Iraqi Journal of Information and Communications Technology(sss)
Vol. 8, Issue 1, April 2025
www.ijict.edu.iq
ISSN: 2222-758X
e-ISSN: 2789-7362

ENHANCED YOLOV8 FOR EDGE COMPUTING: A DEEP LEARNING APPROACH FOR MEMORY OPTIMIZATION

Athraa Kh. Al-Zihairy [1], **A. E. Abdelkareem** [2]
[1] College of Language, Baghdad University, Jadriya, Baghdad, Iraq
[2] Department of Computer Networks Engineering, College of Information Engineering, Al-Nahrain University, Baghdad, Iraq
athraa.khalid@coie-nahrain.edu.iq[1], ammar.algassab@nahrainuniv.edu.iq[2]
Corresponding Author: **A. E. Abdelkareem**
Received:21/05/2024; Revised:02/08/2024; Accepted:29/10/2024
DOI:10.31987/ijict.8.1.283

***Abstract*- Enhanced YOLOv8 for Edge computing with a deep learning approach for memory optimization is proposed as an optimized Artificial Intelligence (AI) algorithm specifically designed for edge learning, which can efficiently enable Convolutional Neural Networks (CNN) deployment on Internet of Things (IoT) edge devices with less memory footprint and more rapid inference. This paper optimizes the YOLOv8 model architecture using the structured pruning method (Layer Pruning) to minimize non-contributing parameters. The proposed pruned model with loaded pre-trained model weights on the COCO dataset and after fine-tuning on the self-driving car dataset maintains high accuracy, with a negligible 0.011 decrease compared to the baseline, while reducing the model size by 33% and increasing detection speed to 196 Frame Per Second (FPS). These results are significant in making the proposed model well-suited for edge environments with limited memory capacity, especially for the dataset used allowing the vehicle to react rapidly to environmental changes for practical applications.**

keywords: **Edge intelligence; Convolutional Neural Networks (CNNs), YOLOv8, Self-driving, Deep learning, Optimized AI algorithm.**

I. INTRODUCTION

Intelligence on the edge [1], quickly predicts outcomes for several instantaneous applications, like virtual reality, sophisticated monitoring programs, and autonomous vehicles, by transferring intelligence from the cloud to peripheral devices. Edge [2] devices replace AI-based cloud computing methods because real-time applications require the lowest latency. Consequently, the implementation of edge intelligence installation [3] decreases time to response and network capacity requirements for moving data from a device to a cloud network enabling analysis and outcomes retrieval. Smarter device-based applications for home automation, wearable healthcare, smart parking, intruder detection with smart cameras and smartphones, and smart manufacturing applications have enhanced people's daily lives [4], [5]. Even while intelligent Internet of Things (IoT) edge computing has benefited from several applications, there are significant obstacles in integrating intelligence through deep learning models into IoT devices due to resource constraints related to memory, processing power, and energy efficiency.

The size of deep learning models keeps growing as the number of nodes and layers rises, which makes it challenging to implement on portable devices with limited resources (memory, CPU, energy, and internet). Deep Learning (DL) algorithms are inserted into edge devices to give them intelligence for decision-making. A popular framework, Convolutional Neural Networks (CNNs) have shown impressive results in a variety of applications, including virtual reality, speech recognition, augmented reality, computer vision, robotic vision, and autonomous driving. With the help of edge intelligence [6-8]. CNN

ISSN: 2222-758X
e-ISSN: 2789-7362

deployment on IoT devices is difficult, nevertheless, because DL models' heavy-weight architecture makes them resource-intensive. Furthermore, it affects several performance indicators, including energy consumption, memory footprint, accuracy, throughput, end-to-end latency, and communication size. Existing research focuses on optimizing AI algorithms specifically designed for edge learning, which can leverage the available resources efficiently while maintaining high accuracy and low latency. A framework was suggested to minimize the CNN model footprint being included in devices of IoT by systematically optimizing YOLOv8 model. The driverless vehicle uses case serves as an example of IoT edge computing through the crucial application of computer vision. The autonomous vehicle uses its front and rear cameras to identify objects and people. The cameras record continually, and if an object is detected while the car is driving, it has to be halted or slowed down right away. In this case, processing the vehicle's collected photographs on the cloud for inference takes time and can result in accidents. On the other hand, results can be produced more rapidly and reliably when processing is done directly on the edge device. Because local data processing handles data at the source edge devices, it reduces transmission time, which is advantageous for instantaneous applications.

There are two types of DL-based detection techniques: single-stage algorithms and double-stage algorithms. Single-stage algorithms, such as YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, SSD, solve the object detection problem as a regression problem. For object detection, the double-stage techniques use region suggestion networks or selective search algorithms, like faster R-CNN, R-CNN, R-FCN, and fast R-CNN. They are slow, which is a drawback, but they also have great precision. Single-stage algorithms, on the other hand, have the benefit of reaching an acceptable balance between speed and accuracy. Deploying them on embedded devices is simpler.

All studies as mentioned above have utilized different models with different pruning methods. In this paper, the layer pruning method is used with the YOLOv8 model to reduce its layers for optimization purposes to get as reduced model size as possible and faster inference time to fit the model into memory-constraints edge devices for different industrial applications. the contributions of this paper are summarized as follows:

- Layer pruning approach is presented to prune the layers characterized for detection of small and medium objects, to effectively compress neural network models meeting compression requirements on resource-limited edge devices.
- The pruning method is applied to the state-of-the-art YOLOv8 model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility.
- Evaluate the performance of state-of-the-art deep neural networks on the Udacity Self-Driving Cars dataset to demonstrate the effectiveness of our approach. Layer pruning successfully achieved significant enhancement for the DL model in terms of reducing model size and inference time.

The rest of this paper is organized as follows. In Section II, the literature survey of the project is introduced. The proposed methodology is in Section III. In Section IV, the proposed model implementation is presented. In Section V, experimental results are presented. Section VI discusses the results. Finally, Section VII draws the conclusions.

ISSN: 2222-758X
e-ISSN: 2789-7362

II. LITERATURE SURVEY

Edge intelligence in the IoT has emerged through optimizingâdevices with limited resources with deep learning models installed to facilitate quick decision-making. In addition, by bringing advanced analytics nearer to its origins, edge intelligence lowers latency and network overload [9]. However, DL models require a large amount of computing resources. When it comes to large-scale applications, Xuehui Shen et al [10], proposed YOLOv5 to detect the wearing condition of safety helmets and anti-slip shoes. The channel pruning strategy was used to lower both costs and hardware required for the model to operate. The weight of the connection points in the neural network's network model is determined by the weight coefficient, which is the basis for the channel pruning technique.

