Diyala Journal of Engineering Sciences

Journal homepage: https://djes.info/index.php/djes

ISSN: 1999-8716 (Print); 2616-6909 (Online)

# Lightweight Optimization of YOLO Models for Resource-Constrained Devices: A Comprehensive Review

Ula Kh. Altaie[1], A.E. Abdelkareem[1*], Abdullah Alhasanat[2]

[1] Department of Computer Networks Engineering, Al-Nahrain University, College of Information Engineering, Baghdad, Iraq
[2] Alhussein Bin Talal University, College of Engineering, Jordan

**ARTICLE INFO**

**Article history:**
Received: 24/06/2025.
Revised: 03/11/2025.
Accepted: 22/11/2025.
Available online: 10/12/2025

**Keywords:**
YOLO
Edge Computing
IoT
Lightweight Modules
Model Compression Techniques

**ABSTRACT**

The growing adoption of Internet of Things (IoT) and edge computing has increased the need for effective real-time object detectors that can operate in resource-constrained environments. YOLO (You Only Look Once), a leading state-of-the-art object detection algorithm, is well known for its remarkable real-time performance in a variety of applications. However, traditional YOLO models remain computationally heavy for resource-constrained environments since they are designed for high-performance systems, making them less practical for low-power, embedded platforms such as Raspberry Pi, ARM-based processors, and NVIDIA Jetson edge devices. This paper aims to investigate and analyse optimization strategies that enhance YOLO's efficiency for edge deployments and provides a comprehensive review of various optimization techniques to overcome the deployment challenges. Two main approaches are explored, structural modification using lightweight modules like ShuffleNet, MobileNet, and GhostNet, and model compression via knowledge distillation, quantization, and pruning. The reviewed works demonstrate significant reductions in model size and complexity, with generally enhanced inference speed and improving accuracy, however, in certain cases, a slight drop in accuracy and frame rate occurs as the cost of achieving higher efficiency. Structural modifications generally support model stability, efficiency, and generalization, while compression-based techniques further improve models' compactness and inference throughput. A combined or hybrid optimization strategy offers the most balanced solution, achieving strong detection accuracy with reductions in model size, GFLOPs, and overall inference cost. This narrative synthesis review provides guidance for developing scalable, energy-aware YOLO models suitable for edge-based detection applications in fields like autonomous vehicles, smart cities, and IoT-driven systems.

## 1. INTRODUCTION

The rapid development of the Internet of Things (IoT) and edge computing has accelerated the demand for efficient deep learning models capable of performing real-time object detection under limited computational resources. Edge devices such as drones, mobile robots, and embedded processors must analyze data locally, where latency, energy consumption and bandwidth restrictions present major problems. Although the YOLO (You Only Look Once) detector family has become a benchmark for real-time object detection, its deployment on resource-constrained devices remains challenging due to its computational complexity and high memory requirements [1].

Traditional YOLO architecture, generally optimized for GPU-based inference, are impractical for embedded devices with low-power capabilities. Therefore, optimization has been essential to adapt YOLO to the edge without sacrificing accuracy [2].

*Corresponding author's Email: ammar.algassab@nahrainuniv.edu.iq
DOI: 10.24237/djes.2025.18401

Recent research efforts have explored two main approaches, the first one includes architecture redesign through incorporating modules such as ShuffleNet, GhostNet, EfficientNet, attention methods and other lightweight structures as YOLO building blocks, as well as additional structural changes, for example replacing default loss functions. These modifications reduce computational operations and the number of parameters, resulting in minimized model size and faster inference times. The other approach is applying model reduction and compression methods such as pruning, quantization, and knowledge distillation, which minimize the computational resources memory usage while preserving accuracy. These Optimization approaches are important for real-world deployments in IoT-based systems, autonomous navigation, industrial inspection applications, and agricultural monitoring [3-5].

Despite significant research progress, most existing reviews address optimization strategies for general computer vision algorithms or domain specific applications and do not provide a comprehensive detailed analysis of YOLO optimization [6-8]. While some studies mention YOLO, there remains a gap in evaluations that integrate and compare architectural lightweighting and compression approaches specifically for YOLO deployment on resource-constrained settings. Most studies focus on individual optimization aspects or field-specific implementations, leaving a lack of unified insights into how both of those approaches affect YOLO's performance efficiency balance. Addressing this gap is critical for developing framework that guide YOLO optimization across diverse application while balancing accuracy, inference speed, and models compactness on edge computing platforms. This paper makes the following key contributions:

- It provides a comprehensive review of recent advances in lightweight YOLO optimization methods, focusing on both structural modification and compression techniques.
- It demonstrates a comparative framework of various strategies in terms of their performance efficiency, accuracy trade-offs, and suitability for deployment on resource-limited devices.
- It highlights hybrid optimization attempts that combine various techniques from the same approach or integrate lightweight architectural redesign with quantization, pruning, or knowledge distillation for enhanced real-time efficiency.
- It identifies unsolved research gaps in balancing model compactness, generalization, and accuracy across different application domains, and suggests directions for future research in edge-based object detection.

The rest of this paper is organized as follows. Section 2 establishes the necessary background on edge computing, the YOLO architecture, deployment challenges, and evaluation metrics. Section 3 reviews YOLO optimization strategies based on model structure modifications, while Section 4 examines model reduction and compression methods. Finally, Section 5 provides a comparative analysis based on target application fields and insights into research gaps, and Section 6 concludes the paper with directions for future work.

## 2. BACKGROUND

### *2.1 Edge Computing and IoT*

Affected by the fast growth of the underlying technologies, the Internet of Things (IoT) is becoming more and more embedded into our daily lives. Through incorporating complex network systems that allow communication between nodes, millions of devices and sensors keep generating data and exchanging critical information [9, 10]. According to [11] the total number of IoT unit installations globally is expected to rise from 13.8 billion to 30.9 billion by 2025, which means that huge amounts of data will be generated in a fast manner by sensors, actuators, and other smart devices. Such massive amounts of data will then subsequently require extra processing.

Due to its high cost-efficiency and flexibility, which are achieved through a combination of centrally controlled network management, storage space, and computing functions, cloud computing has been developed and employed extensively over the last few decades. The rapidly expanding Internet of Things and mobile internet applications generate severe challenges to the traditional centralized cloud computing architecture. These applications' connections to distant remote cloud servers put extra strain on radio access networks and backhaul networks, increasing latency. New challenges such as strict latency, capacity limitations, and enhanced security threats have come up with the development of information technologies and IoT, which makes the central cloud computing architecture an insufficient solution for them [12].

By putting the power of processing and storage closer between the end user and the cloud providers, edge computing is a model of distributed computing that brings cloud services to the edge of the network. It resolves the drawbacks of cloud-based architecture by providing mobility, lower bandwidth usage, and better response time. Applications incorporating large data streams, such as CCTV surveillance and autonomous cars, gain significantly from the integration of edge computing and IoT. IoT systems can minimize the amount of data that needs to be

forwarded to the cloud through performing data gathering, processing, and analytics locally via the deployment of computational resources at the edge. This strategy not only optimizes network utilization but also maintains data privacy and security by restricting critical data within local resources at the edge [13].

### *2.2 YOLO (You Only Look Once)*

The main focus of computer vision that has become widely popular is the real-time detection of the existence of single or multiple objects within a given frame combined with their classification and bounding boxes. To identify and locate an item within an image or a video, object detection can make use of neural networks. This capability has led to a significant improvement in various fields such as autonomous diving, security applications, facial recognition, robotics and so on. The object detection problem in computer vision has experienced a big shift with the introduction of the-state-of-art model YOLO. The first version of the YOLO algorithm was introduced and published by Redmon et al. in 2016 [14]. Over the last few years, twelve YOLO versions have been developed and released [14-25], each better than the previous in terms of inference time, accuracy of detection, and the ability to identify objects of different sizes [26-28].

Unlike models that require two or more stages to detect objects within an image, You Only Look Once (YOLO) is a single-stage detection model that detects and identifies objects through a single convolutional neural network (CNN) forward pass. Two-stage detection models, like R-CNN, use one stage for region proposals generation through selective search and another stage for classification and refining these regions through CNN, while YOLO outputs bounding boxes and class probability scores for each detected object within a specific image using its neural network by dividing the input image into a grid of cells, each of them responsible for object detecting inside its area. This design makes YOLO faster and more suitable for real-time applications compared to other models [29].

The backbone, neck, and head are the three primary parts of the YOLO architecture. YOLO's head, neck, and backbone designs might differ from one version to the next, and advancements in these parts have greatly enhanced the network's overall accuracy and speed. Depending on the targeted application and the intended trade-off, the backbone, neck, and head selection can impact the YOLO model's speed and accuracy. In order to further improve YOLO performance, the latest versions of YOLO have made enhancements to all three elements [30, 31].The YOLO Architecture is shown in Figure 1.

The backbone's main function is to extract important features from the input image. The backbone is typically a convolutional neural network that has been trained on large datasets such as COCO or ImageNet. From the input images, the backbone serves as the network that extracts features and produces feature maps. In the YOLO architecture, the neck acts as a bridge connecting the head and the backbone. The path aggregation network (PAN) and the spatial pyramid pooling (SPP) module are its two primary parts. The feature maps from the different backbone network levels are combined by the neck and sent to the head. The YOLO's head is in charge of processing the aggregated features and providing classification scores, objectness scores, and bounding boxes. The main responsibility of the head is to produce the network's final output, which consists of class labels and predicted bounding boxes [32, 33].

