaping the research landscape to this day.

Throughout the evolution, the YOLO series has seen a variety of enhancements aimed at improving accuracy, speed, and the number of detectable categories[1, 2, 3]. The evolution includes better small-object detection and anchor-box-independent detection. Despite the strides, YOLO series is too slow for real-world application on edge devices with limited computational power (such as those relying solely on a CPU) because the real situation is changing quickly. Although reducing deep learning layers can cut down on processing time for resource-limited environments, the accuracy decreases, making it hard to develop a practical solution.

To address the limitations of object detection on restricted computational resources, our paper proposes an improved object detection model for edge device implementation. The proposal applies structural pruning to the backbone of YOLOX. The method reduces the number of parameters by removing unnecessary channels. To identify which channels to remove, we use the scale factor of the BN layers.

This paper addresses the social issue of environmental degradation caused by the waste from beverage containers. We aim to develop an object detection model for edge devices that can automatically detect, sort, and collect such waste.

Overall, our main contributions can be summarized below:

- We demonstrate the successful application of iterative structural pruning to the modern, anchor-free YOLOX architecture, providing a practical methodology for deployment on resource-constrained edge devices.
- We have introduced a novel custom dataset of recycling objects (plastic bottles, cans, and glass bottles) specifically designed to address real-world object detection challenges, uniquely incorporating overlapping objects, extreme deformations, and image blur.

The remainder of the article is organized as follows. Section 2 reviews previous lightweight object detection models and related YOLO models. The detailed proposal is introduced in Section 3. Section 4 presents the experiments and the datasets. Finally, Section 5 concludes the paper.

2. Related Work

Object detection aims to classify and predict specific objects belonging to predefined categories within a dataset. With advances in deep learning, object detection is becoming increasingly practical[4]. However, a challenge has emerged: the difficulty of use under conditions with limited computational resources or memory capacity. To solve the problem, various methods have been proposed, successfully reducing the size of the model and improving the speed of inference. Among various methods, YOLO models stand out for the excellent performance, being relatively lightweight and offering a good balance with accuracy, leading to their use in diverse applications.

2.1. Lightweight models for object detection

Research into making object detection models lighter has really taken off in recent years. Approximately in 2017, object detection research is still in the early stages. During the stages, the main approach is to replace the backbone (feature extraction part) of object detectors with lightweight CNN architectures originally proposed for image classification tasks[5].

In 2017, Google's MobileNet paper[6] proposes a groundbreaking network using depth-wise convolution and point-wise convolutional layers, reducing the computation by about nine times. MobileNet-SSD[7] employs the MobileNet architecture as a backbone for the then-fast SSD object detector. The model's introduction pioneers real-time object detection on mobile devices. The model allows many developers to integrate object detection features into smartphone apps.

MobileNetV2[8], released in 2018, introduces Inverted Residuals and Linear Bottlenecks from MobileNet, leading to networks that balance higher accuracy and efficiency. Network models like ShuffleNet[9], which introduces the Channel Shuffle operation, are combined with detectors like SSD and YOLO, resulting in numerous higher-performing models.

In 2020, Google's EfficientDet[10] proposes techniques like BiFPN and Compound Scaling, marking a significant breakthrough in the history of lightweight models. The proposed scaling method provides a systematic approach for selecting an optimal model based on a device's computational power, establishing a new base for lightweight models across various fields.

In recent years, further efficiency has been achieved by fundamentally rethinking the entire detector architecture, not just the backbone. The Generalized Focal Loss, used as a loss function in NanoDet[11] released in 2021, is an innovative loss function. By integrating classification and regression, the model makes the detection part's structure significantly simple and lightweight, enabling high performance even with lightweight models.

2.2. The Evolution of the YOLO Series

The YOLO (You Only Look Once) object detection model[12] is developed with the purpose of balancing real-time performance and accuracy. Before YOLO, object detection models require multistage processing: first, finding candidate regions for objects, and then classifying each region. Although accurate, the model makes them slow and challenging to apply in real-time scenarios. YOLO integrates the candidate region search into a single neural network, simultaneously predicting object location and class. The change dramatically improves processing speed, enabling use in fields where real-time capability is crucial, such as autonomous driving, surveillance cameras, and robot vision.