In [9], the authors, Soumyalatha Naveen and ManjunathR.Kounte, proposed a framework to introduce a complete solution that makes it possible to install CNNs on dispersed Internet of Things devices for quicker inference and smaller memory footprints. This framework takes into account a pretrained YOLOv2 model, to which a weight pruning technique is implemented in order to minimize the number of non-contributing parameters. To divide the CNN's fused layers vertically and distribute the division among the edge devices for input processing, they employ the fused layer partitioning technique. proposed model achieved inference latency of 5 to 7 seconds for 3×3 to 5×5 fused layer partitioning for five devices with a 9% improvement in memory footprint.

In [11], Yifei Zhang et al, proposed a network pruning method inspired by the PSO algorithm, named SPSO-Pruner to improve the detection accuracy and speed, while reducing the model parameters. YOLOv5 is used as the up-to-date one-stage detector to adapt to real-time requirements. The proposed optimization method can reach better accuracy with less than 50% parameters.

In [12], Gamanayake C et al., proposed a novel greedy approach called cluster pruning, which provides a structured way of removing filters in a CNN by considering the importance of filters and the underlying hardware architecture. Cluster pruning outperformed conventional filter pruning on multiple hardware architectures.

In [13], Guangli Li et al., introduced FlexPruner, an innovative approach for compressing and accelerating neural network models via flexible-rate filter pruning, which can automatically select the number of filters to be pruned to achieve more flexible rate settings, thereby effectively compressing and accelerating deep learning models on intelligent edge accelerators. This method reduces model size with minimal accuracy loss.

However, All of these studies used different weight and filter pruning methods, which have limitations such as the iterative process to keep a minimum number of filters per layer during optimization to allow for data passing. In addition, filter pruning is constrained by the depth of the model and the connection dependency in the architecture.

The layer pruning method over filter pruning has advantages in terms of latency reduction in that the former is not constrained by the original model's depth and thus allows for a larger range of latency reduction. This paper proposed YOLOv8 model as a state-of-the-art model optimized using the layer pruning method to reduce its size and inference time to be more suitable for memory-constrained edge devices.

ISSN: 2222-758X
e-ISSN: 2789-7362

III. PROPOSED METHODOLOGY

A CNN-based optimized model is proposed for edge devices with the structure pruning method. The suggested framework's implementation notices the limited resources of IoT edge devices. The essential elements that form the basis of the research shall be thoroughly outlined in the methodology section. This comprises a complete explanation of the experimental setup, an in-depth investigation of the dataset, a detailed discussion of the YOLOv8 model architecture, and insights into the model modification method. The process illustrated in Fig. 1, offers a workflow that helps to navigate these important research components.

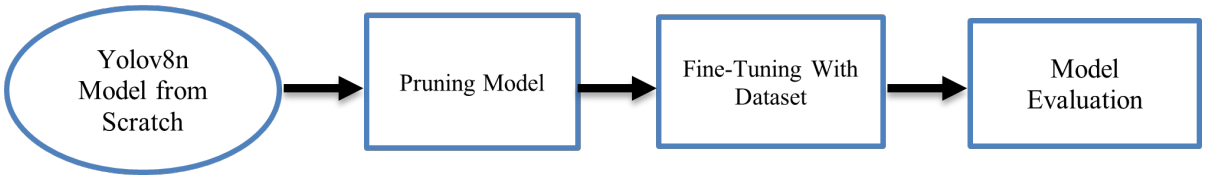


Figure 1: The proposed methodology workflow.

A. *YOLOv8 Model Architecture*

As real-time object detection systems progress, the YOLOv8 (You Only Look Once version 8) [14] model architecture stands as the best example. Starting with a strong backbone network (CSPDarknet53), convolutional layers extract feature maps from input pictures. The Cross-Stage Partial (CSP) connection, which YOLOv8 offers, makes it easier to integrate features from various network stages effectively. The YOLO head is made up of several detection layers, each of which is in charge of making different-scale object predictions. For class probabilities, objectness scores, and bounding box predictions, anchor boxes are used. YOLOv8 frequently uses Feature Pyramid Network (FPN) concepts to address objects of various sizes.

With the goal of multi-object detection, it generates predictions in a format that consists of bounding box coordinates, objectless scores, and class probabilities. all had the goal of detecting many objects. YOLOv8 uses a combination of loss functions, such as localization loss, confidence loss, and class loss, for model optimization during training. YOLOv8 is a reference-free object detection model that generates bounding boxes dynamically within the detection process, providing more versatility and precision in the detection outcomes, as opposed to relying on predefined reference boxes to detect objects. YOLOv8 is the best option for real-time and near-real-time object identification jobs because it strikes a compromise between high accuracy and processing economy.

It has multiple versions, including YOLOv8-tiny and YOLOv8-s, each designed to meet certain speed and accuracy needs [14]. This guarantees its adaptability to a wide range of computer vision applications, such as surveillance and autonomous driving.

ISSN: 2222-758X
e-ISSN: 2789-7362

B. *Pruning Model*

In this research, a novel method of layer pruning-based model optimization is presented, specifically designed for effective object detection at different sizes. To be more precise, the model's architecture is subjected to layer pruning to identify and eliminate elements that predominantly contribute to large object detection while keeping layers that are essential for small-size and medium-sized object recognition tasks. This methodical pruning technique is intended to drastically shrink the model's overall size while maintaining its capacity to detect objects with high accuracy at various scales. Through the identification and removal of superfluous or irrelevant layers associated with large object attributes, like comprehensive contextual data or high-resolution representations, the pruned model attains a more efficient structure that is optimized for resource utilization without sacrificing performance. This method is well-suited for practical deployment in scenarios demanding strong capabilities because it not only reduces computing complexity and memory requirements but also guarantees that the model maintains its ability to discriminate and localize objects of different sizes. Structured pruning can reconstruct a limited model with a regular structure by eliminating entire channels, filters, neurons, or even layers [15], as illustrated in Fig. 2. It can immediately speed up networks and shrink the size of neural networks without the assistance of specialized hardware or software (like sparse convolution libraries). Weights can be removed anywhere using unstructured pruning techniques, which can also achieve large prune ratios with minimal effect on accuracy. On the other hand, structured pruning involves removing parts of complete filters (or neurons, blocks, layers, channels, etc.), leading to a much smaller network and faster inference [16].