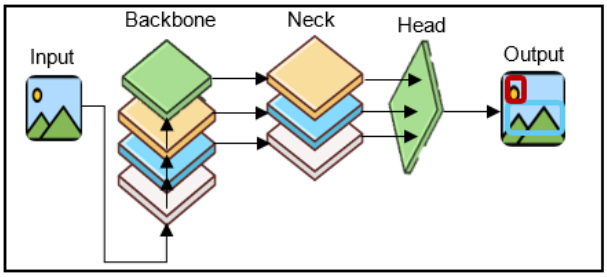


**Figure 1.** YOLO Structure

While advances in YOLO architecture have greatly improved detection accuracy and scalability, these enhancements also increase architectural depth, parameter count, and computational demand. Such complexity affects the feasibility of deploying YOLO on edge devices where processing power, memory, and energy are limited. Analyzing the relation between model architecture and deployment constraints is important to identify where optimization efforts should focus on [34, 35].

### *2.3 The Challenges of YOLO Deployment on Edge*

Since edge environments are inherently resource-constrained, deploying the original YOLO models on edge devices is extremely difficult. Several critical factors contribute to the challenges limit the feasibility of using standard YOLO models in such settings. First, the computational demands of YOLO models, particularly those with heavy backbones like CSPDarknet53, usually exceed the processing capability of most edge devices, creating a bottleneck that prevents these models from achieving real-time performance, which is necessary for many edge applications. Besides, the high memory footprint of these models often exceeds the limited memory of edge hardware, making the deployment of these models unrealistic without major modifications.

Another critical concern is power consumption. Most edge devices such as drones, IoT sensors, and mobile robots operate on very limited energy resources. In addition to the energy consumed by edge devices for various purposes such as communication and data transfer, especially in distributed IoT systems, the high energy usage of standard YOLO computations make it unfit for long term operations in such environments, as they can quickly drain the available power. Latency is another challenge, as the computationally intensive nature of YOLO models is often incompatible with the demands of real-time applications such as autonomous navigation, surveillance, and industrial automation. There's another layer of complexity since these YOLO models were originally designed for general-purpose object detection. While they give good performance over a wide variety of object detections tasks, domain specific fine-tuning is often required for applications, like small object detection in agricultural fields or defect detection in manufacturing processes. However, such fine-tuning increases computational and memory demands during training, and the resulting models may not always be efficient for deployment on resource-constrained edge devices, potentially limiting their real-time performance. Besides, YOLO models can be deployed on hardware accelerators such as TPUs and FPGAs, but this usually requires major changes in either the models or the hardware to make deployment possible, since they are not natively designed for such accelerators. Another problem with the use of YOLO models in edge use situations is overfitting. Most edge use cases involve smaller, more specialized datasets, which limit the generalizability of a model on new data by causing overfitting problems. This restriction may not guarantee reliability and dependability in real-world situations. To address these challenges, many researchers have attempted various combinations of approaches to enable efficient deployment of YOLO models on edge platforms [36-40].

### *2.4 Evaluation Metrics*

Evaluating lightweight YOLO models for deployment on resource-constrained devices, which will be discussed in the following sections, requires a comprehensive understanding of both accuracy and efficiency-based performance metrics. In such environments, the optimization goal is not only to maintain high detection accuracy but also to minimize computational complexity, memory usage, and inference latency. The following metrics are commonly adopted in recent studies to benchmark YOLO versions and their lightweight modifications [41, 42].

#### *2.4.1 Accuracy Metrics*

Accuracy in object detection models is typically measured via precision-recall analysis, represented by Average Precision (AP) and its mean across multiple classes or thresholds (mAP) [43].

Average Precision (AP): This metric measures the area under the precision-recall curve for a single class, providing an indicator of the model's performance accuracy and confident calibration.

Mean Average Precision (mAP): This represents the meaning of AP values across all classes and multiple IoU thresholds, this metric provides a more comprehensive assessment of detection robustness.

$AP_{50}$ and $mAP_{50}$: demonstrate the average precision and its mean at an Intersection over Union (IoU) threshold of 0.5, meaning a predicted bounding box is considered correct if its object intersection exceeds 50% with the ground truth value, these metrics are widely used in YOLO evaluations.

A higher value of these metrics indicates that the model achieves better localization and classification performance across different object scales. In lightweight YOLO optimization, maintaining a high AP and mAP despite structural simplification or compression is important, as a significant accuracy drop compromises the benefits of computational efficiency.

#### *2.4.2 Efficiency Metrics*

In edge computing environments, efficiency metrics determine a model's feasibility for fast inference on hardware with limited processing, memory, and energy capabilities [44, 45].

Inference Time (ms): It presents the average time the model needs to process a single image, this value is inversely related to FPS and critical for time-sensitive applications where latency must be low.

Frame Per Second (FPS): This measures how many frames the model can process per second. A higher FPS demonstrates faster detection and better suitability for real-time tasks such as surveillance, industrial monitoring, or autonomous navigation.

Model size (MB): This metric reflects the storage footprint of the trained model; smaller models are essential for deployment on embedded devices like Raspberry Pi or NVIDIA Jetson with limited flash and RAM storage.

Giga Floating-Points Operations (GFLOPs): It indicates the total number of floating-point operations for a forward pass through the model, lower GFLOPs reflect reduced computational complexity and energy consumption.

With compatible deployments, reducing model size and GFLOPs directly contribute to lower power consumption and faster inference. This becomes

particularly important in battery-powered IoT nodes or distributed edge environments.

### *2.5* YOLO *Optimization Approaches*

Despite the massive computational, memory, and power requirements to deploy object detection models on resource-constrained edge environments, particularly YOLO, these models continue to be essential for real-time applications in different domains. As a result, many researchers have focused on optimizing and simplifying YOLO models to fit into resource-limited devices with efficient performance. The optimization efforts are typically divided into two commonly used approaches that focus on different aspects [46, 47].

The first approach concentrates on modifying the model's structure by integrating more lightweight modules. This paper focuses on the optimization attempts which mainly using ShuffleNet, MobileNet, and GhostNet modules' concepts. These architectures are designed to be computationally efficient and lightweight while providing powerful feature extraction capabilities, they use lightweight procedures, such as depthwise separable convolutions or channel shuffling, instead of heavy traditional convolutional layers, resulting in fewer parameter requirements and faster inference time. These modules are also designed to be modular and scalable, allowing them to be integrated into the YOLO structure and combined in compatible way with other lightweight structures like attention methods to create lightweight efficient object detection models suitable for edge devices [48, 49].

The other approach uses model compression methods in order to enable YOLO models for the edge environment. Applying pruning, quantization, and knowledge distillation methods helps to make the overall model much simpler and lighter while keeping performance degradation low. This allows for the development of compact and efficient YOLO models that are quite suitable for resource-constrained edge devices, hence assuring real-time object detection capability in practical applications [50, 51].

By combining these two approaches, incorporating lightweight architectures and applying compression techniques, researchers achieve highly optimized YOLO models that maintain their accuracy and speed. These approaches further open the door for real-time object detection in a number of applications including IoT systems, self-driving cars, and surveillance. As the demand for edge AI grows, these optimization and reduction methods will be key to developing the next wave of resource-efficient and intelligent systems [52].

Despite these advances, a clear gap remains in integrating evaluations of both architectural and compression-based YOLO optimization strategies under resource-constrained conditions. To address this, the paper adopts a narrative synthesis review. Relevant works were collected from major databases, including IEEE Xplore, Scopus, and ScienceDirect, focusing on studies published from 2022 that exploring the YOLO optimization across different application domains, strategies, and deployment constrained environments. One of the selection criteria was to include studies that examine different innovative techniques and combinations within and between the two main optimization approaches, while also covering multiple application fields to evaluate how trade-offs between accuracy, efficiency, and inference speed are achieved. Selected works were analyzed to identify trends, performance trade-offs, and practical insights for deploying YOLO on resource-constrained devices. This provides a clear framework for understanding the impact of different strategies and highlights the gaps and potentials for future research. Furthermore, a comparative analysis based on application domains is presented in following section to illustrate the relations between optimization strategies, model efficiency, and detection performance.

## 3.MODEL ARCHITECTURE MODIFICATION TECHNIQUES

### *3.1 ShuffleNetv2*

ShuffleNetv2 represents a lightweight CNN module, basically developed to optimize efficiency while minimizing the computational overhead, making it very useful for resource-constrained devices like those used in edge computing. Proposed by Ma et al. in (2018) [53], ShuffleNetv2 refines its predecessor by fixing computational bottlenecks through its arrangement of the splitter which groups the channels, depth-wise convolution and channel shuffling, as seen in Figure 2, as a result ensuring both good accuracy and high inference speed. The data flow in this module follows a split, transform, and the merge process, where input maps are divided into two branches, one applies depthwise and pointwise convolutions, while the other is used as a shortcut. These branches are then concatenated and shuffled to enhance information exchange between channels. This core concept achieves balanced computation, reduced memory, and efficient feature representation, which makes ShuffleNetv2 one of the top candidate modules for real-time and low-power deployments. With these benefits, ShuffleNetv2 provides a strong foundation for many studies focused on designing lightweight and efficient models, including adaptation of the YOLO framework for edge computing-based objects detection.

