Iraqi Journal of Information and Communications Technology(IJICT)
Vol. 8, Issue 2, August 2025
www.ijict.edu.iq
ISSN:2222-758X
e-ISSN:2789-7362

YOLO OPTIMIZATION FOR EDGE AI: A LIGHTWEIGHT APPROACH FOR DEEP SKY OBJECT DETECTION

Ula Kh. Altaie [1], A. E. Abdelkareem [2], Abdullah Alhasanat [3]
[1,2] Department of Computer Networks Engineering, College of Information Engineering, Al-Nahrain University, Jadriya, Baghdad, Iraq
[3] Alhussein Bin Talal University, Jordan
ula.k@nahrainuniv.edu.iq [1], ammar.algassab@nahrainuniv.edu.iq [2], abad@ahu.edu.jo [3]
Corresponding Author: **A. E. Abdelkareem**
Received:03/05/2025; Revised:15/06/2025; Accepted:06/08/2025
DOI:10.31987/ijict.8.2.331

***Abstract*- Accurate and efficient object detectors on resource-constrained edge devices are essential for real-time space applications such as satellite monitoring and deep sky exploration. However, Deep Learning (DL) models face difficulties when implemented in environments with limited memory and computational capabilities. To address this problem, this study proposes two lightweight architectures based on YOLOv10s and YOLOv10n that integrate ShuffleNetv2 and MobileNetv2. The Shuffle architecture improves efficiency by taking the advantages of the depthwise separable convolution and channel shuffling. The Mobile-Shuffle architecture combines Shuffle blocks with MobileNetv2-inspired blocks to enhance feature extraction and representation while maintaining low computational cost. This work evaluates both architectures on the augmented DeepSapceYOLODataset dataset, which contains a collection of annotated deep sky objects with different sizes and complexities. The evaluations demonstrate that the Shuffle architecture achieves the highest mAP50 on both sets, the fastest inference speed, and the smallest model size, especially in YOLOv10s architecture. While the Mobile-Shuffle architecture shows its strength in the YOLOv10n variant by improving the detection accuracy, it comes at the cost of the increased inference latency. These findings demonstrate that the proposed models offer a practical trade-off between accuracy, speed, and size, making them well-suited for edge deployment in space-related scenarios.**

keywords: **YOLOv10, Edge Computing, Structure Modification, ShuffleNetv2, MobileNetv2, Deep Sky Objects.**

I. INTRODUCTION

Object detection task is one of the most critical and fundamental subfields in computer vision domain, it is the foundation for a wide range of applications, from autonomous driving to satellite imagery analysis [1]. Edge computing has become essential for applications requiring real-time processing, low latency, and data privacy. However, deploying deep learning-based object detectors on edge devices remains challenging because of their limited memory and computational resources, and their low-power processing capabilities [2, 3].

Many versions of the popular object detectors, such as You Only Look Once (YOLO), which offer a high detection performance, are not suitable for efficient edge deployment due to their significant computational demands [4, 5]. To overcome these challenges, the researchers have focused on optimizing YOLO versions using different approaches to achieve a better balance between the performance efficiency and model complexity [6, 7].

In the context of space-related applications such as satellite and deep-sky exploration, the need for lightweight and accurate object detection models is a requirement. These applications involve processing high-resolution astronomical imagery in real time, using resource-limited devices within the satellites, space probes, drones, and deep-space telescopes. This paper addresses these challenges by optimizing YOLOv10s and YOLOv10n [8] for efficient edge deployment, specifically targeting the detection of deep-space objects.

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

This work proposes two lightweight architectures relying on the concepts of ShuffleNetv2 [9] and MobileNetv2 [10] lightweight modules and evaluate our approach on the augmented version of DeepSpaceYoloDataset [11],[12] which includes objects of different sizes and complexities. Also analyze the proposed optimized models: first with the entire dataset and then with a subset that containing only large objects to discuss the impact of object size on the detection accuracy and model efficiency. The main contributions of this paper are as follows:

- To reduce the models' complexity while maintaining detection accuracy, two lightweight YOLO architectures are proposed. The first architecture integrates ShuffleNetv2 inspired blocks into its backbone, while the second combines the previous Shuffle blocks with simplifies MobileNetv2-based blocks.
- To prove the optimized models' suitability for edge deployment, comprehensive performance analyses are performed on the entire augmented DeepSpaceYoloDataset, as well as on the large-object subset.
- Benchmarking results demonstrate the trade-offs between detection accuracy, model size, and inference speed, showing the practicability of our architectures for real-world edge applications.

The experimental results indicate that the proposed architectures of this work reduce the model size and computational cost compared with the default YOLOv10s and YOLOv10n architectures, without compromising the detection accuracy, which makes these models good options for real-time object detection edge applications, especially in space exploration and satellite imagery analysis.

The subsequent sections of this paper are organized as follows: Section II provides a survey of related works in the field. Section III introduces the default YOLOv10 structure and then describes the proposed architectures which using Shuffle and Mobile blocks. Section IV shows the experimental configuration details. Then Section V analyzes the obtained results. Section VI concludes the paper and shows the potential future works.

II. RELATED WORKS

Recent space-related applications including space explorations tools, satellite-based space surveillance systems, and astronomical object detectors have widely explored Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) technologies. Various studies have optimized and evaluated these technologies to enhance the identification and tracking of deep space objects, like galaxies and black holes, and other astronomical phenomena [13, 14, 15]. Many research efforts have focused on optimizing YOLO performance for space-related scenarios to achieve better detection accuracy and model adoptability for processing the complex space imagery. However, these improvements were designed based on high-performance systems, and not on resource-constrained edge devices [16, 17, 18].

Despite the growing importance of edge computing in the computer vision field, there are only few studies that focus on optimizing deep learning-based object detection models designed for space applications in resource limited environments. For example, in [19] proposed STar-DETR, a model that give the priority to the model's efficiency for space object detectors deployment on resource-constrained sensors. To achieve a good balance between model's complexity and detection accuracy, they used an improved MobileNetv4 module and the group shuffle convolution for model complexity and parameters reduction, and they incorporated the dynamic depthwise shuffle transformer feature fusion module and MPDSIoU

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

loss function for detection performance improvement. The methodology used to develop Star-DETR provided 64.8% parameters reduction, and 41.8% complexity reduction with 89.9% mAP0.5:0.95. This highlights a significant research gap in developing fast, lightweight, real-time, space-related detection models based on YOLO algorithm.