Since the initial release in 2015, YOLO has undergone numerous improvements. YOLOv2[13], released in 2016, improves the accuracy of location prediction by incorporating batch normalization and anchor boxes. YOLOv3[14], released in 2018, adopts the concept of FPN (Feature Pyramid Network), predicting objects on three different scales, significantly increasing the detection accuracy of small objects.

YOLOv4[15], released in 2020, introduces the "Bag of Specials" concept, optimally combining various techniques considered effective in object detection research. The model greatly improves accuracy while maintaining the practicality of being trainable and runnable on a single GPU. YOLOv5[16] introduces flexible model scaling. The model is easier to balance speed and accuracy

depending on the device and application, thus accelerating the practical adoption. YOLOv6[17], released in 2022, improves the head section, increasing inference speed. YOLOv7[18], released in the same year, achieves state-of-the-art performance in both speed and accuracy through advances in network architecture and training methods.

YOLOX[19], introduced in 2021 (between v5 and v6), differs from previous models in the YOLO series by integrating a suite of contemporary high performance techniques, signifying a new design approach.

3. Methodology

In this study, we apply structural pruning to its backbone, CSPDarknet-53, to reduce the number of parameters. Experiments are conducted using an object detection dataset comprising plastic bottles, cans, and glass bottles.

3.1. YOLOX

YOLOX, released in July 2021, is positioned as a technologically significant model that brings a major transformation to the design philosophy of the YOLO series. While based on YOLOv3, YOLOX incorporates the state-of-the-art techniques in object detection research at the time, such as anchor-free detection and decoupled heads.

The YOLOX network is composed of three stages: backbone, neck, and head. The model's scaling factor allows for adjusting the network's width and depth, enabling the modification of detection performance and inference time based on specific objectives and devices.

The backbone network is responsible for efficiently extracting essential features from the input image for object detection. The network uses an architecture called CSPDarknet, a hybrid design combining CSPNet with the Darknet-53 framework introduced in YOLOv3. Darknet-53 is a 52-layer convolutional network with one fully connected layer. The network's main feature is a structure designed to prevent gradient vanishing even in deep networks, enabling efficient training. CSPNet's design aims to resolve computational bottlenecks within the network, allowing for richer feature extraction with lower computational cost. The integrated networks further improve the computational efficiency and gradient flow compared to the original Darknet-53.

The neck is responsible for creating an optimal set of feature maps for detection, combining both precise location information and rich semantic information. The neck incorporates PAFPN, which combines a Feature Pyramid Network (FPN) that transmits semantic information from deeper to shallower backbone layers, and a Path Aggregation Network (PANet) that feeds back localization information from shallower to deeper layers. The integration has improved detection accuracy for objects of various sizes.

The head employs a decoupled structure,a notable departure from conventional YOLOv5, that establishes distinct pathways for class classification and localization. The architectural separation allows for task-specific optimization, resulting in considerable gains in model accuracy and training efficiency.

The overall YOLOX architecture introduces an anchor-free method, eliminating the need for the previous intermediate concept of anchor boxes. Consequently, SimOTA is introduced as the

label assignment algorithm. The method represents innovative changes from previous models.

3.2. Pruning

Pruning is a technique that aims to make trained deep learning models lighter and faster by removing less important parameters. The method is highly valued, because the deployment of standard deep learning models on edge devices with limited computational resources is extremely difficult. Among pruning methods, structural pruning is particularly promising as the method can lead to practical speed-ups by removing entire structural units like filters or channels. A widely recognized and effective method for channel-level pruning is to use the scaling factor (γ) in Batch Normalization (BN) layers as an importance criterion, as proposed by Liu et al. in their work on Network Slimming [20]. The method is expected to improve the processing speed because the dense matrix structure is maintained even after pruning.