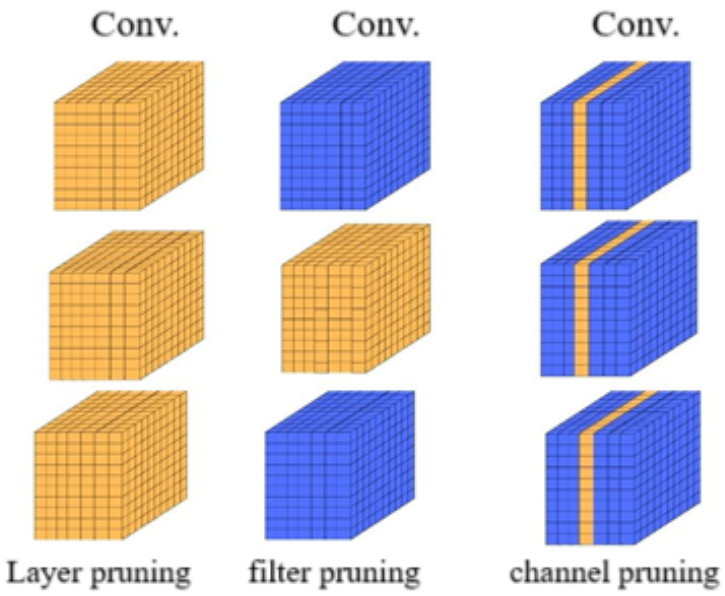


Figure 2: Three types of structured pruning.

ISSN: 2222-758X
e-ISSN: 2789-7362

C. *Fine-tuning with Dataset*

This research focuses on an enhanced version of the Udacity Self-Driving Car Dataset [17]. The dataset addresses critical shortcomings in object annotations, featuring:

1) Dataset Name: Self-Driving Car Dataset.
2) Total Classes: 11 objects relevant to self-driving cars.

Annotations are provided in CSV format for easy integration. The dataset contains 97,942 labels across 11 classes and 15,000 images, ensuring accuracy and reliability. The 11 classes include car, pedestrian, biker, truck, and 7 different types of traffic lights. The dataset aligns with the MIT License, facilitating research and development in self-driving cars. Its high-quality annotations, user-friendly 512×512 image resolution, and compatibility with diverse ML models make it invaluable.

Data division is a technique that creates a directory structure with the usual "train," "test," and "valid" folders to aid with dataset splitting. The appropriate ratios (e.g., 70% for training, 15% for testing, and 15% for validation) can be specified by users for each set. Fig. 3 shows some samples of Udacity Self-Driving Car Dataset.

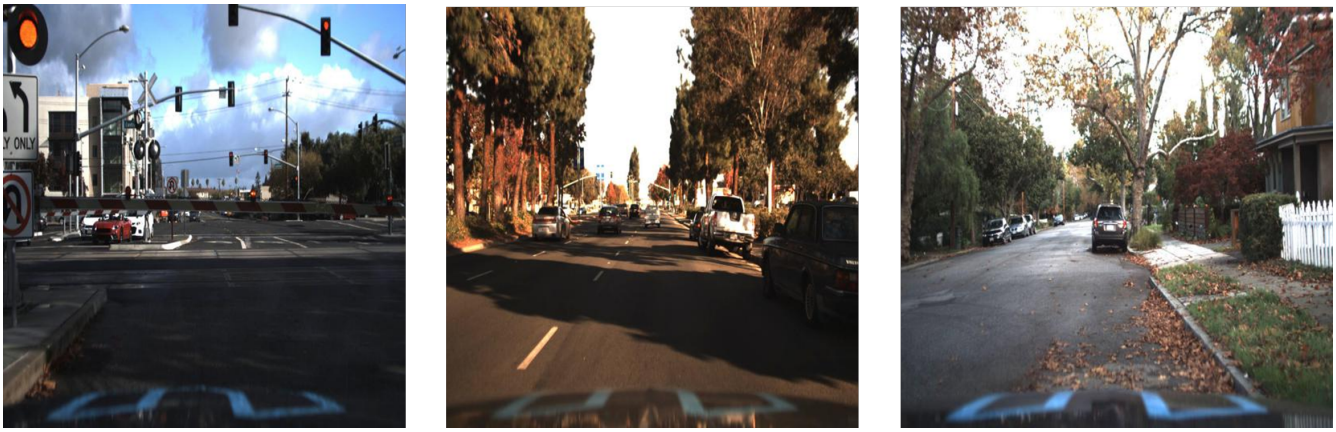


Figure 3: Samples of Udacity Self-Driving Car Dataset.

D. *Model Evaluation*

The metrics have been computed to assess the proposed model's effectiveness and compare it with the original model. Object detection models need mAP and inference time metrics to evaluate their performance. The calculation of mAP demand Average Precision (AP) and number of detection categories (c), for the dataset used c=11. The formula for mAP is as follows:

$$\text{mAP} = \frac{\sum_{c=1}^{C} \text{AP}(c)}{C} \tag{1}$$

ISSN: 2222-758X
e-ISSN: 2789-7362

IV. THE PROPOSED MODEL IMPLEMENTATION

The goal of the YOLOv8n model's pruning implementation is to produce a lightweight variant that is edge device optimized. The model is first organized as Fig. 4 of the YOLOv8n.yaml file shows, but it is carefully pruned to become smaller and use less processing power. Algorithm 1 explains the steps of Pruning YOLOv8n.