While YOLO optimization for lightweight space object detection remains limited, many studies have concentrated on modifying YOLO to be an appropriate model for autonomous driving applications, medical imagery tools, industrial and agricultural management systems, which are running on resource-limited hardware. Various techniques have been explored to design efficient object detectors such as model pruning [20],[21], quantization [22], or/and replacing the standard heavy convolutional blocks with more lightweight modules like ShuffleNet, MobileNet and GhostNet [23]. This paper review some researches that primarily integrating the ShuffleNetv2 and MobileNet lightweight modules in the YOLO structure, as they inspired our modified YOLO.

In [24] worked on their own augmented dataset of wheat grain, they integrated the ShuffleNetv2 lightweight module into YOLOv8n backbone and modified the model's neck with the large separable kernel attention module combined with depthwise separable convolutions. The proposed model, YOLO-SDL, is a fast lightweight detector with low complexity and 96.5% mAP50 and 85.9% mAP0.5:0.95, these results make it a good option as a wheat grain detector in resource-limited environment.

YOLOv5s-ShuffleNetV2-SE-EIOU is a lightweight model designed to avoid the issues facing drone-based vehicle detectors by Xu In [25]. To reduce model size and speed up the detection task, the researchers introduced the ShuffleNetv2 module into YOLOv5 backbone. To improve the detection accuracy especially in complex environment, they also applied SE attention module in the model's backbone. Finally, to enhance the regression accuracy and speed up the prediction boxes' convergence, they used EIOU function as loss function. This methodology is suitable for resource-constrained hardware, like drones, as it achieved 23.4% parameter reduction, 2.9% mAP50 and 3.8% mAP0.5:0.95 improvements.

In [26] modified the YOLOv8 backbone and neck in order to introduce a lightweight insulator small defect detector suitable for edge deployment. The researchers integrated the ShuffleNetv2 module into the model's backbone for parameters reduction and applied the cross-scale feature fusion module in the model's neck for small defect detection ability improvement. The results prove the adaptability of the proposed model to edge environments as it requires only 2.7million parameters and 8.2 GFLOPs with 99.1% mAP50.

For autonomous driving vehicles scenarios,in [27], designed a lightweight object detector. The researchers enhanced the detection speed by integrating the MobileNetv1 module with the feature extraction part of YOLOv4. To improve the model's ability to detect small objects, they replaced the default 13×13 prediction head with 140×140 head. At last, they used the k-means technique to improve the dataset and provide the network's primary anchor boxes. Despite a 2.66% decrement in mAP, this methodology reduces the model size by 76.3% and enhances the detection speed to 1.66 times that of the default model, making it suitable for real-time lightweight detection.

ALAD-YOLO is a lightweight apple leaf diseases detector designed by [28]. At first the researcher collected the diseases dataset and augmented it to enhance the generalization. To enhance the detection quality and reduce the computational burden, they replaced the YOLOv5 backbone with MobileNetv3 module and used a modified C3 module in model's neck.

To improve model's performance on various resolution images and avoid overfitting, they designed PPCSPC_GC block as a bridge between the model's backbone and neck. Finally, they used the CA attention technique for better detection accuracy. In comparison with YOLOv5s performance ALAD-YOLO enhances the accuracy by 7.9% and decreases the floating-point operations by 9.7G, achieving a good balance between the inference time and accuracy in edge environment. In [29] proposed a general purpose lightweight YOLOv3 version by simplifying its architecture and using the attention technique. The researchers simplified the model's architecture by applying the MobileNetv2 module as the model's backbone and using 3×3 depthwise separable convolution and SE blocks as a replacement of 3×3 heavy convolutional block in the model's head and neck. Then they improved the model's accuracy by using convolutional block attention modules in the modified neck, adding a non-local block to the MobilNetv2 module, and introducing the original images' high-frequency wavelets into the inputs. Finally, to improve the model's learning ability, they also used local 3×3 convolution branches in the backbone. Comparing the performance of this model with other models showed that it surpasses the other performances in one or more metric.

III. PROPOSED WORK

This paper focuses on modifying YOLOv10 architecture, particularly its backbone, by leveraging the key concepts of ShuffleNetv2 and MobileNetv2 lightweight modules. Shuffle blocks enhance the model's efficiency through group separable convolution and channel shuffling, while Mobile blocks employ inverted residual structure with depthwise separable convolutions and linear bottlenecks, both of which significantly reduce the computational overhead and represent the features efficiently [30, 31]. By replacing some of YOLO's original blocks with these optimized blocks, this work aims to develop a lightweight and accurate object detection model suitable for real-time applications running on edge environment. The following subsections are organized as follows: the first present an overview of the original YOLOv10 architecture, followed by dedicated subsections describing the proposed Shuffle and Mobile-Shuffle architectures.

A. YOLOv10 Architecture Overview

YOLO [32] is a state-of-art single-stage object detection algorithm designed for real-time applications. Unlike two-stage algorithms, such as R-CNN, which first propose objects' regions then refine and classify those objects in the second step, YOLO provides the predicted bounding boxes and their class probabilities through only single network pass. This property speeds up the detection while maintaining competitive accuracy, making YOLO an appropriate computer vision tool for various real-time applications. The YOLO architecture consists primarily of three parts which are the backbone, the neck, and the head. The backbone is responsible for extraction the inputted images' features and typically includes CNN blocks designed to achieve best detection accuracy and efficiency. YOLO's neck processes the previously extracted features often by using feature pyramid blocks to improve multi-scale detection capability. Based on the generated feature maps, the head provides the final object classification and bounding box regression. Over the years, many YOLO versions have been released. To improve detection accuracy, robustness, and computational efficiency, each new release introducing different architectural developments such as the introduction of innovative backbones, better prediction strategies, and models scalability and training optimizations [33, 34, 35].

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

YOLOv10 is one of the latest versions of YOLO series which was released in May 2024. It designed as real-time, end-to-end object detector with improved accuracy and efficiency. Unlike the previous versions, YOLOv10 excludes the need of non-maximum suppression in post-processing stage by developing a consistent dual assignment strategy, enabling fully end-to-end detection that aims to reduce inference time. Furthermore, to enhance the model's accuracy and efficiency YOLOv10 adopts a holistic model design strategy that modifies its structural components. A rank-guided block design, spatial-channel decoupled downsampling and lightweight classification head are introduced to minimize the inference time and reduce the computational complexity and overhead, while integrating the large-kernel convolutions and partial self-attention for enhancing the detection accuracy. YOLOv10 is provided in various model variants which are YOLOv10x (extra-large), YOLOv10l (large), YOLOv10b (balanced), YOLOv10m (medium), YOLOv10s (small), and YOLOv10n (nano) to fit different application deployments based on the available resources and the required detection accuracy, making YOLOv10 a robust and scalable object detector for a wide range of environment from resource-limited setups to high performance systems. This work modify the structural blocks of YOLOv10s and YOLOv10n to make them more suitable for edge devices due to their minimal parameter count and complexity [36, 37]. Fig. 1 shows the original architecture of YOLOv10.