Consider a single convolutional layer in basic structural pruning.

$$I \in \mathbb{R}^{C_{in} \times H_{in} \times W_{in}} \tag{1}$$

$$W \in \mathbb{R}^{C_{out} \times C_{in} \times K_i \times K_w} \tag{2}$$

$$O \in \mathbb{R}^{C_{out} \times H_{out} \times W_{out}} \tag{3}$$

Here, I represents the input feature map, W represents the weight filter, and O represents the output feature map. H, W denote the height and width of the feature map, and K_h, K_w denote the kernel sizes. For the layer, p filters are removed through structural pruning. Pruning reduces the dimensionality of W and, consequently, the computational cost of O. The effect implies a cascading reduction not only in the computational cost of the specific layer but also in the subsequent processing. The overall reduction in computational load across the model directly leads to improved inference performance. The pruned weight W' and output O' are represented as follows:

$$W' \in \mathbb{R}^{(C_{out}-p) \times C_{in} \times K_i \times K_w} \tag{4}$$

$$O' \in \mathbb{R}^{(C_{out}-p) \times H_{out} \times W_{out}} \tag{5}$$

3.3. Improved structure

In this study, we aim to leverage powerful parameter reduction techniques and lightweight design to reduce the number of parameters, improve FLOPs and FPS compared to the standard YOLO series, and achieve better performance on edge devices with limited computational resources. The model consists of three parts: backbone, neck, and head. Figure 1 shows the backbone before pruning, and Figure 2 illustrates how channels are removed from each convolutional layer through pruning. The backbone, responsible for feature extraction, is a crucial component for the accuracy of an object detection model and is composed of numerous convolutional layers. Thus, we aim to reduce the parameter count. In the CSP layer, features are passed to the next output by separating into a path that goes through the traditional Darknet block and a path that performs only 1x1 convolutions, and then reintegrating. By applying

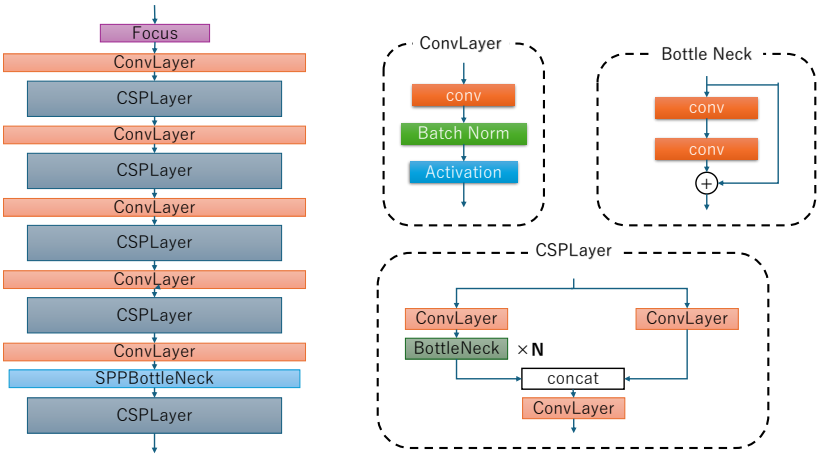

Figure 1: The general outline of the model's backbone. Structural pruning is applied to the convolutional parts in the diagram.

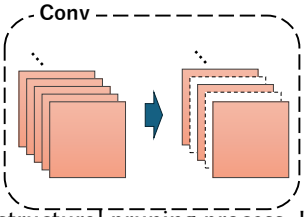

Figure 2: The general outline of the structural pruning process. Less important channels are removed from the backbone's convolutional layers.

structural pruning to the layers and reducing the number of channels, the model's parameter count can be significantly decreased, improving performance on edge devices.

In the model's neck, feature maps from deep layers and shallow feature maps are integrated to create new fused feature maps. Feature maps from deeper layers are upsampled to match the feature size of a shallower layer and then integrated. The process of extracting features using a CSP layer is repeated twice, after which downsampling is performed by a convolutional layer before being passed to the head. The head is divided into three branches, each handling detection processing for objects of specific sizes.

4. Experiment

In this section, we create a unique dataset of beverage containers for recycling. With the dataset, we conduct comprehensive experiments with models featuring backbones of varying channel counts, demonstrating the effectiveness of our proposed model.