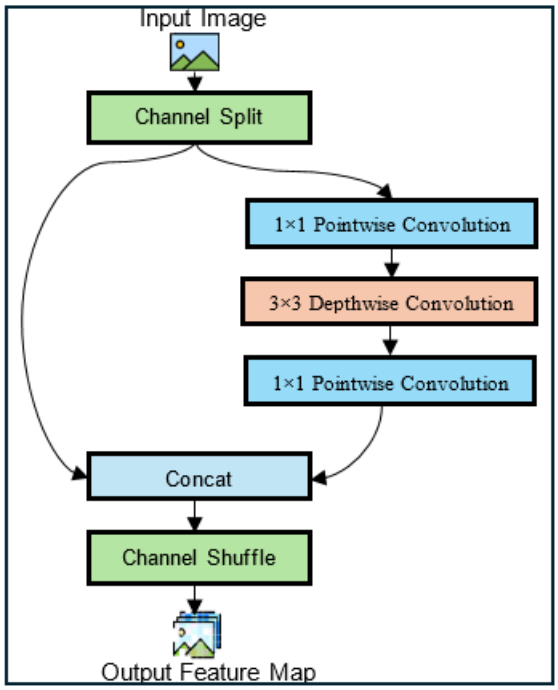


**Figure 2.** Building blocks of ShuffleNetv2

Yassir and Ahn [54] proposed an enhanced metallic surface defect detector that made use of a modified YOLOv5 model. They integrated a ShuffleNetv2-based lightweight backbone into the model for further improvement in real-time performance of steel strip surface detection specialized for small networks with limited resources. They highlighted the reduction in computational complexity due to the replacement of traditional convolutional blocks with depthwise convolution and channel shuffling building blocks, providing considerably better performance in terms of accuracy and speed. Their model was more efficient compared with other models, achieving a mAP of 70.18% for the CG10-DET dataset and 77.5% mAP for the NEU-DET dataset with 61.64% less computational time compared with the original YOLOv5 model for both datasets. This consequently proves the proposed model's effectiveness in detecting small, complex defects.

Wang et al. (2023) [55] introduced S2F-YOLO, a lightweight model specialized for fish detection and classification in challenging conditions. Based on the original YOLOv5 model, S2F-YOLO's backbone was changed with the ShuffleNetv2 module to reduce the processing demands while maintaining reasonable accuracy. Focal loss was used in the proposed model to overcome class imbalance, guaranteeing consistent training and improving the model's accuracy. The combination of the ShuffleNetv2 module and the spatial pyramid pooling fast (SPPF) resulted in model's speed enhancements. The collected fish dataset was improved by applying data augmentation techniques like mix-up and mosaic. The researchers developed a high-speed model with a frame rate ten times higher than the original YOLOv5x, with only a 2.24% $AP_{50}$ loss and 95.5% and 92.87% reductions in GFLOPs and parameters, respectively.

To optimize the performance of intelligent object detectors on mobile platforms with resource limitations, Gong [56] suggested an enhanced lightweight object detection model based on YOLOv7, combiningShuffleNetv2 and Vision Transformer (ViT). The researcher used two proposed modules, which are the Dynamic Group Convolution Shuffle Module (DGSM) and the Dynamic Grouped Convolution Shuffle Transformer (DGST). The original backbone of YOLOv7 was enhanced by introducing the DGSM module, which included group convolution and channel shuffling techniques to greatly increase the computational efficiency while ensuring high model performance. The YOLOv7 neck was replaced with the DGST module, which in its turn combined group convolution, channel shuffling, and vision transformer techniques together to improve effectiveness and a simplify the network structure. The experiments on a dataset of masked and unmasked individuals showed that the optimized model provided a more balanced solution for GPU usage and loss, with a significant 66.39% reduction parameter count compared to the YOLOv7 Tiny performance. Thit resulted in lower model complexity, a 51.73% shorter inference time, and a 0.2% $mAP_{50}$ improvement.

Electronic bicycles present a safety danger when they enter elevators, because of that Su et al. [57] introduced a lightweight electric bike detector suitable for edge devices within elevators. The researchers designed an optimized YOLOv5s version and used a ShuffleNetv2 module as its backbone to minimize computational operations and model size, making it more appropriate for edge deployment. The model's ability to express captures' features was improved by inserting the Swin Transformer technique into its layers. Additionally, for better feature extraction capability the SimAM attention technique was deployed at the backbone's last layer. To balance the performance accuracy, complexity, and inference speed, they also modified the model's neck and predictor with more lightweight modules and techniques. The evaluation tests on a collected dataset showed that the introduced model achieved 95.5% detection accuracy with only a 0.9% mAP reduction compared to the YOLOv5s. It also provided 84.81% less computation load, an 81.03% reduction in model size, and only 1.1 million parameters with a rate of 106 frames per second, providing a potential design suitable for edge computing-based detection applications.

In the world of edge computing models' optimization, He et al. [58] introduced ALSS-YOLO, a modified architecture focused on detecting small wildlife targets in thermal infrared (TIR) images captured by unmanned aerial vehicles (UAVs). The researchers designed a lightweight and adaptive channel split and shuffle module inspired by the concepts of the ShuffleNetv2 architecture, which improved the feature extraction and information exchange between channels, especially for small objects that are blurry or overlapped. Besides, the ALSS-YOLO integrated a lightweight coordinate attention (LCA) module for spatial information integration enhancement, a single-channel focusing module for infrared images feature representation efficiency improvement, and a modified function for localization loss for accurate small object localization. Their model showed superior performance on the BIRDSAI and ISOD TIR UAV wildlife datasets, reaching better mAP values than default YOLOv8 models that used as their baseline with fewer parameters, as well as a good frame rate per second.

### *3.2 MobileNet*

MobileNet refers to a family of CNNs developed for effective mobile and edge computing detection applications, first presented by Howard et al. [59].The design of MobileNet relies on depthwise separable convolutions, a technique that makes the computation more efficient and the model small by splitting the convolution process into two parts, depthwise and pointwise convolutions. This technique makes MobileNet a good option for real-time tasks on devices with limited resources by achieving high performance while remaining computationally efficient. The design has been continuously improved from the first version introduced previously, as follows:

MobileNetv2 [60]: The second version introduced the concept of inverted residual blocks and linear bottlenecks, improving the model's efficiency and thus allowing it to go deeper without generating huge computational costs. MobileNetv2 was designed to perform more accurately on mobile devices as a lightweight network.

MobileNetv3 [61]: This version further optimized MobileNet by using neural architecture search (NAS) and squeeze-and-excitation modules. MobileNetv3 offers better accuracy and higher performance compared with the second version, especially for real-time applications with limited computational resources.

MobileNetv4 [62]:The most recent version of the MobileNet family introduced the universal inverted bottleneck (UIB), a mobile multi-query attention (MQA) attention module, and an advanced NAS module. For mobile applications with constrained resources, MobileNetv4 provides better performance at a reduced computational cost and with smaller model size.

Yu et al. [63] and Li et al. [64] evaluated the effects of integrating the first three versions of the MobileNet module as backbones for YOLO models to determine the optimal architecture for their real-time object detectors, which were applied in complex agricultural environments.

In the research on detecting sugarcane stem nodes based on YOLOv5s model by Kang et al. [63], MobileNetv2 was the most efficient backbone, providing a good balance of the accuracy and the computational efficiency. Compared with the default YOLOv5s model, the MobileNet v2-YOLOv5s model reduced the model complexity by about 40% in GFLOPs, 34.5% in model size, and 35.7% in parameters and increased the detection speed by 195% with only a 0.8% AP decrement. Finally, a TensorRT-based test also proved the enhanced performance pf the MobileNet v2-YOLOv5s on an embedded device, the Jetson Nano, compared to the original model.

Similarly, MobileNetv3 showed the best performance for the analysis of pineapple maturity in the case of Li et al. [65] MobileNet V3-YOLOv4 research due to the enhanced squeeze-and-excitation modules, leading to increased accuracy and faster inference speed. As a result, the detection performance using MobileNet V3-YOLOv4 showed a 4.49% AP increase for the semi-ripe stage and a 6.06% AP increase for the ripe stage. Compared to the original YOLOv4 model, this model reached a 5.28% higher mAP, a frame rate improvement by 99.29%, and a 77.99% reduction in parameter size. Therefore, the model fits in with the complex real-time environment.

The MobileNet-CA-YOLO model, an enhanced version of the YOLOv7 model, was proposed by Jia et al. [56] for real-time rice disease and pest detection. This model used a MobileNetv3 module as the backbone with the aim of reducing the parameter count and the computational complexity in order to meet the performance requirements of resource-constrained mobile devices. To enhance the model's capability of detecting pests and diseases in a more efficient manner, the researchers applied the coordinate attention (CA) technique into the feature fusion layer. To avoid overfitting, the SIoU loss function was utilized which helped in improving the model's robustness and generalization and accelerate its convergence time. The proposed model was evaluated using 3773 images of rice pests and diseases. The results showed that the model achieved a 0.9% $mAP_{50}$ enhancement and a 62.71% higher

FPS, while the parameter count was significantly reduced by 81.31%, all that with only 28% GPU memory consumption. Because of these results, the proposed approach can be appropriate for use in agricultural monitoring systems that operate in real time.