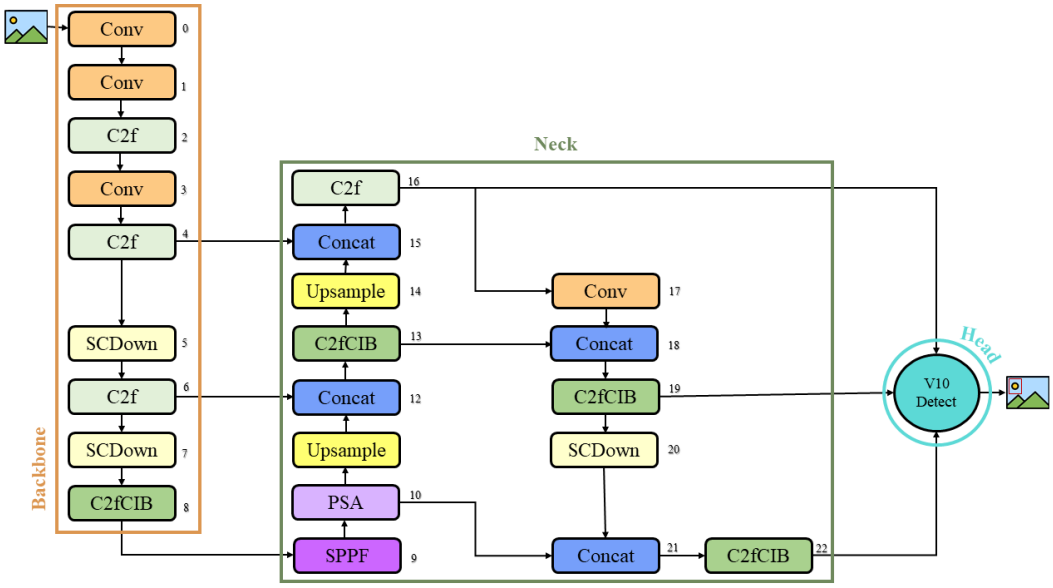


Figure 1: Standard YOLOv10 Functional Blocks.

In this proposed architectures, it replace some of the backbone blocks, which are Conv, C2f, and C2fCIB, with ShufflNetv2 and MobileNetv2 based blocks as will be explained later. The convolutional block (Conv) is a fundamental component in the YOLO structure, functioning as an independent block and as a part of more complex blocks. Conv block consists of 2D convolutional layer, 2D batch normalization, and a SILU activation function. C2f block is a faster version with two convolutions of cross stage partial bottleneck. By employing multilayer convolution and feature vector switching, C2f

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

blocks are capable of extracting features at multiple scales and expanding the scope of receptive domains. C2fCIB block is an enhanced version of C2f block, it uses the compact inverted block as a replacement to the traditional C2f's bottleneck, making it more efficient while preserving important features [38].

Each block replacement maintains the original input/output channel dimensions and downsampling strides, ensuring compatibility with the spatial sizes and channel counts expected by the model's neck and head. In particular, the proposed Shuffle and Mobile blocks are customized to replace the default blocks without altering the output shapes, number of channels, or downsampling steps. Each block in the modified backbone produces feature maps with the same spatial resolution required by the model's neck and head. As a result, no changes are necessary to the neck or head blocks, and no additional reshaping or padding is introduced. This compatibility is validated through successful model training, evaluation, and export to the ONNX format without errors or mismatches.

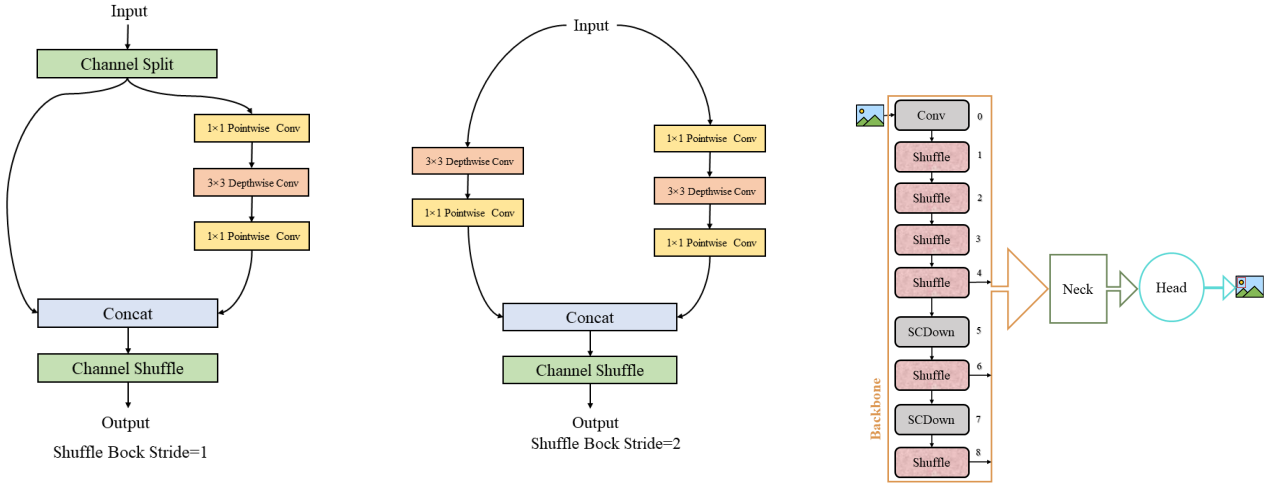
B. Shuffle Architecture

To enhance YOLOv10 for efficient edge deployment, this work incorporate custom-designed Shuffle blocks inspired by the main principles of the lightweight ShuffleNetv2 module. These Shuffle bocks optimize the model's efficiency by reducing the design redundancy while maintaining strong feature extraction capabilities. This is achieved through channel splitting, depthwise separable convolution, and channel shuffling. As shown in Fig. 2(a), when the input stride is set to 1, at first, the feature map is split along the channels into two equal parts. The first part bypasses any processing, serving as a shortcut path, while the other channels' part is processed through a 1×1 point wise convolution for dimension reduction, followed by a 3×3 depthwise convolution for spatial features extraction with minimal computations, and then another 1×1 pointwise convolution for dimension restoration. Each convolution layer is followed by batch normalization to stabilize training and ReLU activation function to introduce non-linearity.

lastly, the processed output is concatenated with the shortcut path, and the combined tensor is passed through a channel shuffle operation to reshape and permute the tensor dimensions to enable cross-channel information exchange. When the stride is set to 2, as illustrated in Fig. 2(b), the data processing flow is slightly changed to perform downsampling. in this case, the full input tensor is fed to two parallel paths, the first one is responsible for downsampling and dimension reduction using a 3×3 depthwise convolution followed by a 1×1 pointwise convolution, and the other path applies the same sequence of pointwise, depthwise and pointwise convolutions as in the stride 1 case. The outputs of each path are concatenated and then shuffles across channels to maintain a good features representation.