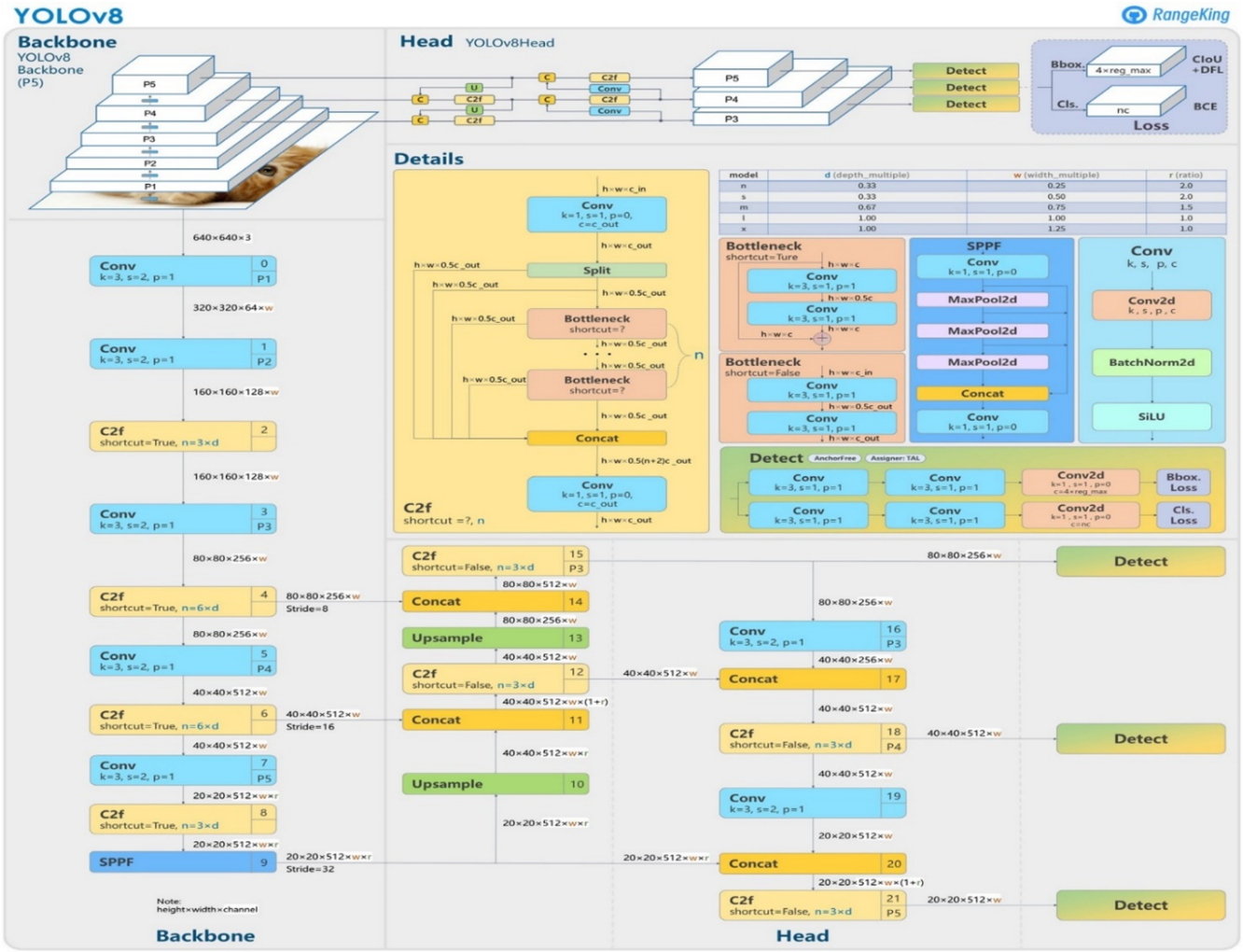


Figure 4: YOLOv8 architecture [18].

As shown in Fig. 5 and Fig. 6, this pruning technique entailed removing layers made especially for identifying large objects in images while keeping layers essential for identifying small and medium-sized objects. This can be done using the YAML file of YOLOv8n where can add and remove layers, or change the configuration of the network. After the pruning stage, a dataset is used to fine-tune the pruned model to restore any accuracy that has been lost. The pruned model is well-suited for deployment because this fine-tuning stage is essential to ensuring that it maintains its performance standards, notably in properly detecting objects of varied sizes.

ISSN: 2222-758X
e-ISSN: 2789-7362

Algorithm 1: Pruned YOLOv8n

Input: M (Modified YAML file of YOLOv8n model), D (training dataset)
Output: Optimized YOLOv8n model trained on the dataset

Begin
Sample the dataset D' from D
Split D' into train_ratio = 0.7, valid_ratio = 0.15, test_ratio = 0.15
Initialize the pruned model M
Load weights of pre-trained YOLOv8n model file into pruned model M
Fine-tune the weights of M using D'
Obtain the final compressed model M
End

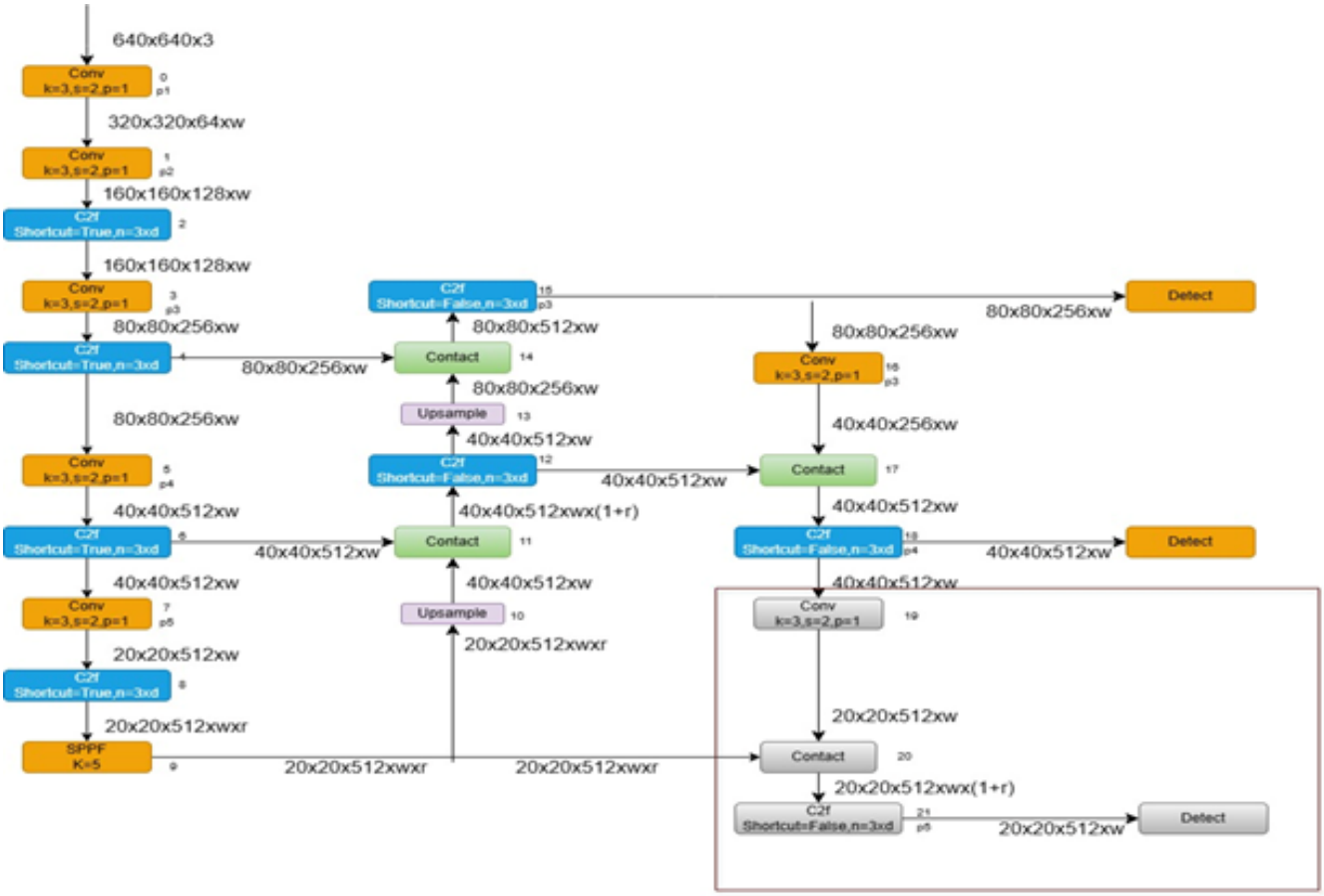


Figure 5: YOLOv8 architecture with shaded blocks for pruning.