Lang et al. [66] and Qiaomei et al. [67] modified the structure of the YOLOv5s model using a similar style to develop lightweight, real-time object detection models for industrial and agricultural fields, respectively. The researchers used the MobileNetv3 module and an attention technique within the model's backbone and integrated the SIoU loss function into their architectures.

MR-YOLO is a lightweight version of the YOLOv5s model specialized for surface defect detection of industrial magnetic rings [66]. The researchers adopted the lightweight MobileNetv3 module as their model backbone to significantly reduce the number of parameters while maintaining the detection's speed and accuracy. For better feature selection capabilities, they also integrated the SE attention technique into the model's backbone as a replacement for the SPPF module. To increase the model's robustness and generalization, random noise was added to the training data, and mosaic data enhancement techniques were applied. The CIoU loss function was replaced by the SIoU to enhance the algorithm's regression speed and locating accuracy. Based on the experimental tests, the proposed model achieved a 16.6% detection speed enhancement, parameters reduction by 47.9%, and a model size decrement by 48.1% with only 40.6% of the original floating-point operations while maintaining its mAP with only a 0.3% loss. Because of these improvements, MR-YOLO is a good option to use as real-time, resource-constrained industrial applications.

YOLO-MobileNet-CBAM is a lightweight plant pest detection model that designed to address the problem of agricultural pest detection's low accuracy in changeable early-stage, small target detection [67]. YOLOv5s's backbone was replaced by the lightweight MobileNetv3 module for parameters computation reduction. To achieve better accuracy in small target detection, the CBAM attention technique was inserted to enhance both the channel and spatial dimensions' essential feature extraction. The researchers substituted the H-SiLU activation function for SiLU in the convolutional module of the default model to speed up training and prevent gradient vanishing. To further enhance the accuracy of small target localization, the prediction box regression loss function was used to account for shape loss by using the SIoU function rather than the GIoU function of the default model. Lastly, for identifying large-scale diseases, small diseases, and pest targets, thus improving the small targets detection accuracy, 4 detection heads of different scales were fed by the feature pyramid. According to the results, the suggested methodology improved the frame rate from 59.7 FPS to 72.43 FPS and decreased the operation from 15.9 GFLOPs to 6.4 GFLOPs in comparison to YOLOv5s with a 92.38% accuracy rate. The research offers a good method for handheld terminal detection applications in the agricultural field by accomplishing a lightweight real-time model design while successfully increasing detection accuracy.

A rebar counting application based on YOLOv4, a rectified MobileNet lightweight feature pyramid network (RM-LFPN-YOLO), was introduced by Liu et al. [68]. The model's lightweight backbone used the MobileNetv2 module, modified by the addition of the CA technique specifically to its inverse residual module. This improved the model's capability to handle dense targets' spatial location information and detections. LFPN was developed as a lightweight module due to its depthwise convolution and pointwise convolution usage. This helped to increase the accuracy and efficiency of the detection by more effectively using the shallow, mid, and deep feature maps' semantic information for better multi-scale object detection capabilities. The positive-to-negative samples ratio in the rebar dataset restored balance by using the focal loss function to modify the confidence loss. Furthermore, when working with densely packed targets, the EIOU loss substituted the default CIOU loss for regression, which improved localization performance. While minimizing the parameters size by 89.79%, the introduced model showed an enhancement in average precision of 1.57% when compared to the original YOLOv4. It also had a 33.14% higher frame rate and an 87.49% significant decrement in processing complexity. The introduced lightweight model worked effectively for rebar recognition and counting and was suitable to be implemented in an embedded environment.

Although space observation is typically related with longer processing times, the use of YOLO addresses the increasing demand for onboard, real-time analysis in edge environments, such as small satellites or smart telescopes. In such systems, transmitting large volumes of data to central ground stations is impractical, making real-time detection and data filtering at the edge essential. Because of that, we previously proposed a lightweight optimization of YOLOv10 for deep space object detection [69]. The research introduced two modified architectures by incorporating ShuffleNetv2 and MobileNetv2 inspired blocks into the YOLOv10 backbone to minimize computational cost and parameter count

while maintaining detection accuracy. The first architecture is the Mobile-Shuffle architecture that incorporated mixed optimized blocks in its backbone, which are Shuffle and Mobile blocks, while the second architecture is the Shuffle architecture that replaced some of the original backbone blocks with Shuffle blocks only. The evaluations based on the augmented DeepSpaceYOLODataset showed, when comparing the proposed architectures with the original YOLOv10s, that the Mobile-Shuffle architecture achieved 23.72% and 21.48% reductions in model size and parameters, respectively, with similar detection accuracy in terms of $mAP_{50}$ but at the cost of about 10% slower inference. While the Shuffle architecture obtained 25.53% and 23.14% reductions in model size and parameter count, respectively, with a 1.1% $mAP_{50}$ improvement and 30.67% acceleration in detection speed.

*3.3 GhostNet*

GhostNet is a novel CNN module that was designed by Han et al. [70] to optimize object identification performance while improving computational efficiency. The Ghost module is GhostNet's key innovation, which creates numerous ghost feature maps from a set of intrinsic feature maps that are produced from expensive and heavy computations as shown in Figure 3. The ghost feature maps are derived by applying multiple cheap, low-cost linear transformations that utilize fewer parameters and operations compared to traditional convolutional operations. Similar to residual blocks in ResNet [71], the architecture uses Ghost bottlenecks, which are stacked Ghost modules to facilitate effective information flow through the GhostNet. The adaptability and efficiency of GhostNet have enabled its integration as a foundation of any model for lightweight vision tasks, as shown below with YOLO.

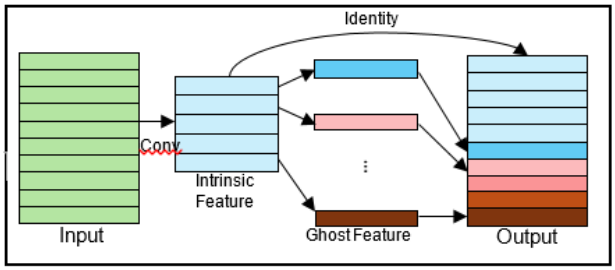


**Figure 3.** Ghost Module

Cao et al. [72] developed the GCL-YOLO, a lightweight object detection model designed for small and low-pixel targets in UAV pictures, to solve the issues of complicated surrounding environments, dense distribution, and the existence of different scales. the researchers adopted a GhostConv-based backbone instead of the default YOLOv5 backbone. The proposed backbone deployed the depth-separable convolution on one half of the channels and after that concatenated the original feature channels for better redundant feature usage. this greatly lowered the number of parameters without impacting the accuracy of the model. The researchers removed the default model's head that was used for large object detection in a normal environment and then inserted a new head specialized for small object detection. In the end, Focal-EIOU loss was incorporated for localization enhancement in datasets that were unbalanced. When GCL-YOLO was tested on the VisDrone-DET2021 and UAVDT datasets, it used 76.7% fewer parameter count and 32.3% less computations than YOLOv5s. it also achieved $mAP_{50}$ improvements of 6.9% and 1.8% on the two dataset, respectively. These improvements came at the cost of an 11.67% lower frame rate. Despite this trade-off, the model remains a good choice for UAV applications that require small object detection.

LAG-YOLO is an enhanced real-time object detection model designed for road damage detection. In order to develop a lightweight and accurate model, Chen et al. [73] designed the attention ghost module and used it as a replacement for the C3 module within the backbone and the neck of the YOLOv5 model. They replaced the traditional convolution with the ghost module to accelerate the detection process and reduce the number of parameters, while to enhance the accuracy of the detection they inserted the SimAM attention technique into the module. Moreover, to achieve multi-scale feature fusion, the researchers used the FPN + PAN structure for better feature extraction capabilities. Based on the RDD2020 and Hualu datasets, LAG-YOLO showed good mAP results compared to the original YOLOv5 model. It achieved it with only 4.19 million parameters and 32.74 GFLOPs, at cost of reduced speed of 50 FPS. This model can be used as real-time road condition monitor in in intelligent transportation systems and vehicle-mounted devices due to its efficient performance and lightweight design.

Yan et al. [74] introduced Ghost-YOLONet, a lightweight object detection model designed for resource-constrained and real-time applications. To reduce the model's size and complexity, the researchers modified the YOLOv4 backbone by replacing the CSPDarkenet53 module with the GhostNet backbone. They also combined the Ghost module with a decoupled fully connected attention technique to enhance the model's global feature extraction abilities. Lastly, they modified the model's neck to enhance its efficiency by optimizing the SPP structure to SimSPPF and used the deep separable convolutions instead of PANet's 3*3 convolutions. According to tests on the PASCAL VOC 2007 and

VOC 2012 datasets, the introduced model obtained 81.4% $mAP_{50}$ with an 81.5% model volume reduction and an 81.1% reduction in model size. Additionally, the frame rate increased by 9.12% FPS. These outcomes showed that the model could be used for competitively accurate real-time detection tasks on embedded devices.