The integration of the proposed Shuffle block into the backbone of YOLOv10 structure is presented in Fig. 2(c), this work replace six default feature extraction blocks with the proposed lightweight blocks, more specifically, blocks number 1, 2, 3, 4, 6, and 8. This architecture fits well with requirements of on edge applications, such as drones and deep-space telescopes detectors, where memory and power constraints are critical and efficient, real-time inference is important. By embedding the Shuffle blocks into YOLOv10's backbone, it can achieve a balances model efficiency and detection performance, ensuring that the proposed Shuffle architecture is both fast and lightweight for real-time object detection in space-relates edge applications.

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



(a) (b) (c)

Figure 2: (a) Processing Flow of Shuffle Block When Stride = 1 (b) Processing Flow of Shuffle Block When Stride = 2 (c) Shuffle Architecture.

C. Mobile-Shuffle Architecture

In addition to evaluating the integration of the shuffle block into the YOLOv10 architecture, this work further examine another lightweight design by introducing a MobileNetv2-based blocks and propose a hybrid YOLOv10 backbone that includes both the previously described Shuffle blocks and the Mobile block. Due to its low parameter count and reduced computations, MobileNetv2 is widely adopted for resource-limited devices such as mobile and edge computing applications. The Mobile blocks are based on the inverted residual structure, depthwise convolution with linear bottleneck, the core features of the MobileNetv2 module architecture. The implemented Mobile block starts with an expansion layer representing by a 1×1 pointwise convolution that expands the number of channels by a defined factor to enhance the model's capability to capture richer features. After this layer, a 3×3 depthwise convolution is applied to process each input channel separately which leads to computation reduction compared to default convolutions. Finally, a 1×1 pointwise projection convolution layer that compresses the feature maps back to the needed output channel numbers. Each convolution is followed by batch normalization to improve the stabilize and accelerate the performance. ReLU6 activation function is used in each layer except the projection layer, which does not use a non-linearity, to improve the model performance especially on low-precision hardware. When the input and output channel numbers match and the stride is 1, a residual connection adds the original input to the processed features.

To create a more robust and lightweight backbone, this work combine the previously described Shuffle blocks with these Mobile blocks, introducing an effective hybrid architecture as shown in Fig. 3. These blocks are carefully inserted into the YOLOv10 structure, replacing the first four default blocks with Mobile blocks, and blocks 4, 6, and 8 with Shuffle blocks of the original backbone, in a way that remains compatible with the model's neck and head. This hybrid architecture enables

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

us to evaluate the performance impact of gathering the strength of both ShuffleNetv2 and MobileNetv2 modules to achieve an effective balance between model efficiency and feature expressiveness. ShuffleNetv2 provides strong computational efficiency through channel splitting, depthwise convolution, and channel shuffling, however, it may struggle in learning rich spatial features in deeper layers. On the other hand, MobileNetv2 uses inverted residual structure with linear bottleneck, which maintains spatial information and enhance feature reuse. By combining these two block types, the hybrid design leverages the speed and compactness of ShuffleNetv2 with the features representation capacity of MobileNetv2, making it a good option for edge tasks. To support this design choice, an additional architecture with Mobile blocks only was examined, but we found that it is less ideal for our lightweight requirement.

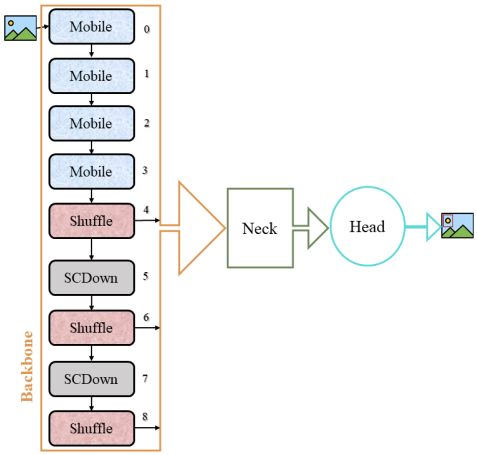


Figure 3: Mobile-Shuffle Architecture.

IV. Experiment Configuration

A. Dataset

The DeepSpaceYoloDataset dataset [11] consists of 4,696 annotated astronomical images, following the YOLO format. These images were captured by smart telescopes from locations across Luxembourg, Belgium, and France between March 2022 and September 2023. The dataset specializes in detecting Deep Sky objects (DSO) such as galaxies, nebulae, and globular clusters, which are often challenging to detect due to varying observation conditions like bright moon and light pollution. This paper utilizes an augmented version of DeepSpaceYoloDataset [12]. Through the deployment of precisely selected data augmentation methods, the enhanced dataset size is expanded to 8,421 images, each with resolution of 608×608 pixels. The augmentation process was designed to imitate the common changes observed in the deep sky imaging, including telescope motion and atmospheric fluctuations. Specifically, the methods applied include cropping and zooming up to 20% to simulate varying focal lengths, rotations from -15°to +15° to account for telescope orientation and Earth's rotation, brightness adjustments ranging from -15% to +15% to replicate lighting inconsistencies, blurring effects up to 2.5 pixels to mimic optical defocusing, and finally adding noise up to 0.15% of pixels to represent sensor artifacts.

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

This augmented dataset aims to improve models' generalization and robustness under realistic, noisy, and dynamic observational scenarios. At first, the entire dataset was used to leverage its full diversity, including objects with different levels of visibility and noise. The augmented images followed the same class distribution and were split consistently with the original dataset for comparative evaluation using a popular data splitting strategy, 80% of the images were used for model training, 10% for validation, and 10% for testing, then expanded the training set to include 7,486. After that, this work focused on a subset of 3,038 images where the DSOs occupy at least 25% of the image area, targeting the large objects. These images were extracted from the already split augmented dataset. Fig. 4 shows sample images of the DSOs included in the datasets.

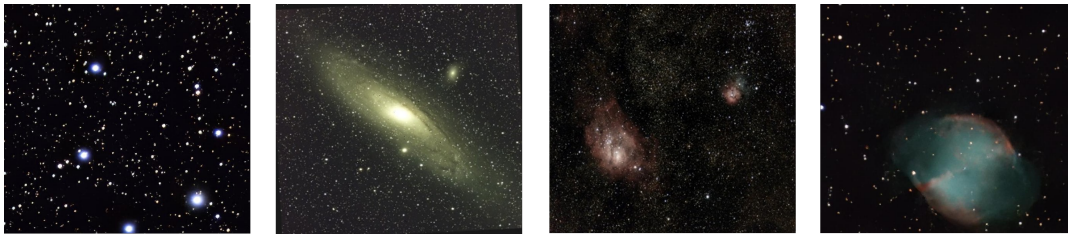


Figure 4: Sample Images for DSOs in the augmented DeepSpaceYoloDataset.