4.1. Dataset

For the experiment, we use a custom dataset specifically created for the task of beverage container object detection. The dataset used in this study comprises three categories: plastic

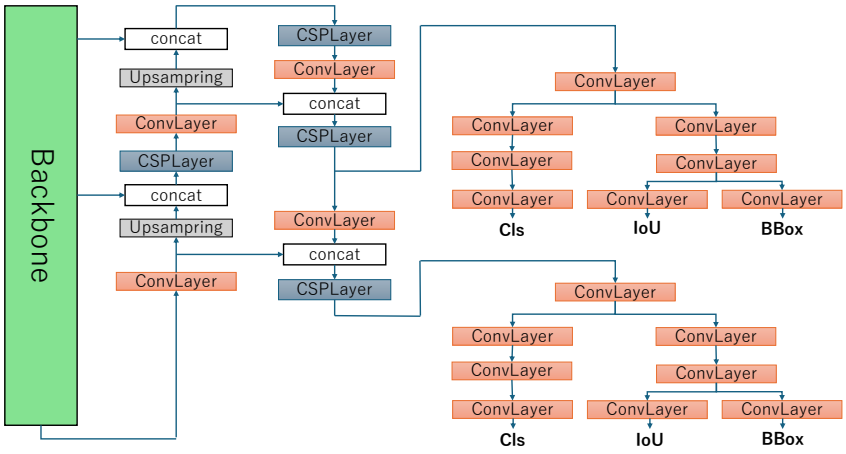

Figure 3: The overall architecture of the object detection model. The neck creates fused feature maps, and the head performs the detection.

bottles, glass bottles, and cans. To address the lack of data diversity, online image augmentation is applied to each image.

The beverage container dataset totals 720 images, with 583 images in the training set, 65 in the validation set, and 72 in the test set. We design the dataset to present three specific challenges for object detection:

1. Overlapping Objects. We include images where multiple objects overlap, assuming that scattered waste in real-world scenarios isn't always isolated.
2. Extreme Deformation: Recognizing that empty beverage containers found on roadsides aren't always in pristine condition, we include flattened plastic bottles and cans in their respective categories.
3. Image Blur. To account for operation in environments where edge devices might be in motion, we include images with blurred objects.

To demonstrate the unique characteristics of our dataset, Figure 4 shows sample images featuring (a) overlapping objects, (b) exuyytreme deformations, and (c) image blur. The dataset is not publicly available at this time.

4.2. Implementation details

All experiments are conducted on a platform equipped with an NVIDIA GeForce RTX 4090 graphics card, utilizing PyTorch 2.4.0 with CUDA 12.4 as the deep learning framework. The method proposed in this study is implemented based on a highly flexible, open-source framework.

The experiment is primarily divided into two stages:

In the first stage, the conventional YOLOX model is trained on the custom dataset. The number of epochs and batch size are set to 1000 and 8, respectively. Training is performed using the SGD optimizer, with an initial learning rate of 0.01 and a minimum learning rate set to 0.01

(a)overlapping objects (b)extreme deformation (c)image blur

Figure 4: The characteristic samples of our dataset

times the initial learning rate. Momentum is set to 0.937, and weight decay to 0.0005. A cosine annealing learning rate decay schedule is adopted. Data augmentation is applied to the dataset. Both Mosaic and Mixup data augmentations are applied with a 50% probability and set to occur during the first 70% of the training epochs.

In the second stage, structural pruning is performed on the previously trained model. The importance of each channel in the backbone's convolutional layers is evaluated using the scale factor(γ) of the BN layer immediately following each convolutional layer, removing approximately 10% of the backbone's parameters. After pruning, the created model is fine-tuned to recover accuracy. For fine-tuning, the number of epochs is set to 150, with other parameters remaining the same as in the first stage. This cycle of pruning and fine-tuning, known as iterative pruning, is repeated until the number of parameters in the backbone is less than 50% of the original size. The iterative approach is a widely adopted practice, as the approach has been shown to maintain model accuracy more effectively than a single, large pruning step [21]. The 50% reduction threshold is determined through preliminary experiments aiming to find the optimal balance between improving real-time performance on the target device and minimizing accuracy degradation. We observe that pruning beyond 50% initiated a non-negligible drop in accuracy, leading us to conclude that the threshold is the optimal trade-off point between computational cost and precision.