YOLOX is a high-performance, anchor-free model that is separate version created outside of the original YOLO series by Megvii Technology researchers [75]. Wang and Yu [76] proposed RD-YOLO, an improved YOLOX model to solve the problems like inefficient model calculations, poor detection accuracy, and high miss rate of the road object detection in complicated environments. In order to speed up the detection process and greatly reduce the complexity of the model, the researchers inserted Ghost convolutions instead of the default 2D convolutions in the backbone and neck of the YOLOX model. Then, for better feature extraction abilities, an attention technique, which was the multidimensional collaborative attention (MCA) module, was incorporated into the model's backbone to remove the irrelevant spaces and the noise in the background. Moreover, a branch to detect high resolutions was included, which enhanced the accuracy of the detection and achieved better small targets' feature extraction. Lastly, to enhance the localization accuracy and achieve faster algorithm convergence, they removed the default loss function of the base model and added MPDIoU boundary regression loss function instead of it. Based on the BDD100K dataset and tested on an embedded device, which was the Jetson Nano, the evaluation results showed that RD-YOLO's performance exceeded that of many popular two-stage and single-stage object detection models in terms of detection accuracy, increasing the mAP by 2.1%, while achieving a 39.8% increment in FPS, with 49.8% and 51.76% reductions in parameter count and GFLOPs, respectively. The deployment of the proposed model on the limited resources proves the proposed model effectiveness as road object detector in real-world situations.

Lie et al. [77] combined the features of MobileNetv3 and GhostNet modules to develop a lightweight and efficient biometric human identification model based on human ear recognition. At first, the backbone of the base model was changed with the lightweight MobileNetv3 module. Inspired by the features of the GhostNet module, the researchers replaced the default C3 and Conv blocks with the C3Ghost and GhostConv modules in the neck of the base model. The experiments on EarVN1.0, USTB, and CCU-DE datasets demonstrated that the developed model maintained high ear recognition accuracy, while requiring less computations, smaller model size, and better frame rate compared to the base model, which was YOLOv5. For example, on the CCU-DE dataset, the introduced models achieved 68.61%, 77.44%, and 69.63% reductions in model's size, GFLOPs, and parameter count, respectively, with only a 0.2% $mAP_{50}$ drop and a 116.42% FPS increase. This methodology showed a good balance between the accuracy of the model and its efficiency, which makes it appropriate for real-time biometric identification applications in the resource-constrained scenarios. Table 1 summarizes the reviewed papers that using model modification techniques.

## 4. MODEL REDUCTION AND COMPRESSION TECNIQUES

### *4.1 Quantization*

Despite the fact that neural networks have led to major advancements in several domains, their computational cost is a challenge. With the goal of integrating modern networks into edge devices with strict power and computation constraints, it is essential that neural network inference be possible with less power and lower latency. Model quantization is one of the most effective techniques for reducing computation and memory costs, providing a practical trade-off between accuracy and efficiency. It works by representing the network's weights and activations with lower bit precision, typically reducing from 32-bit floating-point to 8-bit integers, which can decrease memory usage by a factor of 4 and computational cost of matrix multiplication falls quadratically by a factor of 16. Quantization is generally applied in two main ways, Quantization-Aware Training (QAT) which includes quantization effects during training for higher accuracy, and Post-Training Quantization (PTQ) which applies quantization after training for faster deployment. However, its effectiveness can vary depending on the hardware support, as some edge devices may lack low-precision operations. Research has been demonstrated that neural networks are generally flexible to quantization, which implies that their accuracy of the results is not significantly affected by quantization to lower bit widths [78-80].

**Table 1.** Summary of The Reviewed Papers That Using Model Architecture Modification Techniques

| Ref | Year | Use Case | Optimized YOLO Version | Effective Used Techniques |
|---|---|---|---|---|
| [54] | 2023 | Metallic Surface Defect Detector (Industrial Applications) | YOLOv5 | ShuffleNetv2 module |
| [55] | 2023 | Fish Detection and Classification (Smart Fishing Applications) | YOLOv5 | ShuffleNetv2 module, focal loss function |
| [56] | 2024 | Smart individual Detectors on Mobile Platforms (Safety Management Applications) | YOLOv7 | ShuffleNetv2 module, and ViT method |
| [57] | 2024 | Electric Bike Detection in Elevators (Safety Management Applications) | YOLOv5s | ShuffleNetv2 module, swim transformer technique, SimAM attention method, lightweight GSConv and VoV-GSCSP modules in the neck and fast converging and EIOU error function in the predictor |
| [58] | 2024 | Small Wildlife Targets Detection in Thermal TIR Images Captured UAVs (UAV Applications) | YOLOv8n | ALSS Module (based on channel splitting and shuffling), single channel focusing module, LCA method, and FineSIOU |
| [63] | 2023 | Sugarcane Stem Nodes Detector (Agricultural Application) | YOLOv5s | Examine MobileNetv1, MobileNetv2 and MobileNetv3 modules |
| [64] | 2023 | Pineapple Maturity Detector (Agricultural Application) | YOLOv4 | Examine MobileNetv1, MobileNetv2 and MobileNetv3 modules |
| [65] | 2023 | Rice Disease and Pest Detector (Agricultural Application) | YOLOv7 | MobileNetv3 Module, CA technique, and SIoU loss function |
| [66] | 2022 | Surface Defect Detector of Industrial Magnetic Rings (Industrial Applications) | YOLOv5s | MobileNetv3 module, SE technique, CIoU loss function, and adding noise to training data |
| [67] | 2023 | Plant Pest Detection (Agricultural Application) | YOLOv5s | MobileNetv3 module, CBAM technique, H-SiLU activation function in convolutional module, SIoU loss function, and 4 multi-scale heads outputted through the feature pyramid |
| [68] | 2024 | Rebar Counting Application (Industrial Applications) | YOLOv4 | MobileNetv2 module, CA technique, LFPN (lightweight module based on depthwise and pointwise convolutions), and focal and EIOU loss functions |
| [69] | 2025 | Deep Sky Object Detector (Space object Aware montoring) | YOLOv10 | ShuffleNetv2, and MobileNetv2 |
| [72] | 2023 | Small And Low-Pixel Targets Detector in UAV Pictures (UAV Applications) | YOLOv5 | GhostConv-based as the backbone, replace large scale detection head by specialized one for small object detection, and Focal-EIOU loss |
| [73] | 2024 | Road Damage Detector (Transportation Safety and Autonomous Vehicles Applications) | YOLOv5 | Attention ghost module (based on ghost module and SimAM technique), and FPN + PAN structure |
| [74] | 2024 | Smart Object Detectors for Embedded Devices (General Purpose Applications) | YOLOv4 | GhostNet module combined with a decoupled fully connected attention technique, SimSPPF, and deep separable convolutions |
| [76] | 2024 | Road Object Detection (Transportation Safety and Autonomous Vehicles Applications) | YOLOX | Ghost convolutions, MCA module, a branch to detect high resolutions, and MPDIoU boundary regression loss function |
| [77] | 2023 | Biometric Human Identification Model Based on Human Ear Recognition (Security Applications) | YOLOv5s | MobileNetv3 module, and C3Ghost and GhostConv modules (base on the features of the GhostNet module) |

Gunay et al. [81] proposed LPYOLO, a low-precision TinyYOLOv3-based model for real-time face detection on FPGA-based edge devices. With the goal of minimizing the model latency and power consumption with an acceptable level of accuracy, the researchers used the quantization-aware training (QAT) method which takes quantization into account during training process. They also set up FPGA's on chip memory as the network parameters storage, replaced the last activation function with Hard-Tanh, and configured FPGA's logical resources were set to a high level of parallelism. The model was implemented on the PYNQ-Z2 board with a Xilinx Zynq 7020 SoC using the WiderFace dataset. The results of the 4-precision model showed that the proposed model achieved an accuracy rate of 75.7% mAP with only 0.8% accuracy drop on the easy category when compared with the unquantized model, a throughput of 18 FPS with a 50 times faster inference speed on PS+PL run, and a system power consumption decreased to only 2.4 watts, making the system suitable for low-power and fast face detection applications.

Due to the limited feature information provided by infrared-based images compared to full-coloured ones, detecting objects within the infrared captures is challenging. Cui et al. [82] examined various techniques to optimize YOLOv5 for better ship detection results based on infrared inputs. At first, the researchers combined the MobileNet module with the YOLOv5 to create more lightweight model, this combination led to 26.57% reduction in parameter size with only a 5% drop in accuracy. To further optimize the model, they also examined quantization techniques, including QAT and static quantization, which reduce the precision of both weights and activations by using calibration data. The experiments showed that despite the loss in detection accuracy caused by the quantization techniques, a 6% drop with QAT and an 8% drop with static quantization, they greatly decreased the size of the model parameters by 80.07% compared to the original YOLOv5 model.