B. Implementation Details

To train and evaluate the proposed lightweight YOLO structures, we utilized the default training and validation frameworks provided by Ultralytics library, applying its default hyperparameters for implementing YOLOv10. At first, all models were trained for 100 epochs on the full dataset using the SGD (Stochastic Gradient Descent) optimizer with an initial learning rate of 0.01 and momentum of 0.9. After that, the pretrained models were retrained for an additional 100 epochs on the large space objects subset using the AdamW optimizer with an initial learning rate pf 0.02 and momentum of 0.9. Training was performed with a batch size of 16 and input images were set to 640×640 pixels. Standard YOLO data augmentation techniques such as mosaic and light albumentations were applied during training. Our proposed blocks were integrated directly into the YOLOv10 backbone definition in compatible way with the original neck and head modules. The both datasets were preprocessed to fit with the YOLOv10 input requirements. All the experiments were performed using Google Colab platform, a cloud-based environment equipped with NVIDIA Tesla L4 GPUs, which provides an optimal environment for better training and supports YOLO's hardware requirements. All the results to be analyzed were gathered during the evaluation stage using the platform's CPU to simulate the limited resources of edge devices.

C. Performance Metrics

To evaluate the proposed architectures and compare them with the original architecture, we consider a set of performance metrices that express both detection accuracy and computational efficiency. We use the mean average precision at intersection over union of 50% (mAP50) as a metric to measure the accuracy of the object detection. mAP50 is a widely used evaluation metrices in object detection task that expresses how accurate the model detects and localizes objects. It is obtained from

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

the average precision for each class (APx) based-on the precision-recall curve and then averaging over all classes following the below equations. Precision is:

$$P = \frac{TP}{TP + FP} \tag{1}$$

Recall is:

$$R = \frac{TP}{TP + FN} \tag{2}$$

The average precision for class x is given by:

$$AP_x = \int_0^1 P_x(R)\, dR \tag{3}$$

While the mean average precision at IoU of threshold 50% is given by:

$$mAP_{50} = \frac{1}{N} \sum_{x=1}^{N} AP_x \tag{4}$$

Box precision refers to the accuracy of the model's bounding box predictions, ignoring the number of missed detections. Inference time (msec/image) presents the average time required by the model to process a single image during inference. Lower inference time indicates better suitability for real-time detection applications, which is especially important for resource-limited edge environment. The size of each trained model is measured in kilobytes (KBytes) using the Open Neural Network Exchange (ONNX) format, which is widely used for deploying models across various platforms. Finally, Params indicates the model's total number of parameters. A smaller model size with fewer parameters is more efficient for operation, storage, and transfer in edge and embedded systems [39, 40].

V. RESULTS AND ANALYSIS

A. Full Dataset Case

At first, this work evaluate six YOLOv10 models on the full augmented DeepSpaceYoloDataset dataset, then it compare the performance of the proposed architectures with the performance of the original small and nano models. The results show an efficient balance between the detection accuracy and model compactness.

1) Comparison Between the Shuffle Architecture and Original Architecture:

   As mentioned previously the shuffle blocks aim to improve the detection speed and reduce the models' complexity while maintaining good detection quality, this is reflected in models' performance as will be discussed next. Compared to the original YOLOv10s, the Shuffle architecture achieves a slightly higher mAP50, an increase of 1.01%, and improves the inference speed from 476.4ms to 330.3ms per image. Additionally, the ONNX model size is reduced by 25.5% compared to the original, and the number of parameters drops from 8.07M to 6.20M, proving improved compactness. For the nano YOLO variant, YOLOv10n Shuffle also outperforms the original YOLOv10n, achieving a 0.9% increase in mAP50 and a 5.3% improvement in precision, while reducing inference time from 171.4ms to 128.1ms per image and compressing the model size from 8,248KBytes to 6,409KBytes. These findings demonstrate the effectiveness of the Shuffle architecture in optimizing model size, accelerating the inference, and enhancing object detection performance.

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

2) Comparison Between the Mobile-Shuffle Architecture and Original Architecture:
The Mobile-Shuffle architecture is evaluated against the original YOLOv10s and YOLOv10n variants to examine its performance in detecting deep sky objects in various scales. For YOLOv10s model, the Mobile-Shuffle architecture maintains the same mAP50 at 62% while minimizing the model size from 28,432 KBytes to 21,688 KBytes and lowering the parameters count from 8.07M to 6.33M. However, this efficiency resulted in a 49.2ms per image increase in inference time and a 7.7% decrease in precision. Comparing to YOLOv10n, the Mobile-Shuffle architecture improves mAP50 by 2.4% and precision by 6.3%, while reducing model size from 8,248KBytes to 6,545KBytes and parameters from 2.50M to 2.06M, all these improvements also come at cost of a moderate increase in inference time from 171.4ms to 218.1ms per image. The Mobile-Shuffle architecture proves its effectiveness in achieving a good balance between accuracy and compactness, especially in modifying YOLOv10n architecture.

B. Subset Case

This sub section demonstrates the models' evaluation based on the large space objects subset. The results show a significant improvement in detection accuracy among all models when compared to the full dataset. This enhancement is achieved because of that the larger objects occupying more pixels and offering richer features for the models to learn from.

1) Comparison Between the Shuffle Architecture and Original Architecture:
In the large deep sky objects scenario, the integration of the Shuffle architecture with the YOLOv10s improves the original model mAP50 from 80.8% to 81.8%, while reducing inference time by about 31.4% and compressing the model's size from 28,432KBytes to 21,173KBytes. However, the cost of these improvements is a decrease in precision by 7.2%. Similarly, comparing the original YOLOv10n and Shuffle based YOLOv10n shows enhancement in mAP50 from 77.9% to 79.6% and in precision from 90.6% to 91.5%, while accelerating the inference by approximately 25% and minimizing the size by 1,839kBytes. Based on the large object subset, the Shuffle architecture shows a good performance trade-off with better detection quality compared to the entire dataset, making it suitable for fast, lightweight deployments.
2) Comparison Between the Mobile-Shuffle Architecture and Original Architecture:
For the small YOLOv10 variant, the Mobile-Shuffle architecture maintains a comparable mAP50, differing by only 0.4%, with slight decrement in precision from 91.8% to 89.8%, while the model size is reduced from 28,432KBytes to 21,688KBytes and drops 1.73M parameter. However, the inference time increases by 38.3ms per image, indicating a trade-off between size, accuracy and speed. Mobile-Shuffle architecture also improves the detection quality of the original YOLOv10n by 2.2% mAP50, while slightly decreasing the precision from 90.6% to 89%. Furthermore, it drops 1,702KBytes from the model size and 439,408 parameters, all these at cost of 47ms/image increment in latency as a trade-off. These results demonstrate that the Mobile-Shuffle architecture improves detection accuracy and compactness, especially for nano models.