ISSN: 2222-758X
e-ISSN: 2789-7362

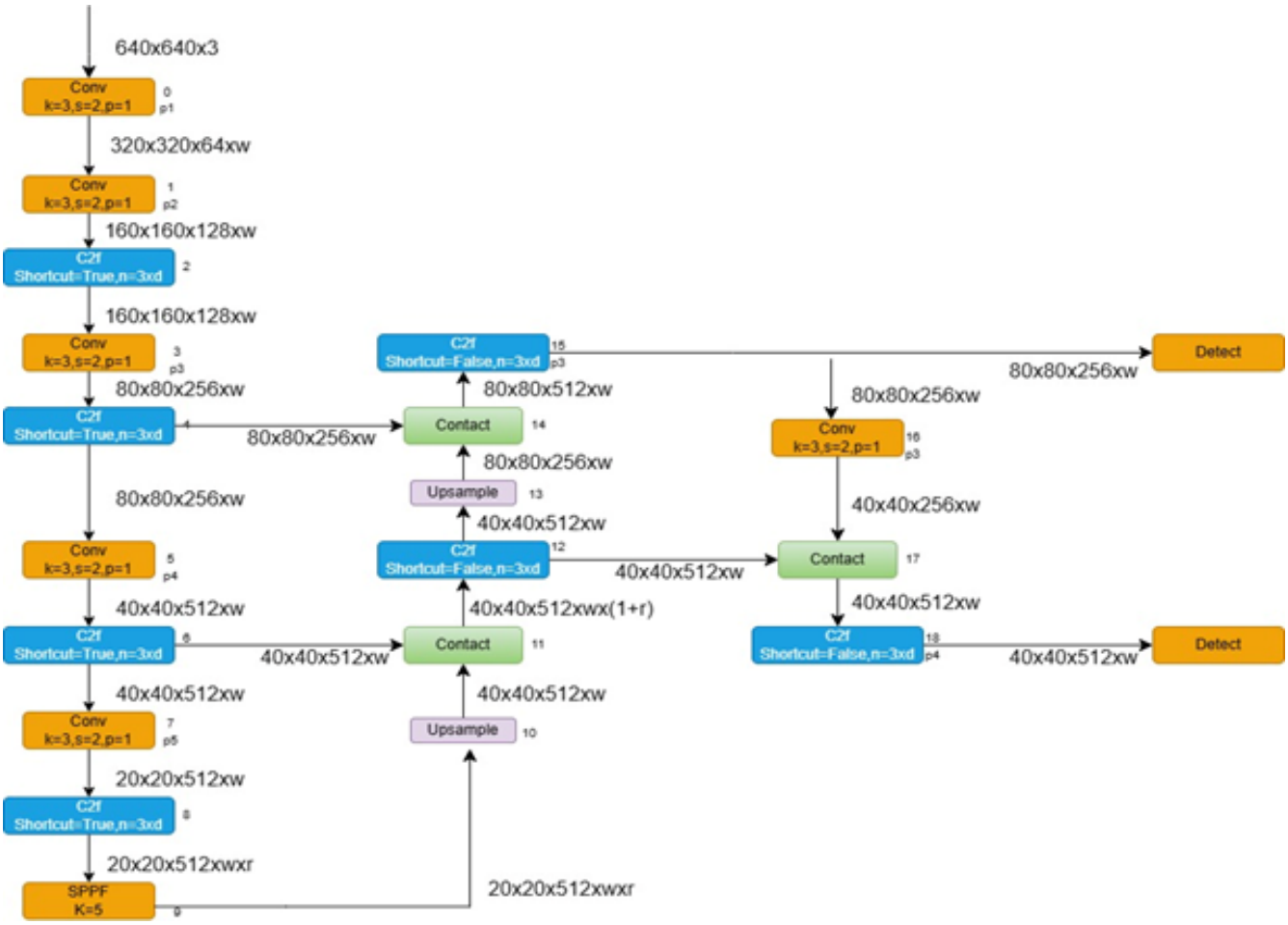


Figure 6: YOLOv8 architecture after pruning.

V. RESULT

There are 15,000 photos in the Udacity Self-Driving Car Dataset. However, because of the processing power constraint, a subset of 4000 images is used, split into 2,800 for training, 600 for validation, and 600 more for testing. For 100 epochs, the baseline model is trained from scratch. For every run, the default Adam W optimizer (learning rate = 0.001, weight decay = 0.0005, momentum = 0.9) was employed.

A. *Experiment (A)*

Part of the implementation is pruning the YOLOv8n.yaml model, which has a lot of layers and parameters. It is challenging to maintain the exact precision of the original model following size reduction. Fig. 7, presents the results of the Pruned YOLOv8n, which shows the successful training process with all performance metrics increasing and all losses decreasing. Table I shows only little accuracy is lost (1.6%) when the original model is pruned. Following pruning,

ISSN: 2222-758X
e-ISSN: 2789-7362

the model's dimensions, number of layers, and number of parameters all dropped by 33%, 33%, and 25%, in that order. Regaining the lost precision is the aim of fine-tuning, which reduces the model's size to 7.75 MB and maintains its computational efficiency for edge devices.