4.3. Evaluation metrics

In this study, in addition to typical evaluation metrics commonly used in object detection tasks, including Precision, Recall, mAP50, and mAP50-95, we also evaluate the model using metrics such as pruning rate, parameter reduction rate, FLOPs reduction rate, and FPS to assess the effectiveness of structural pruning. The definitions of these metrics are as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7}$$

$$\text{AP} = \sum_{n=1}^{N} (R_{n+1} - R_n) \text{Precision}_{\max}(R_{n+1}) \tag{8}$$

$$\text{mAP} = \frac{1}{C} \sum_{j=1}^{C} \text{AP}_j \tag{9}$$

TP represents the number of correctly predicted objects, FP represents false positives (misdetections), and FN represents false negatives (undetected objects). Precision and Recall are calculated using TP, FP and FN. AP is determined by the area under the precision-recall curve. mAP50 is obtained by averaging the AP for each class in the dataset when IoU = 0.5. mAP50-95 is obtained by averaging the mAP at different IoU thresholds from 0.5 to 0.95. FLOPs represent the computational load of the model, and FPS represents the number of images that can be processed per second.

Furthermore, to estimate the real-time performance on resource-constrained hardware, we define the estimated inference time and FPS based on the model's computational load as follows:

$$\text{Estimated Inference Time (s)} = \frac{\text{Model GFLOPs}}{\text{CPU GFLOPs} \times \text{k}_{\text{efficiency}}} \tag{10}$$

$$\text{Estimated FPS} = \frac{1}{\text{Estimated Inference Time (s)}} \tag{11}$$

Here, $\text{k}_{\text{efficiency}}$ represents the CPU efficiency factor.

4.4. Performance Comparison

To demonstrate the effectiveness of proposal using an open-source framework, we compare our pruned YOLOX model with other structurally pruned models. Table 1 shows the result. The table evaluates the object detection accuracy using the test set when structural pruning, at a pruning rate of 10% on the backbone, is applied 7 times until the backbone's parameters fall below 50%.The table indicates that despite reducing up to approximately 43% of parameters solely through backbone pruning, there is almost no change in the accuracy metrics compared to the original model. The result suggests that the iterative structural pruning effectively reduces parameters while maintaining robust feature representation.

Table 2 presents a comparison of the YOLOX and the final pruned model in terms of parameter count, channel count, computational load, and FPS. In the comparison, the computational load is calculated with an input size of 3x640x640, and the FPS is computed using the same input size in addition to a batch size of 1. The table clearly demonstrates that the proposed model significantly reduces computational load compared to the original model.

While the experiments in this study are conducted in a GPU environment, we perform a theoretical estimation to evaluate the model's performance on an edge device, which is a key objective of our paper. We select the Raspberry Pi 4 as the target device because we plan to deploy deep learning models in the future. Based on the official specifications, we assume a theoretical computational performance of 6 GFLOPS for the CPU. Assuming a commonly used CPU efficiency factor of 30% and using Equations (10) and (11), we estimate the inference time and FPS. Table 3 shows the estimated values for the edge CPU. The result suggests a potential performance improvement of approximately 71% on the target edge device, while also demonstrating the difficulty of using high-end GPUs—which showed only a 2.5% speedup—to properly evaluate performance on the edge device.

Table 1
Comparing the Original YOLOX and the Pruned YOLOX Model Using the Beverage Container Dataset

Model	class	Precision	Recall	AP	mAP50	mAP50-95
Original YOLOX-nano	glass bottle	100.0	98.63	99.98	97.76	96.29
	bottle	97.52	98.21	98.55		
	can	95.95	93.79	94.75		
YOLOX-nano-prune*1	glass bottle	98.63	98.63	99.98	97.94	95.87
	bottle	97.53	98.57	98.56		
	can	95.43	94.35	94.35		
YOLOX-nano-prune*2	glass bottle	98.63	98.63	99.98	97.94	95.98
	bottle	97.53	98.57	98.56		
	can	95.43	94.35	95.28		
YOLOX-nano-prune*3	glass bottle	98.63	98.63	99.98	97.85	96.09
	bottle	97.18	98.57	98.89		
	can	95.95	93.79	94.67		
YOLOX-nano-prune*4	glass bottle	100.0	98.63	99.98	97.76	96.22
	bottle	96.83	98.21	98.54		
	can	96.53	94.35	94.75		
YOLOX-nano-prune*5	glass bottle	100.0	98.63	98.63	97.34	95.83
	bottle	96.14	97.86	98.16		
	can	95.98	94.35	95.24		
YOLOX-nano-prune*6	glass bottle	100.0	98.63	98.63	97.30	95.84
	bottle	96.83	98.21	98.53		
	can	96.53	94.35	94.74		
YOLOX-nano-prune*7	glass bottle	100.0	98.63	98.63	97.18	95.55
	bottle	96.48	97.86	98.16		
	can	95.98	94.35	94.75		