C. Summary

Table I provides a summary of the comparison between the original YOLOv10s and YOLOv10n models, their Shuffle versions, and the Mobile-Shuffle versions in terms of detection accuracy (mAP50), box precision, inference time per image (milliseconds per image), model size in ONNX format (Kbytes), and parameters count on both full dataset, which are highlighted by pink, and large objects subset, which are highlighted by blue.

TABLE I
A Comprehensive Performance Comparison

Model No.	Model Name	mAP50	Box (Precision)	Inference Time (msec/image)	Model Size ONNX (KBytes)	Number of Parameters
1	YOLOv10s	62%	79.10%	476.4	28,432	8,067,126
2	YOLOv10s Shuffle	63.1%	73.80%	330.3	21,173	6,200,278
3	YOLOv10s Mobile-Shuffle	62%	71.40%	525.6	21,688	6,334,006
4	YOLOv10n	58.1%	68.60%	171.4	8,248	2,496,998
5	YOLOv10n Shuffle	59%	73.90%	128.1	6,409	2,021,558
6	YOLOv10n Mobile-Shuffle	60.5%	74.90%	218.1	6,545	2,057,590
7	YOLOv10s	80.8%	91.80%	471.9	28,432	8,067,126
8	YOLOv10s Shuffle	81.6%	84.60%	323.6	21,173	6,200,278
9	YOLOv10s Mobile-Shuffle	80.4%	89.80%	510.2	21,688	6,334,006
10	YOLOv10n	77.9%	90.60%	172.6	8,248	2,496,998
11	YOLOv10n Shuffle	79.6%	91.50%	129.7	6,409	2,021,558
12	YOLOv10n Mobile-Shuffle	80.1%	89.00%	219.6	6,545	2,057,590

To inspect the reliability of the reported performance metrics, each model was evaluated multiple times under identical conditions. The results demonstrate that the mAP50, precision, parameters count and model size remain exactly the same across all runs, proving their stability and reproductivity. In contrast, inference time experiences some variability across runs, which cause by the dynamic nature of GoogleColab's shared environment, such as background resource usage and CPU scheduling.

Fig. 5 demonstrates a comparison across all models for all metrics. Fig. 5(a) shows that the Shuffle based models achieve the best mAP50 scores across both evaluation sets, recording the highest mAP50 of 81.6% by model no. 8, while the original YOLOv10s, model no. 1 and 7, have the highest precision in each set following by YOLOv10n Shuffle, model no. 11, with only 0.4% difference. Fig. 5(b), (c), and (d) demonstrate that Shuffle based models offer faster inference and greater compactness across all evaluated scenarios.

Fig. 6 illustrates samples of the deep sky object detection performed by our proposed models.

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

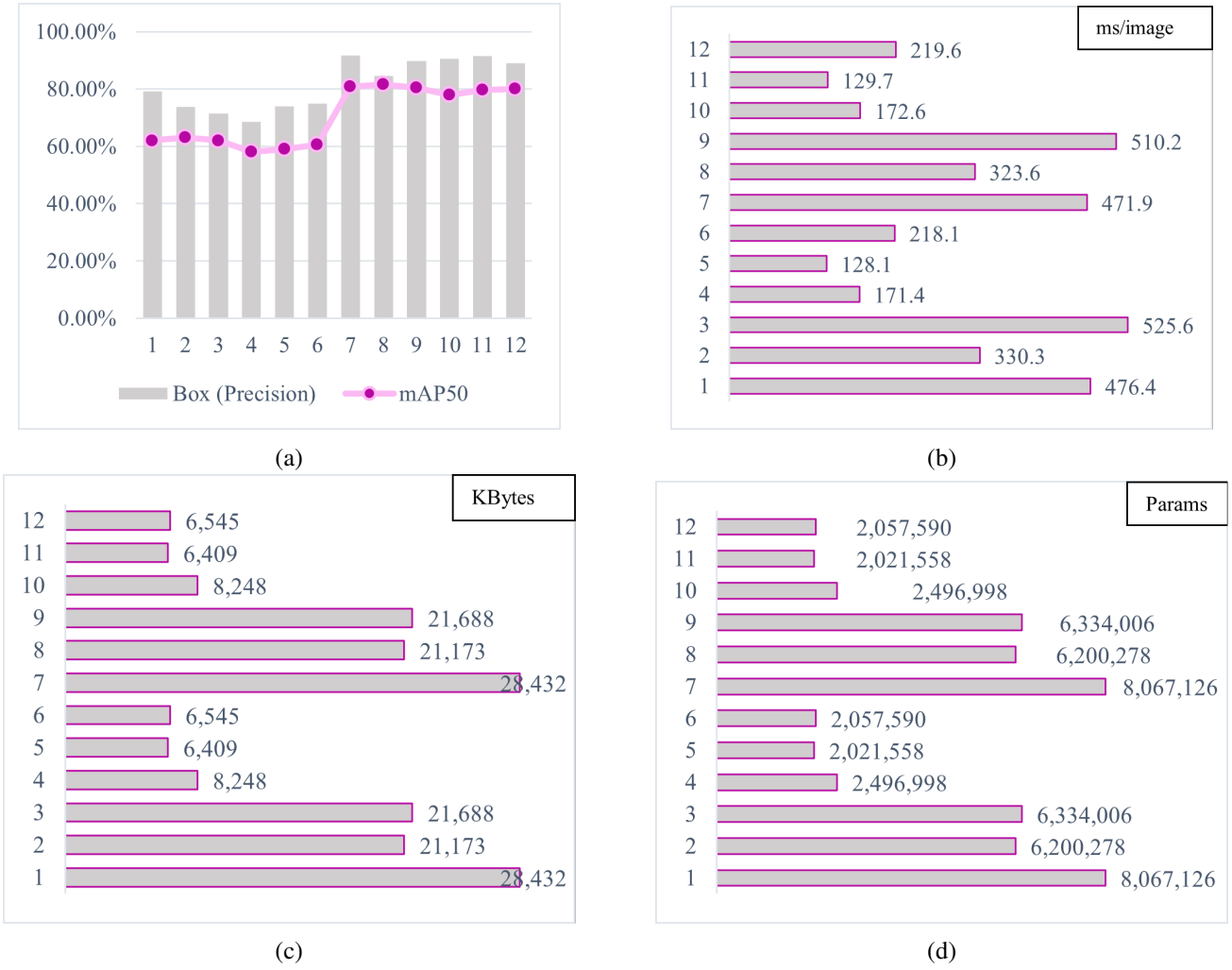


Figure 5: A Comparison Across All Models for Each Metric Separately (a) mAP50 and Precision (b) Inference Time (c) Model Size (d) Params.