To cover the challenges faced by YOLO deployment on limited resource devices, Wang et al. [83] designed Q-YOLO, an effective one-stage object detection model. The researchers investigated low-bit scale quantization techniques to avoid the problem of performance deterioration produced caused by imbalances in activation distribution found in traditional quantization. They combined a fully end-to-end post-training quantization (PTQ) technique with an activation quantization technique, which was unilateral histogram-based (UH). The UH activation quantization technique used histogram analysis through lowering the quantization mean square error (MSE) to find the maximum truncation values. The designed methodology was applied to different variants of YOLOv5 and YOLOv7 on various CPU and GPU platforms. The evaluations on the COCOval20217 dataset showed that their quantization technique performed better than other PTQ techniques by maintaining the accuracy of detection while greatly reducing the memory and the processing requirements. For example, comparing the performance of YOLOv5 with its 8-bit quantized version on an i9-10900(×86) CPU using OpenVino library showed a 75% reduction in model size and about 31% improvement in detection speed with only a 0.1% drop in AP, while on GPU-based platforms the quantized models were three times faster.

YOLO-GD is a lightweight object detector designed for real-time applications on embedded devices, specifically Jetson Nano devices mounted on robots for recycling empty dishes, proposed by Yue et al. [84]. The researchers used GhostNet as a replacement for YOLOv4's original backbone and applied depthwise separable convolutions instead of traditional convolutions to decrease the computational overhead. Then, the model was quantized by FP16 post-training TensorRT quantization to improve the detection performance without further accuracy loss. Based on experiments, the quantized YOLO-GD model outperformed YOLOv4 by 3.45% in mAP and reached 97.42%, while the inference time was decreased by 84.25%, achieving 30.53 FPS, and a power consumption of only 5-10 watts. The YOLO-GD obtained 82.2%, 82.5%, and 88.68% reductions in model size, parameter count, and FLOPs, respectively, making the proposed model a practical option for resource-limited environment.

A unique ultra-low mixed-precision quantization technique called MPQ-YOLO was introduced by Liu et al. [85] for facilitating YOLO deployment on edge devices with limited resources, specifically YOLOv5. To achieve a remarkable balance between the detection precision and efficiency, the researchers used a 4-bit quantized head and a 1-bit quantized backbone. They combined the model with two training methods, which were the progressive network quantization (PNQ) and a trainable scale, to connect the head and backbone of different precisions to achieve a full quantization training. To reduce the oscillation generated by the mixed precision training, the PNQ technique was applied for smoother training and quicker model convergence, while a trainable scale was deployed to 4-bit head's weights and activations for efficient gradient propagation. In comparison to full-precision model, the introduced model showed major decreases in model size of 54% and computational complexity of 78%, with only

about 8% loss of accuracy, making MPQ-YOLO appropriate for real-time applications in resource-constrained edge environments.

*4.2 Pruning*

As a way to accelerate inference time and enable the implementation of modern, large language models in resource-limited environments, pruning methods have gained interest as a neural network compression research area [86]. These methods are generally categorized into structured and unstructured pruning approaches, each one with its specific contribution. Unstructured pruning focuses on eliminating individual weights, while structured pruning eliminates entire networks or components like filters or channels. This method has attracted interest since it can reduce both the size of the model and the computing power needed while maintaining performance, making it a suitable solution for edge devices [87-88].

To satisfy the requirements of deploying deep learning models in resource-limited environments, Joen et al. [89] introduced the target capacity filter pruning (TCFP) methodology to optimize YOLOv5 for embedded systems. At first, the researchers performed sparsity learning to find the parameters and computations target numbers based on the pre-trained model. To help the pruning process in achieving these target numbers, they used a saliency score $\gamma$ in the pruning, which derived from the size of the batch-norm layer's scaling factor. To optimize the pruned model for guaranteeing hardware compatibility and improving inference speed, they deployed an 8n extension process during model pruning. Lastly, according to the model architecture produced by the pruning process, they initialized parameter values and retrained the model. Different pruning rates were evaluated on PASCAL VOC, COCO, and VISDRONE datasets. The evaluation of the introduced pruning methodology on the NVIDIA Jetson Xavier NX platform using the PASCAL VOC dataset demonstrated that when 30%, 40%, and 50% of parameters and FLOPs were pruned, the inference time enhanced by 14.32%, 26.41%, and 34.47%, respectively, while the mAP dropped by 0.6%, 2.3%, and 2.9%, respectively, and GFLOPs reduced by 28.53%, 39.44%, and 46.16%, respectively.

In a different perspective, Ahn et al. [90] proposed SAFP-YOLO, a methodology that integrating spatial attention techniques and filter pruning for accelerating the YOLO object detection model with minimal loss in accuracy. The researchers assessed the importance of each filter based on the difference in the feature maps before and after the application of the convolutional block attention module (CBAM) and coordinate attention (CA) attention mechanisms and then pruned the unimportant ones. This methodology was applied to YOLOv4 and YOLOv7 models using Argoverse-HD vehicle dataset. The results from tests on an NVIDIA Jetson TX-2 low-cost embedded board with a pruning rate of 87.5% demonstrated that the inference speeds of the SAFP-YOLO-based models increased by up to 435.9%, and the parameter count was reduced by 70.1%. The proposed methodology still maintained acceptable accuracy with an average drop of 5.75% in $mAP_{50}$ and is very suitable for real-time applications in limited resource environments, such as mobile devices and embedded devices on the edge.

An enhanced version of YOLOv5, ShuffleNetv2-YOLOv5-Lite-E, was designed by Zhang et al. [91] for edge devices and was intended for real-time identification of tea leaves with one bud and two leaves. To minimize the model's size, the researchers eliminated the focus layer, inserted the ShuffleNetv2 module in the model instead of the traditional conventional feature extraction section, and applied channel pruning to the neck layer's head. This methodology resulted in a 73.02% reduction in model size compared to YOLOv5. The designed model achieved a mAP of 94.52% on a Raspberry Pi edge device only a 1.75% drop from the original model, and a detection speed 68.6% faster than YOLOv5 reaching 8.6 FPS. These outcomes make ShuffleNetv2-YOLOv5-Lite-E a good option as a real-time agricultural management application on devices with limited resources.

To develop an efficient dense vehicle detector appropriate for limited resources edge devices on autonomous driving vehicles, Zhou et al. [92] proposed MP-YOLO. The researchers combined various techniques in order to optimize YOLOv8. A hierarchical feature fusion (HFF) module and a multi-scale feature fusion block (MSFB) module were integrated into the model for improving its capacity to handle multi-scale features, while a 160*160 scale head was included to enhance its ability of detecting small objects. the WIOU loss function was used to solve the road targets high overlap problem. At last, the layer adaptive sparsity for magnitude-based pruning (LAMP) technique was selected as the pruning technique to minimize the size of the model. The evaluation results based on the DAIR-V2X dataset showed that the size, parameter count, and GFLOPs of the proposed model decreased by 63.33%, 70.63%, and 13.58%, respectively, with a 4.7% $AP_{50}$ improvement at cost of an 11.76% reduction in frame rate when compared to YOLOV8n performance. MP-YOLO outperformed other traditional models in speed and accuracy, making it a good option for real-time use in autonomous driving systems.

To optimize the effectiveness of the object detection models without sacrificing their performance Zhou and Liu [93] introduced a filter pruning technique for environments with constrained resources, which is called MDPruner. As opposite to the previous reviewed static pruning techniques, MDPruner is a dynamic technique with the ability of modifying the pruning rate according to the input image's complexity. A higher pruning rate is applied to prune computing costs for simpler images, whereas a lower pruning rate is applied for complex images to preserve detection accuracy. The need for extra parameters was eliminated using a supernet framework to simultaneously train and optimize several pruned subnetworks of different sizes. Each image's detection complexity was automatically determined by the use of a meta-network, a lightweight module that was deployed to choose the best subnetwork depending on the image complexity. Extensive tests on COCO and PASCAL VOC datasets using YOLOv3 and YOLOv7 revealed that the suggested model could outperform competing techniques like LCP, ThinNet, and HRank. For instance, the MDPruner-0.5 test on the COCO dataset with YOLOv3 achieved a 46.63% reduction in parameters and a 58.85% reduction in GFLOPs, with only a 1.2 mAP loss compared to the original YOLOv3 performance.

### *4.3 Knowledge Distillation*

One of the commonly used model compression methods is knowledge distillation. As introduced in [94] the fundamental idea of knowledge distillation is the process of passing the knowledge from a larger, deeper, and more complicated model (teacher) to a smaller, simpler model (student). Through the training of the student model, this method uses the teacher's soft probabilities (logits) as a soft target to guide the students' learning trend. This method demonstrates that the student model can perform at the same level or even better than the teacher model, despite its compactness and computational efficiency [88, 95].

In order to further enhance the efficiency and deployability of YOLOv4, Xing et al. [96] presented DD-YOLO, a lightweight and effective object detection model that combined knowledge distillation with differentiable architecture search (DARTS). First, based on the COCO2017 dataset, the best cell computations were found using the DARTS search approach. After that, the YOLOv4 backbone was modified using the stack of the searched cells in order to minimize the number of parameters. Additionally, the knowledge distillation was used to increase the detection accuracy by teaching the student model, which was the previously modified YOLOv4, by the ResNet101 as the teacher model. The results on the COCO2017 test-dev datasets proved that the used methodology could reduce the number of parameters by 61.4% and GFLOPs by 97.5%, and improve the mAP by 2.4%, making the presented model appropriate for mobile environments.