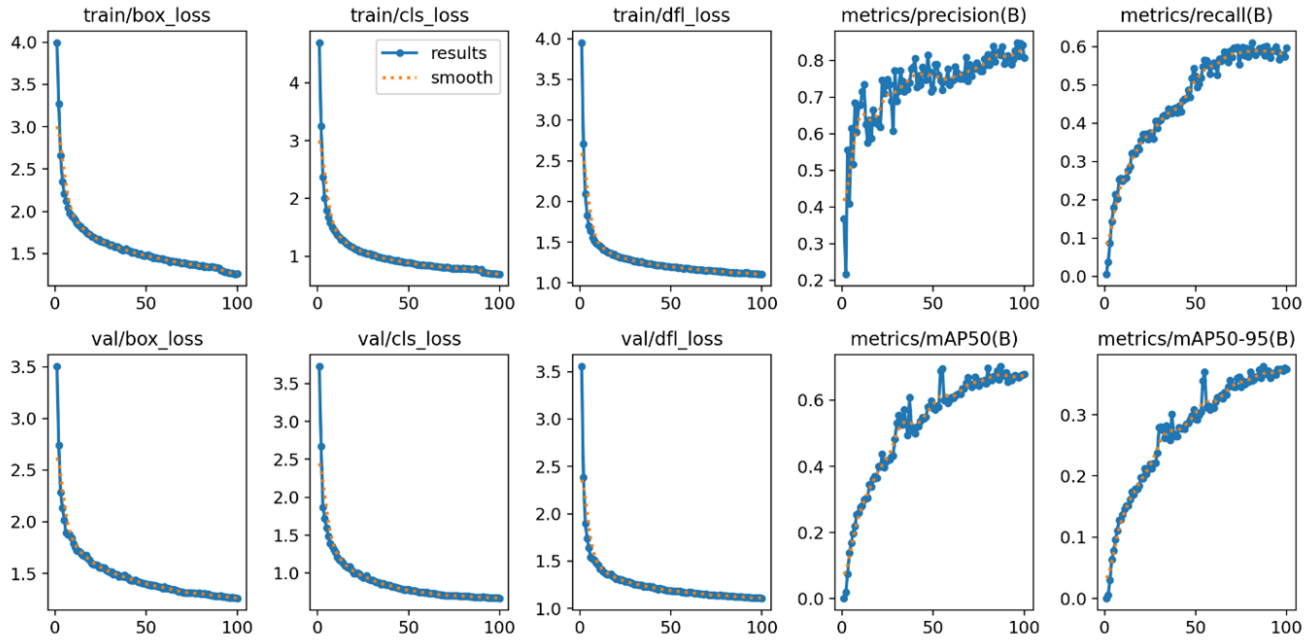


Figure 7: Results of fine-tuning Pruned YOLOv8n model.

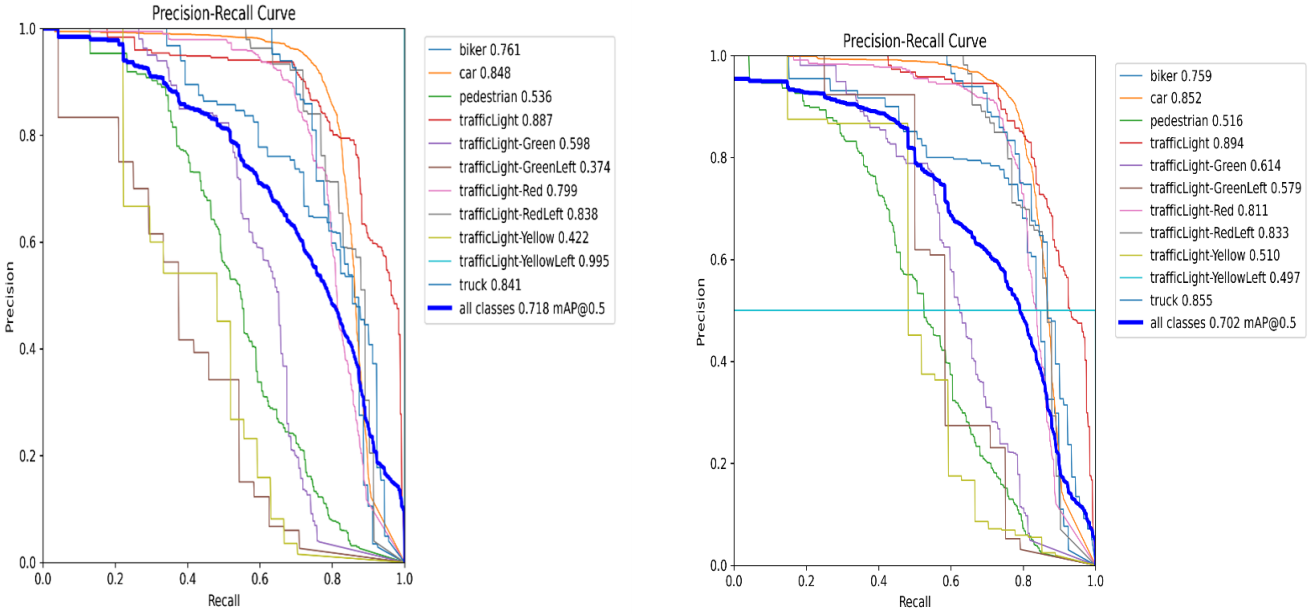
TABLE I
The Confusion Matrix

Model	No. of Parameters	No. of Layers	mAP@50 (%)	Model Size (MB)	Inference Time (ms)
Original YOLOv8n	3,012,993	255	71.8	11.6	2.0
Pruned YOLOv8n	1,998,198	190	70.2	7.75	2.0

Fig. 8 presents the Precision-Recall (PR) Curve for the model before and after pruning. Precision measures positive prediction accuracy, while recall assesses a modelâs ability to identify all positive cases. The curve showcases precision values across various classes, revealing individual class performance.

For instance, the 'Car' class for the pruned model boasts an impressive precision score of 0.871, signifying 87.1% accuracy in predicting cars. mAP, a comprehensive performance metric attains a score of 0.852, representing a 85.2% AP at a 0.5 confidence threshold, which is increased by 0.004 after pruning.

ISSN: 2222-758X
e-ISSN: 2789-7362



((a)) YOLOv8n model before pruning

((b)) YOLOv8n model after pruning

Figure 8: Precision-Recall Curve for YOLOv8n: (a) before and (b) after pruning without pre-trained.

B. Experiment (B)

The pre-trained original model on the COCO dataset (YOLOv8n.pt) has been loaded on the scratch file (YOLOv8n.yaml), which is loaded on the pruned model, in order to improve the model's accuracy. Fig. 9 shows the performance metrics of the Pruned YOLOv8n loaded with weights of the pre-trained model across 100 training epochs.

Table II presents metrics to evaluate the model and comparison between the model before and after pruning. The suggested model is 7.75 MB in size and has an accuracy of 83%, while the actual model is 12.1 MB in size and has an accuracy of 83.8%. The model's size significantly decreases as a result of the pruning. After pruning the YOLOv8n architecture, there is only a 0.8% loss in accuracy. Additionally, the inference time decreases from 2.3 ms to 1.8 ms, increasing the detection speed to 196 FPS.

TABLE II
Results of YOLOv8n model before and after pruning with pre-trained YOLOv8n

Model	No. of Parameters	No. of Layers	mAP@50 (%)	Model Size (MB)	Inference Time (ms)
Original YOLOv8n with pre-trained YOLOv8n	3,012,993	255	83.8	12.1	2.3
Pruned YOLOv8n with pre-trained YOLOv8n	1,998,198	190	83.0	7.75	1.8

For this experiment the Precision-Recall (PR) Curve in Fig. 10, for the pruned model for instance, the 'Car' class for the pruned model boasts an impressive precision score of 0.925, signifying 92.5% accuracy in predicting cars. mAP, a comprehensive performance metric attains a score of 0.868 representing an 86.8% AP at a 0.5 confidence threshold, which is decreased by only 0.011 after pruning.