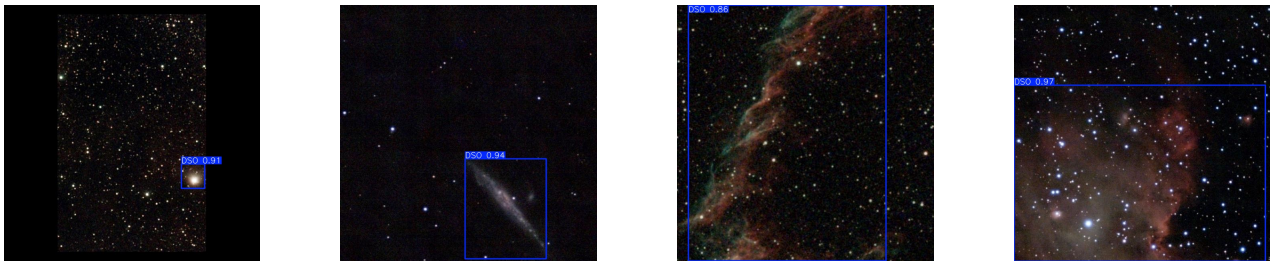


Figure 6: Deep Sky Object Detection Samples by Model No. (a) 2 (b) 6 (c) 9 (d) 11.

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

VI. CONCLUSIONS AND FUTURE WORKS

This study successfully optimizes YOLOv10s and YOLOv10n for better edge deployment specifically in case of deep sky object detection by integrating ShuffleNetv2 and MobileNetv2 inspired blocks into models' backbone, resulting in two efficient architectures, the Shuffle architecture and Mobile-Shuffle architecture. Based on the evaluation experiments on the full augmented DeepSapceYOLODataset dataset and its large objects subset, each model demonstrates different trade-offs between detection accuracy, inference time, and model complexity depending on the model variant. Across the full dataset, the Shuffle architecture constantly achieves faster inference and a smaller memory footprint, with minimal or even better effect on mAP50 and precision. Despite the latency increment, the Mobile-Shuffle based architecture shows a good trade-off by reducing model size and improving the detection quality, particularly in the nano variant. Otherwise, the evaluations based on the large space objects subset demonstrate that both architectures enhance the detection accuracy, this improvement proves the advantage of retraining with larger and well-defined features in supporting the learning efficiency. As a conclusion, in case of YOLOv10s the Shuffle models outperform Mobile-Shuffle across all the key metrics, including mAP50, inference time, and model complexity. In contrast, Mobile-Shuffle provides slightly better accuracy and precision for YOLOv10n at cost of increased inference latency. While the proposed lightweight YOLOv10 architectures show good potentials in space edge scenarios according to GoogleColab testing environment , future research can be in different directions, for example deploying and evaluating the models on real-world edge hardware such as NVIDIA Jetson devices to address practical deployment problems by providing a more comprehensive architectural analysis through exploring FLOPs, and FPS base on real hardware setups, which allow for deeper insight into computational efficiency and further validate the practical advantages of the proposed models under real-time constraints. Further model optimizations can be done by applying model compression techniques such as quantization and pruning to reduce the detection costs without significant accuracy loss. Additionally, enhance models' capabilities to detect small objects through multi-scale feature enhancements.

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] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," Multimedia Tools and Applications, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/s11042-022-13644-y.

[2] A. Setyanto, T. B. Sasongko, M. A. Fikri, and I. K. Kim, "Near-Edge Computing Aware Object Detection: A Review," IEEE Access, vol. 12, pp. 2989–3011, 2024, doi: 10.1109/ACCESS.2023.3347548.

[3] A. E. Abdelkareem, "Hardware considerations of a DSP based wireless coded receiver under limited resources," in 2022 International Conference on Intelligent Technology, System and Service for Internet of Everything (ITSS-IoE), IEEE, Dec. 2022, pp. 1–5. doi: 10.1109/ITSS-IoE56359.2022.9990939.

[4] C. Dong, C. Pang, Z. Li, X. Zeng, and X. Hu, "PG-YOLO: A Novel Lightweight Object Detection Method for Edge Devices in Industrial Internet of Things," IEEE Access, vol. 10, pp. 123736–123745, 2022, doi: 10.1109/ACCESS.2022.3223997.