For reliable, real-time, lightweight, and mobile tuna detection tool, Liu et al. [97]introduced Tuna-YOLO based on YOLOv3. To minimize the total number of computations and parameters, MobileNetv3 was first integrated as the backbone of the model after evaluating the performance of several lightweight backbones. Second, for further feature extraction capability enhancements, a CBAM attention module was substituted for the SENET module. The model accuracy was then improved by using knowledge distillation technique, when the default YOLOv3 was the teacher model and the modified model was the student model. The researchers produced a small dataset from the fishing boats' monitoring videos, labelled the collected data, and applied k-means algorithm for better detection accuracy. According to the evaluation of the introduced model performance compared to the default YOLOv3 model performance, the results showed that Tuna-YOLO improved the detection accuracy by 6.11% $mAP_{50}$, while decreasing 62.32% of the parameter size, and improving the detection speed from 10.12 FPS to 15.23 FPS, with only 8.676 GFLOPs. Because of these improvements, Tuna-YOLO was suited for mobile and edge implementations, especially in fisheries management for electronic monitoring.

Lyu et al. [98] suggested YOLO-HPFD, a customized lightweight object detection model for distinguishing between litchi blossom genders in natural orchard circumstances. In order to optimize the detection power of the student model base on YOLOv4 tiny, the researchers used the multi-teacher pre-activation feature distillation (MPFD) technique employing YOLOv4 and YOLOv5l as teacher models. In the design of the distillation loss, the distillation location prior to the activation function was optimized. The integration of the LogCosh-Squared distillation loss function, margin-activation method, and the convolution and group normalization (Conv-GN) structure enhanced the distillation process by improving the student's ability to learn and extract meaningful features from the teachers, resulting in a 4.42% mAP improvement over the original YOLOv4-tiny. Lastly, the distilled model was then quantified and deployed on an FPGA embedded device, achieving a 73.85% reduction in model size and 93.9% power saving, with only a 0.47% mAP drop and inference speed of 6FPS. These outcomes demonstrated the YOLO-HPFD efficiently meets the

requirements for accurate and energy-efficient litchi gender detection in practical agricultural applications. PG-YOLO, a lightweight, fast, and accurate object detection model tailored for mobile and edge devices in industrial internet of things (IIoT) environments, was proposed by Dong et al. [99]. In order to achieve a lightweight base model, the researchers first substituted all YOLOv5s's convolutional modules using ghost bottleneck. After that, they selectively deleted the ineffective components of the model's backbone, and for additional model compression they applied the R-pruning technique. Lastly, they deployed the knowledge distillation technique to increase the detection quality in order to provide high level of accuracy despite applying multiple compression techniques. When evaluated on the safety-helmet-wearing (SHWD) dataset and deployed on NVDIA Jetson TX2, PG-YOLO reduced the model size by nine times and parameter count by 91%, while the accuracy ended up 93.4%. Additionally, the speed of inference accelerated by 32.72% with 10 FPS. This research shows how lightweight architectures, pruning, and distillation can be combined to produce effective models for edge devices with limited resources.

Huang et al. [100] developed a lightweight YOLOv8s version designed as an industrial ceiling fan detector, combining pruning and knowledge distillation techniques to enhance efficiency without compromising detection accuracy. First, the researchers applied linear decay to dynamically modify the sparsity factor through training, which improved the model's channel representation and increased the pruning effectiveness. This technique eliminated redundant parameters and compressing model size. Then, to recover the detection quality they introduced a dual loss function combining channel-wise knowledge distillation and bridging cross-task protocol knowledge distillation to enhance knowledge transfer from teacher to student models by using the intermediate and output features. Experimental results on their collected dataset demonstrated that the optimized model achieved a 76.6% reduction in parameters and an 85.2% in computations, while maintaining a high 97.6% mAP and an inference speed of 91.4 FPS. This methodology successfully balances model compactness and high detection accuracy, making it suitable for industrial defect detecting application.

Table 2 summarizes the reviewed papers that using model reduction techniques.

## 5. DISCUSSION

This section analyzes and compares the performance outcomes of the optimization studies reviewed in this paper. The selected works are organized according to their application domains, including industrial inspection, UAV and infrared-based applications, safety management, autonomous systems, and general-purpose detection. Each table below summarizes the base model for comparison, dataset used, the evaluation setup, and changes in accuracy, inference speed, and model complexity. The analysis focuses on the capabilities of architectural modification and compression methods to provide effective trade-offs between accuracy and efficiency, making the models suitable for real-time use on low-power or embedded devices.

The selection of datasets across studies illustrates the diversity of target application fields. Publicly available datasets such as COCO, PASCAL VOC and VisDrone are frequently used for general-purpose detection and UAV imagery analysis, enabling standardized, and comparable model evaluation. Meanwhile, many researchers use custom or domain-specific datasets, like agricultural crop imagery and industrial defect datasets, to simulate real-world scenarios for specific applications. Models trained on custom datasets often achieved higher precision due to the controlled environment, while those evaluated on public datasets demonstrated better generalization in different scenarios. Moreover, the reviewed works differ in their evaluation setups and deployment environments. Some studies deployed models on real edge hardware with limited resources, while others evaluated models' performance via standard computing environments or simulated testing for potential real-world deployment. Although all studies provided experimental findings, the differences in hardware configurations and testing setups sometimes affect the measured performance metrics. This variety in datasets and evaluation environments highlights that lightweight YOLO architectures maintain consistent performance accuracy, while also proving the importance of the deployment conditions in determining the overall evaluation across both controlled and real-world scenarios.

**Table 2.** Summary of The Reviewed Papers That Using Model Reduction and Compression Techniques

| Ref | Year | Use Case | Optimized YOLO Version | Effective Used Techniques |
|---|---|---|---|---|
| [81] | 2022 | Face Detection on FPGA-Based Edge Devices (Security Applications) | Tiny YOLOv3 | Quantization-aware training technique |
| [82] | 2022 | Ships Detection in Infrared Images (Ships Infrared-Based monitoring Application) | YOLOv5 | MobileNet module, and examine quantization techniques (quantization-aware training and static quantization) |
| [83] | 2023 | Smart Object Detectors on Limited Resource Environments (General Purpose Applications) | YOLOv5 YOLOv7 | Combined the fully end-to-end post-training quantization technique with unilateral histogram-based activation quantization scheme |
| [84] | 2022 | Recycle Empty Dishes Model on Embedded Devices (Recycling Robots) | YOLOv4 | GhostNet module, depthwise separable convolutions, TensorRT FP16 post-training quantization |
| [85] | 2024 | Smart Object Detectors on Limited Resource Environments (General Purpose Applications) | YOLOv5 | Ultra-low mixed-precision quantization technique (4-bit quantization head and 1-bit quantization backbone combined with PNQ and a trainable scale) |
| [89] | 2022 | Smart Object Detectors for Embedded Devices (General Purpose Applications) | YOLOv5 | Target capacity filter pruning methodology |
| [90] | 2023 | Vehicle Detectors on Limited Resource Environments (Transportation Safety and Autonomous Vehicles Applications | YOLOv4 YOLOv7 | Integrating the spatial attention mechanisms and filter pruning technique |
| [91] | 2023 | Tea Leaves with One Bud and Two Leaves Detector (Agricultural Application) | YOLOv5 | ShuffleNetv2 module, channel pruning, and removing the focus layer |
| [92] | 2024 | Dense Vehicle Detector (Autonomous Vehicles Applications) | YOLOv8 | HFF and MSFB modules, 160*160 scale head, WIOU loss function, and layer adaptive sparsity for magnitude-based pruning technique |
| [93] | 2024 | Smart Object Detectors on Limited Resource Environments (General Purpose Applications) | YOLOv3 YOLOv7 | MDPruner (dynamic filter pruning technique) |
| [96] | 2022 | Smart Object Detectors on Mobile Environments (General Purpose Applications) | YOLOv4 | Knowledge distillation, differentiable architecture search, and modifying the backbone using the stack of cells |
| [97] | 2023 | Mobile Tuna Detector (Smart Fishing Applications) | YOLOv3 | MobileNetv3 module, CBAM module, knowledge distillation technique, and applied k-means algorithm to their dataset |
| [98] | 2023 | Litchi Blossom Genders Detector in Natural Orchard Circumstances (Agricultural Application) | Tiny YOLOv4 | Multi-teacher pre-activation feature distillation technique, optimizing the distillation location, LogCosh-Squared distillation loss function, margin-activation method, and the convolution and group normalization structure |
| [99] | 2022 | Smart helmet Detectors on Limited Resource Environments (Industrial and Safety Applications) | YOLOv5s | Ghost bottleneck, selectively deleted the ineffective components of the model's backbone, R-pruning technique, and knowledge distillation |
| [100] | 2025 | Industrial ceiling defect detector (Industrial Application) | YOLOv8s | Pruning and knowledge distillation |

In the industrial inspection domain, the main objective is to accurately identify small industrial defects, count products, and monitor workers safety, where high precision and fast inference are critical to maintain operation within the manufacturing process. Researchers consistently replaced heavy backbones with lightweight options like ShuffleNetv2 [54] and MobileNetv3 [66]. To prevent the potential loss of feature details, attention mechanisms, such as SE [66] and CA [68], were integrated to enhance extraction capabilities for small targets. Furthermore, advanced loss functions, such as SIoU [66] and EIoU [68], were employed to enhance localization accuracy. Hybrid strategy combined lightweight modules with pruning and knowledge distillation [99] or combining multiple compression methods [100] to enhance efficiency further. This domain shows some of the best balanced optimizations.