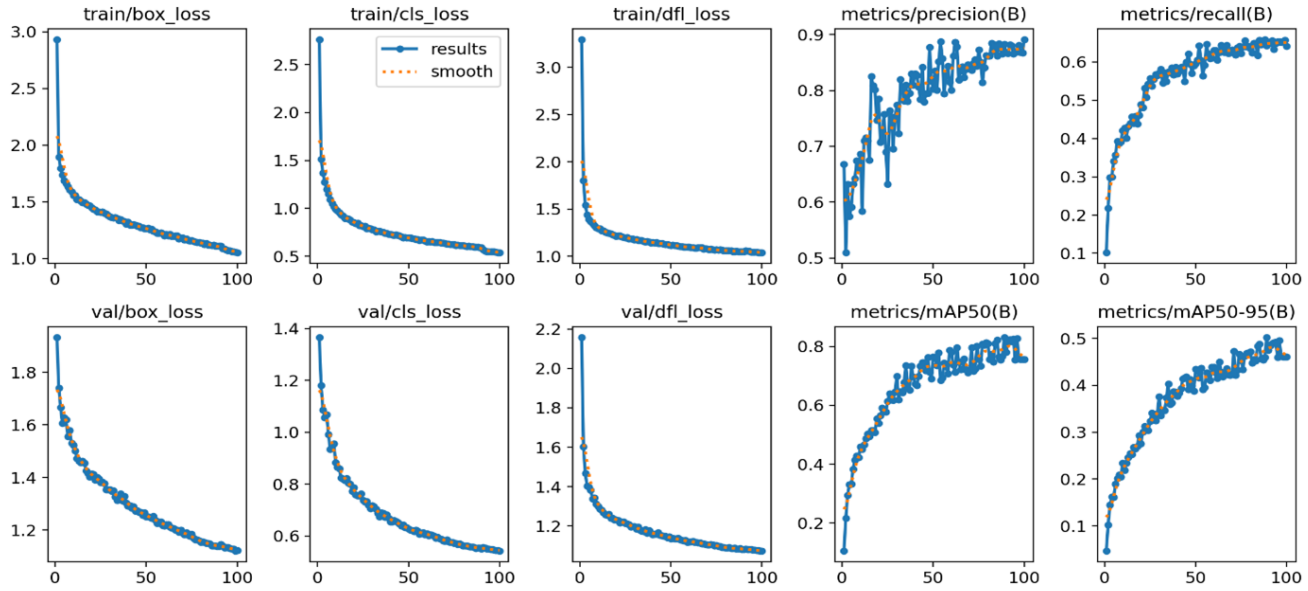
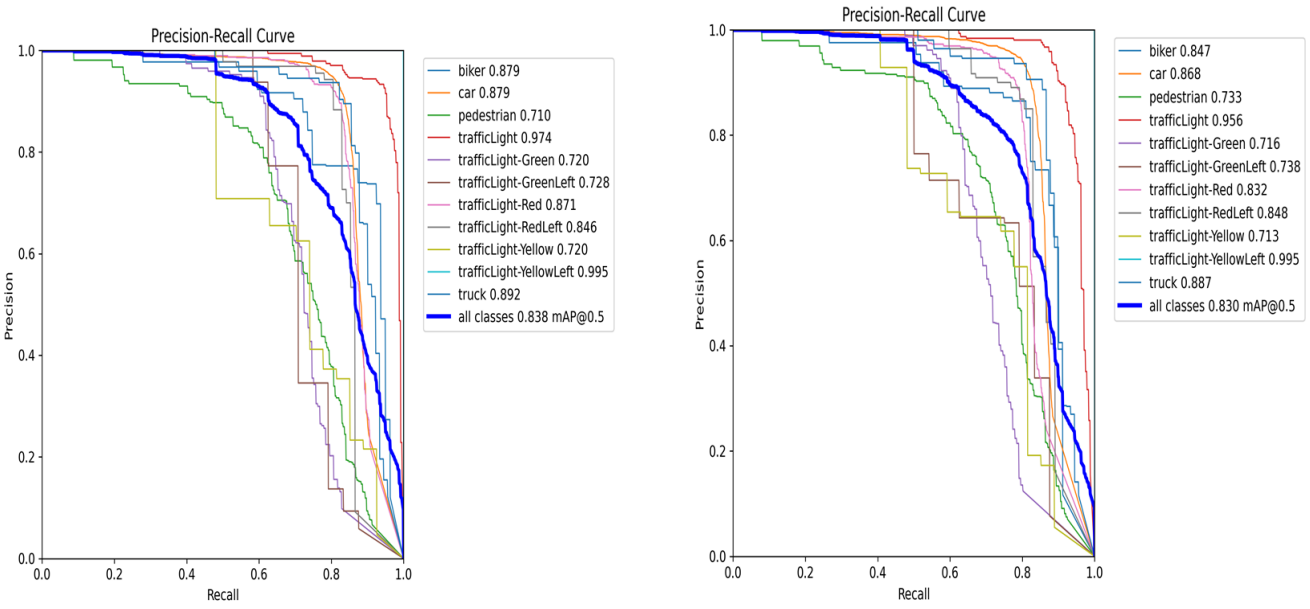


Figure 9: Results of fine-tuning pruned YOLOv8n model with pre-trained YOLOv8n.



((a)) YOLOv8n model before pruning

((b)) YOLOv8n model after pruning

Figure 10: Precision-Recall Curve for YOLOv8n: (a) before and (b) after pruning with pre-trained model.

Fig. 11 presents the visualization of detection results realized by the Pruned YOLO8n model trained on the Udacity Self-Driving Car Dataset, which proves that the proposed model performs well in detecting medium and small objects,

ISSN: 2222-758X
e-ISSN: 2789-7362

e.g., cars, traffic light-red, and pedestrians. In addition, the proposed model can detect individual objects very well. When there are multiple dense targets on an image, e.g., a group of cars near to each other, the proposed model can as well as characterize each object well. The qualitative detection results prove the effectiveness of the proposed model. The proposed model maintains good detection accuracy while achieving a considerable decrease in computing demands and storage space preservation making it more suitable for edge computing environments.

Figure 11: Visualization of detection results realized by Pruned YOLO8n model trained on the Udacity Self-Driving Car Dataset.

VI. DISCUSSION

A comprehensive comparison of the output of various YOLOv8n model configurations in Table I and Table II including modified models before and after pruning.

A. Comparing Pruned and Original YOLOv8n

The original model is 11.6 MB in size, with 255 layers, 3012993 parameters, and a mean Average Precision at 50% IoU (mAP50) of 71.8%. The inference time is 2.0 ms. With considerably fewer parameters (1998198) and layers (190), the trimmed model has a somewhat lower mean approximate posterior probability (mAP50) of 70.2%, a smaller model size of 7.75 MB, and the same inference time of 2.0 ms.