[5] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.
[6] L. Guo, H. Liu, Z. Pang, J. Luo, and J. Shen, "Optimizing YOLO Algorithm for Efficient Object Detection in Resource-Constrained Environments," in 2024 IEEE 4th International Conference on Electronic Technology, Communication and Information (ICETCI), 2024, pp. 1358–1363. doi: 10.1109/ICETCI61221.2024.10594419.
[7] J. Xu, F. Pan, X. Han, L. Wang, Y. Wang, and W. Li, "EdgeTrim-YOLO: Improved Trim YOLO Framework Tailored for Deployment on Edge Devices," in 2024 4th International Conference on Computer Communication and Artificial Intelligence (CCAI), IEEE, May 2024, pp. 113–118. doi: 10.1109/CCAI61966.2024.10602964.
[8] A. Wang et al., "Yolov10: Real-time end-to-end object detection," in 38th Conference on Neural Information Processing Systems (NeurIPS 2024), Dec. 2024, pp. 107984-108011.
[9] X. and Z. H.-T. and S. J. Ma Ningning and Zhang, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design," in Computer Vision – ECCV 2018, M. and S. C. and W. Y. Ferrari Vittorio and Hebert, Ed., Cham: Springer International Publishing, 2018, pp. 122–138.
[10] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and 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.
[11] O. Parisot, "DeepSpaceYoloDataset: Annotated Astronomical Images Captured with Smart Telescopes," Data (Basel), vol. 9, no. 1, p. 12, Jan. 2024, doi: 10.3390/data9010012.
[12] Ramos, L. T., Rivas-Echeverrìa, F. (2025). Deep sky object detection in astronomical imagery using YOLO models: a comparative assessment. Neural Computing and Applications. https://doi.org/https://doi.org/10.1007/s00521-025-11223-4.
[13] M. Mastrofini, "Artificial Intelligence Techniques Applied to Onboard Space Navigation, Surveillance and Tracking," Ph.D. dissertation, Physics Department, Sapienza University of Rome, Rome, Italy, 2023.
[14] M. A. C. Silva, "Machine learning applied to deep space images," M.S. thesis, Engineering college, University of Porto, Porto, Portugal, 2022.
[15] P. Ferrini, "Ground based image generator for detection and tracking of space objects with machine learning techniques," M.S. thesis, Space Engineering, Politecnico university of Milan, Milan, Italy, 2024.
[16] Z. He, B. Qiu, A.-L. Luo, J. Shi, X. Kong, and X. Jiang, "Deep learning applications based on SDSS photometric data: detection and classification of sources," Monthly Notices of the Royal Astronomical Society, vol. 508, no. 2, pp. 2039–2052, Oct. 2021, doi: 10.1093/mnras/stab2243.
[17] Y. Yuan, H. Bai, P. Wu, H. Guo, T. Deng, and W. Qin, "An Intelligent Detection Method for Small and Weak Objects in Space," Remote Sense (Basel), vol. 15, no. 12, p. 3169, Jun. 2023, doi: 10.3390/rs15123169.
[18] Y. Guo, X. Yin, Y. Xiao, Z. Zhao, X. Yang, and C. Dai, "Enhanced YOLOv8-based method for space debris detection using cross-scale feature fusion," Discover Applied Sciences, vol. 7, no. 2, p. 95, Jan. 2025, doi: 10.1007/s42452-025-06502-7.
[19] Y. Xiao, Y. Guo, Q. Pang, X. Yang, Z. Zhao, and X. Yin, "STar-DETR: A Lightweight Real-Time Detection Transformer for Space Targets in Optical Sensor Systems," Sensors, vol. 25, no. 4, p. 1146, Feb. 2025, doi: 10.3390/s25041146.
[20] L. Zhou and X. Liu, "MDPruner: Meta-Learning Driven Dynamic Filter Pruning for Efficient Object Detection," IEEE Access, vol. 12, pp. 136925–136935, 2024, doi: 10.1109/ACCESS.2024.3464576.
[21] A. Kh. Al-Zihairy, and A. E. Abdelkareem, "Enhanced YOLOv8 for Edge Computing: A Deep Learning Approach for Memory Optimization" Iraqi Journal of Information and Communication Technology, vol. 8, no. 1, pp. 23.37, Apr. 2025, doi: https://doi.org/10.31987/ijict.8.1.283.
[22] X. Liu, T. Wang, J. Yang, C. Tang, and J. Lv, "MPQ-YOLO: Ultra low mixed-precision quantization of YOLO for edge devices deployment," Neurocomputing, vol. 574, p. 127210, Mar. 2024, doi: 10.1016/j.neucom.2023.127210.
[23] J. Chen, X. Yu, Q. Li, W. Wang, and B.-G. He, "LAG-YOLO: Efficient road damage detector via lightweight attention ghost module," Journal of Intelligent Construction, vol. 2, no. 1, p. 9180032, Mar. 2024, doi: 10.26599/JIC.2023.9180032.
[24] Z. Qiu et al., "YOLO-SDL: a lightweight wheat grain detection technology based on an improved YOLOv8n model," Front Plant Sci, vol. 15, Nov. 2024, doi: 10.3389/fpls.2024.1495222.
[25] H. Xu, Y. Zhang, and Y. Chen, "Research on vehicle detection algorithm based on YOLOv5s-ShuffleNetV2-SE-EIOU," in Eighth International Conference on Traffic Engineering and Transportation System (ICTETS 2024), X. Xiao and J. Yao, Eds., SPIE, Dec. 2024, p. 170. doi: 10.1117/12.3054703.
[26] L. Wang, J. Qin, K. Yan, H. Sun, S. Sun, and Y. Geng, "A Lightweight Insulator Defect Detection Model Based on YOLOv8," in 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), IEEE, Mar. 2024, pp. 2349–2353. doi: 10.1109/AINIT61980.2024.10581815.
[27] H. Wang and W. Zang, "Research On Object Detection Method In Driving Scenario Based On Improved YOLOv4," in 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), IEEE, Mar. 2022, pp. 1751–1754. doi: 10.1109/ITOEC53115.2022.9734559.
[28] W. Xu and R. Wang, "ALAD-YOLO:an lightweight and accurate detector for apple leaf diseases," Frontiers in Plant Science, vol. 14, Aug. 2023, doi: 10.3389/fpls.2023.1204569.
[29] S. Sun, X. Yang, and J. Peng, "Yolo-Based Lightweight Object Detection With Structure Simplification And Attention Enhancement," in ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, Jun. 2023, pp. 1–5. doi: 10.1109/ICASSP49357.2023.10097155.
[30] Y. Li, A. Li, X. Li, and D. Liang, "Detection and Identification of Peach Leaf Diseases based on YOLO v5 Improved Model," in 2022 The 5th International Conference on Control and Computer Vision, New York, NY, USA: ACM, Aug. 2022, pp. 79–84. doi: 10.1145/3561613.3561626.
[31] W. Gong, "Lightweight Object Detection: A Study Based on YOLOv7 Integrated with ShuffleNetv2 and Vision Transformer," arXiv preprint, arXiv:2403.0173, Mar. 2024.
[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
[33] X. Wang, H. Li, X. Yue, and L. Meng, "A comprehensive survey on object detection YOLO," in The 5th International Symposium on Advanced Technologies and Applications in the Internet of Things (ATAIT 2023), Kusatsu, Japan: CEUR Workshop Proceedings, Aug. 2023, pp. 77–89. [Online]. Available: http://ceur-ws.org

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

[34] R., Sapkota et al., "YOLOv10 to its genesis: a decadal and comprehensive review of the you only look once (YOLO) series.," arXiv preprint, arXiv:2406.19407, 2024.
[35] A. Vijayakumar and S. Vairavasundaram, "YOLO-based Object Detection Models: A Review and its Applications," Multimed Tools Appl, vol. 83, no. 35, pp. 83535–83574, Mar. 2024, doi: 10.1007/s11042-024-18872-y.
[36] M. A. R. Alif and M. Hussain, "YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain," arXiv preprint, arXiv:2406.10139, 2024.
[37] M. Hussain, "Yolov5, yolov8 and yolov10: The go-to detectors for real-time vision," arXiv preprint, arXiv:2407.02988, 2024.
[38] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review," arXiv preprint, arXiv:2501.13400, 2025.
[39] J. Zhao et al., "YOLO-Granada: a lightweight attentioned Yolo for pomegranates fruit detection," Scientific Reports, vol. 14, no. 1, p. 16848, Jul. 2024, doi: 10.1038/s41598-024-67526-4.
[40] A. He et al., "ALSS-YOLO: An Adaptive Lightweight Channel Split and Shuffling Network for TIR Wildlife Detection in UAV Imagery," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 17, pp. 17308-17326, 2024, doi: 10.1109/JSTARS.2024.3461172.