Table 2
Results of improved YOLOX proposed in this study on the Beverage Container Dataset

Model	Parameters(rate[%])	channels(Rate[%])	GFLOPs	FPS
Original YOLOX-nano	897,144(100.0)	7720(100.0)	1.28	237
YOLOX-nano-prune*7	507,938(56.62)	5845(75.71)	0.74	243

5. Conclusion

In this study, we create a unique dataset and propose a lightweight model with a reduced number of parameters to achieve a practical inference speed within limited computational resources. By performing iterative structural pruning on the backbone, we reduce computational load and improve inference speed while maintaining the backbone's feature extraction capabilities. The experiment demonstrates that our model, built through structural pruning of the YOLOX

Table 3
Results of Theoretical performance evaluation on edge devices

Model	GFLOPs	Estimated Inference Time	Estimated FPS
Original YOLOX-nano	1.28	0.71	1.4
YOLOX-nano-prune*7	0.74	0.41	2.4

backbone, significantly reduces computational complexity, proving its effectiveness. In essence, our improved network with iterative structural pruning could be a significant step towards achieving robust object detection performance on edge devices.

We acknowledge a limitation of our study is that the evaluation is conducted solely on a custom dataset. However, this dataset includes unique, real-world challenges, such as overlapping objects, extreme deformation, and image blur, that are often underrepresented in public datasets. More importantly, the iterative structural pruning approach we applied to YOLOX is a task-agnostic and universal strategy for model compression. Therefore, we believe the method is broadly applicable for other object detection tasks requiring deployment on edge devices.

Looking ahead, our future work proceeds in two directions. First, we plan to validate the generality of our approach on public benchmark datasets. Second, we aim to apply this lightweight model to even more resource-constrained hardware, such as CPU-only devices, to realize our ultimate goal of developing a system that addresses environmental issues caused by beverage container waste.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] Y. Ge, Z. Li, X. Yue, H. Li, L. Meng, Dataset purification-driven lightweight deep learning model construction for empty-dish recycling robot, IEEE Transactions on Emerging Topics in Computational Intelligence (2025) 1–16.

[2] X. Yue, L. Meng, Yolo-sm: A lightweight single-class multi-deformation object detection network, IEEE Transactions on Emerging Topics in Computational Intelligence 8 (2024) 2467–2480.

[3] X. Yue, L. Meng, Yolo-msa: A multiscale stereoscopic attention network for empty-dish recycling robots, IEEE Transactions on Instrumentation and Measurement 72 (2023) 1–14.

[4] Z. Li, Y. Yan, X. Wang, Y. Ge, L. Meng, A survey of deep learning for industrial visual anomaly detection, Artificial Intelligence Review 58 (2025) 279.

[5] Z. Li, Y. Ge, L. Meng, A multi-scale information fusion framework with interaction-aware global attention for industrial vision anomaly detection and localization, Information Fusion 124 (2025) 103356.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto,

H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).
[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: European conference on computer vision, 2016, pp. 21–37.
[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.
[9] X. Zhang, X. Zhou, M.-H. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.
[10] M. Tan, R. Pang, Q. V. Le, Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2020, pp. 10781–10790.
[11] R. Lyu, NanoDet: Super-fast and light-weight anchor-free object detection model, https://github.com/RangiLyu/nanodet, 2021.
[12] 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.
[13] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 7263–7271.
[14] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).
[15] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, arXiv preprint arXiv:2004.10934 (2020).
[16] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, et al., Yolov5 by ultralytics, https://github.com/ultralytics/yolov5, 2020.
[17] C. Li, L. Li, H. Geng, B. Jiang, X. Zhang, Z. Wu, Q. Dang, C. Chen, X. Wei, Y. Han, et al., Yolov6: A single-stage object detection framework for industrial applications, arXiv preprint arXiv:2209.02976 (2022).
[18] 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.
[19] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, Yolox: Exceeding yolo series in 2021, in: Advances in Neural Information Processing Systems, volume 34, 2021, pp. 26024–26035.
[20] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Chang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE international conference on computer vision, 2017.
[21] S. Han, J. Pool, J. Tran, W. J. Dally, Learning both weights and connections for efficient neural networks, in: Advances in neural information processing systems, 2015.