B. Comparing the Original YOLOv8n and the pre-trained YOLOv8n with the Pruned YOLOv8n in terms of training

The original version exhibits a slightly higher model size of 12.1 MB, a longer inference time of 2.3 ms, and an increased mAP50 of 83.8% (compared to 71.8% without pre-training) when utilizing a pre-trained YOLOv8nmodel. Similar to the pruned model without pre-training, the pruned version with pre-training likewise has fewer layers and parameters. Its mAP50 is a little lower at 83% (as opposed to 83.8% with the pre-trained original model), but it has a faster inference time of 1.8 ms and a smaller model size of 7.75 MB.

Pruning makes the model smaller with a slight decrease in accuracy by reducing the amount of features in the model. It is mentioned, however, that following pruning, fine-tuning is essential to preserving as much of the accuracy of the original model as feasible. By fine-tuning, the pruned model maintains its high accuracy by relearning some of the lost characteristics. With the benefit of smaller models and, in the case of the pruned model with pre-training, faster inference times. The pruned models have maintained high mAP50 scores (70.2% and 83%, respectively). This implies that trimming

ISSN: 2222-758X
e-ISSN: 2789-7362

and fine-tuning together can be a successful approach for optimizing model performance, especially when trying to deploy models on devices where resources such as memory and compute power are limited.

The potential limitation of the proposed method is the inability of the proposed model to detect big objects because of the pruning of the layers of model architecture related to big object detections. The dataset used contains only small and medium objects according to the classes of the dataset and the pruned model achieved high accuracy detection according to the results. However, it is not appropriate for applications that need datasets containing big objects.

The trade-offs between model size reduction and accuracy loss can be solved by reducing the model as much as possible while accuracy loss is minimized by pruning the layers that contribute less to the final results to maintain model performance. Balancing the size reduction and accuracy loss needs to consider some aspects like the available resources, the acceptable degree of accuracy loss, and the potential safety issues for using a less accurate model.

VII. CONCLUSIONS

The layer pruning method is applied in this work to lower the cost for large-scale applications and the hardware requirements for model operation. The YOLOv8 model's layers that are specialized to large object identification are pruned using the layer pruning concept, but the other layers, which are required for small and medium objects in real-time self-driving car applications utilizing the Udacity Self Driving Cars dataset, are maintained. These layers will not impact the final detection result and have minimal bearing on the model's accuracy in the end. Following layer pruning, 65 layers are eliminated in total, resulting in a 25.4% decrease in YOLOv8n's layer density. The mAP reaches 83.2% after consistent parameter fine-tuning, demonstrating YOLOv8n's layer pruning's ability to successfully recognize all Udacity Self-Driving Car classes. With a detection speed of 196 FPS, the model size was lowered by 33% and the mAP by only 0.8%, meeting the requirements for accuracy and real-time performance. This makes it possible to implement large DL models on IoT edge devices with limited resources. Further studies aim to explore a quantization technique in greater detail in order to further minimize the model size.

FUNDING

None

ACKNOWLEDGEMENT

The author would like to thank the reviewers for their valuable contribution in the publication of this paper.

CONFLICTS OF INTEREST

The author declares no conflict of interest

REFERENCES

[1] D. Xu et al., "Edge Intelligence: architectures, challenges, and applications," arXiv.org, Mar. 26, 2020. https://arxiv.org/abs/2003.12172
[2] S. Naveen and M. R. Kounte, "Distributing the cloud into fog and edge: new weather in IoT based deep learning," in Proceedings of the 2nd International Conference on Recent Trends in Machine Learning, Smart Cities and Applications, Springer Nature Singapore, 2022, pp. 749â758.
[3] Y. Liu, M. Eng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: multiaccess edge computing for 5G and Internet of things," IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6722â6747, Aug. 2020, doi: 10.1109/JIOT.2020.3004500.
[4] Singh, S. C. Satapathy, Gutub, and Roy, "I-based mobile edge computing for IoT: applications, challenges, and future scope," Arabian Journal for Science and Engineering, vol. 47, no. 8, pp. 9801â9831, Aug. 2022, doi: 10.1007/s13369-021-06348-2.

[5] H. Sodhro, S. Irbhulal, and V. H. C. de Albuquerque, "Artificial intelligence-driven mechanism for edge computing-based industrial applications," IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 4235â4243, Jul. 2019, doi: 10.1109/TII.2019.2902878.
[6] ZhouZ, ChenX, LiE, Zeng L, Luo K, Zhang J. "Edge intelligence: paving the last mile of artificial intelligence with edge computing," Proc IEEE. 2019;107(8):1738-1762.
[7] Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY. "Edge intelligence: the confluence of edge computing and artificial intelligence," IEEE Internet Things J. 2020;7(8):7457-7469.
[8] LiE, ZhouZ, ChenX. "Edge intelligence: on-demand deep learning model co-inference with device-edge synergy," Proceedings of the 2018 Workshop on Mobile Edge Communications; August 7, 2018:31-36; Budapest, Hungary.
[9] Naveen, Soumyalatha, and Manjunath R. Kounte. "Memory optimization at edge for distributed convolution neural network." Transactions on Emerging Telecommunications Technologies 33.12 (2022): e4648.
[10] Shen, Xuehui, et al. "Yolov5-based channel pruning is used for real-time detection of construction workersâ safety helmets and anti-slip shoes in informationalized construction sites." Journal of Physics: Conference Series. Vol. 2031. No. 1. IOP Publishing, 2021.
[11] Y. Zhang, X. Liu, Y. Chen, L. Xie, H. Wang, and O. Salem, "SPSO-Pruner: a network pruning method on YOLOv5 for fewer categories scenarios," Multimed Tools Appl, vol. 83, no. 4, pp. 11493â11506, 2024, doi: 10.1007/s11042-023-16038-w.
[12] C. Gamanayake, L. Jayasinghe, B. Ng, and C. Yuen, "Cluster Pruning: An Efficient Filter Pruning Method for Edge AI Vision Applications," Mar. 2020, doi: 10.1109/JSTSP.2020.2971418
[13] G. Li et al., "Optimizing deep neural networks on intelligent edge accelerators via flexible-rate filter pruning," Journal of Systems Architecture, vol. 124, p. 102431, 2022.
[14] Ultralytics (2023) YOLOv8 Docs. Retrieved from https://docs.ultralytics.com/. accessed April 27, 2024
[15] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in ECCV, 2018, pp. 304â320.
[16] Cheng, Hongrong, Miao Zhang, and Javen Qinfeng Shi. "A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations." arXiv preprint arXiv:2308.06767 (2023).
[17] Self-Driving Car Dataset (roboflow.com) accessed April 27, 2024
[18] Glenn-Jocher (2023) https://github.com/ultralytics/ultralytics/issues/189, Accessed May 10, 2024.