Researchers achieved high mAP improvements, 22.55% improvement on NEU-DET [54], while simultaneously accelerating detection speed by a 61.64% drop in detection time [54] and minimizing model computational load, including a 91% less parameters [99] and 87.49% reduced GFLOPs [68]. The accuracy loss was negligible, about 0.6%, proving that lightweight models could be both fast and highly accurate for industrial applications, as illustrated in Table 3. RM-LFPN-YOLO [68] achieved the best outcomes within the industrial inspection domain, achieving 99.03% AP and 79.73 FPS while reducing parameters size by about 90% and model complexity by 87.49% in GFLOPs. This balance was obtained due to the combination of MobileNetv2 and CA attention method with an efficient
LFPN for improving the detection of densely small targets like rebars, making it more deployable on edge environments.

The applications based on UAV and infrared imagery face many challenges. Detecting objects from aerial platforms involves handling small image pixels and complex backgrounds,
while infrared images add more difficulties by reducing the rich information available in RGB images.

**Table 3.** Industrial Inspection Applications Summary (↑: Increment, ↓: Decrement)

| Ref | Baseline Model | Dataset | Evaluation setup | Results | | |
|---|---|---|---|---|---|---|
| | | | | Accuracy | Detection Speed | Model Size and Complexity |
| [54] | YOLOv5 | NEU-DET | CPU: AMD Ryzen 7 6000, RAM: 64 GB DDR4, GPU: NVIDIA GTX2080Ti GPU with 24 GB memory, & OS: Ubuntu 18.04 LTS | mAP: 22.55% ↑ | Time: 61.64% ↓ | - |
| [66] | YOLO5s | Collected | CPU: Intel I9-12900K 24, GPU: NVIDIA GeForce RTX 3090 & OS: Windows 10 | mAP: 0.3↓ | Time: 16.6%↑ | Size: 48.1%↓ Params: 47.9%↓ GFLOPs: 59.4%↓ |
| [68] | YOLOv4 | DCIC | CPU: Intel (R) Xeon (R) Gold 6139, RAM: 32 GB, GPU: NVIDIA GeForce RTX 3090, & OS: Linux | AP: 1.57%↑ | FPS: 33.14%↑ | Params: 89.79%↓ GFLOPs: 87.49%↓ |
| [99] | YOLOv5s | SHWD | NVIDIA Jetson TX2, CPU: Dual-core Denver2, RAM: 8 GB, & GPU: 256-core NVIDIA Maxwell | mAP: 0.1%↓ | Time: 32.72%↓ | Size: 88.89%↓ Params: 91%↓ |
| [100] | YOLOv8s | Collected | CPU: Intel Core i7-8750H @ 2.2 GHz, GPU: NVIDIA GeForce GTX 1050 Ti, & OS: Windows 10 | mAP: 1.3%↓ | Time: 45%↓ FPS: 83.17%↑ | Params: 76.6%↓ GFLOPs: 85.2%↓ |

For general UAV scenarios [72], researcher modified the model's head to specialize in small
targets and involved a lightweight backbone, GhostNe. While a specialized module [58], which was the ALSS module, was specifically designed for enhancing channel information exchange in UAV thermal imagery and included LCA for better spatial integration. To enhance infrared base detection capabilities, the model was redesigned using MobileNet and then quantized [82]. The results are mixed but show potentials. On specialized UAV thermal dataset, the detection accuracy improved [58], while on the general UAV dataset [72], which was VisDrone, the 6.9% mAP improvement came at the cost of a lower frame rate. For infrared ship detection, a significant accuracy drop from quantization was achieved but with better resource-constrained deployability capabilities by about 80%

and 33% reductions in model size and GFLOPs, respectively, as summarized in Table 4. Overall, the hybrid strategy of combining a MobileNet-based lightweight architecture to optimize YOLOv5 with quantization techniques [82] resulted in a significant reduction in parameters by up to 80.07% this came with an accuracy drop achieving 54% mAP due to the complexity of infrared-based image detection. In contrast, the ShuffleNetv2-inspired ALSS-YOLO architecture [58], integrated with LCA attention method and FineSIoU evaluated on the BIRDSAI TIR UAV dataset, obtained 89.1% $mAP_{50}$ and 223.7 FPS, demonstrating better accuracy and real-time performance. Therefore, the most effective trade-off within the UAV and infrared domain can be achieved by combining the structural modification approach of ALSS-YOLO [82] with quantization [58], ensuring both computational efficiency and high detection accuracy for edge-based deployment.

**Table 4.** UAV and Infrared Applications Summary (↑: Increment, ↓: Decrement)

| Ref | Baseline Model | Dataset | Evaluation setup | Results | | |
|---|---|---|---|---|---|---|
| | | | | Accuracy | Detection Speed | Model Size and Complexity |
| [58] | YOLOv8n | ISODRIR UAV | CPU: Intel i9-10900K, GPU: NVIDIA GeForce RTX 3090 GPU with 24GB memory, & OS: Ubuntu 20.04 | $mAP_{50}$: 1.7%↑ | - | - |
| [72] | YOLOv5 | VisDrone-DET 2021 | GPU: ×2 NVIDIA GeForce RTX3060, & OS: Ubuntu 20.04 system | $mAP_{50}$: 6.9%↑ | FPS: 11.67%↓ | Params: 76.7%↓ GFLOPs: 32.3%↓ |
| [82] | YOLOv5 | Collected | - | 6%↓ (QAT) 8%↓ (static) | - | Size: 80.07%↓ GFLOPs: 32.7%↓ |

Safety and security management systems, like detecting E-bikes in elevators, recognizing humans, and identifying if a person is wearing a mask or helmet, require real-time performance on resource-limited devices, such as mobile phones, embedded systems, or low power FPGAs.

Researchers employed lightweight modules, like ShuffleNetv2 with a vision transformer [56, 57] and using both MobileNetv2 and GhostNet concepts [77]. They also examined the influence of quantization-aware training [81], and combination of pruning with GhostNet-based modified models and used knowledge distillation to recover accuracy lost during the aggressive compression [99]. As shown in Table 5, this domain demonstrated the most size and complexity reductions of over 80%-90% with maintaining high accuracy levels that exceeded 90% [57, 99] because of their task well definition. The key metrics in this domain were the frame rate and power consumption, with studies achieving ×50 speed acceleration on FPGAs and power consumption dropping to just 2.4 watts [81]. PG-YOLO [99] demonstrated that the hybrid strategy which combining lightweight Ghost bottleneck architecture, pruning, and knowledge distillation could achieve an optimal trade-off, providing high accuracy of 93.4% $mAP_{50}$ while significantly reducing model size and parameters, and improve inference speed for edge deployment.

**Table 5.** Safety and Security Management Applications Summary (↑: Increment, ↓: Decrement)

| Ref | Baseline Model | Dataset | Evaluation setup | Results | | |
|---|---|---|---|---|---|---|
| | | | | Accuracy | Detection Speed | Model Size and Complexity |
| [56] | Tiny YOLOv7 | Collected | Real mobile device | $mAP_{50}$: 0.2%↑ | Time: 51.73%↓ | Params: 66.39%↓ |
| [57] | YOLOv5s | Collected | CPU: AMD Ryzen 7 5800H with Radeon Graphics @3.20 GHz, RAM: 16 GB, GPU: NVIDIA GeForce RTX 3070 GPU with 8 GB of VRA, & OS: Windows 10 | mAP: 0.9%↓ | FPS: 28.76%↑ | Size= 81.03%↓ Params: 84.33%↓ GFLOPs: 84.81% ↓ |
| [77] | YOLOv5 | CCU-DE | CPU: IntelI CoreIi5-10400F @2.9GHz, RAM: 16GB, | $mAP_{50}$: 0.2%↓ | FPS: 116.42%↑ | Size: 68.61%↓ Params: |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | GPU:NVIDIA GeForce RTX 3060, & OS: Windows 10 | | | 69.63%↓ GFLOPs: 77.44%↓ |
| [81] | Tiny YOLOv3 | WiderFace | TUL Embedded PYNQ-Z2 board, CPU: Dual-Core ARM Cortex-A9, SoC: Xilinx Zynq-7020, OS: Ubuntu, & qauntized models runs on PS@667MHz+PL@100MHz | mAP: 0.8%↓ (easy cat.) | FPS: ×50 times↑ | - |
| [99] | YOLOv5s | SHWD | NVIDIA Jetson TX2, CPU: Dual-core Denver2, RAM: 8 GB, & GPU: 256-core NVIDIA Maxwell | mAP: 0.1%↓ | Time: 32.72%↓ | Size: 88.89%↓ Params: 91% |

Unlike other domains, transportation safety and autonomous vehicle systems require robust detection in highly dynamic and complex environments with numerous object classes and varying scales. Failure can have critical risks, so a balance of accuracy, speed, and reliability is required.

Simple backbone replacements were insufficient in this domain, studies included integrating GhostNet concepts with advanced attention methods, which were MCA [76] and SimAM [73]. They developed HFF and MSFB modules [92] and added specialized detection heads for small and high-resolution objects [76, 92]. Advanced loss functions, MPIoU [76] and WIoU [92], were important for handling dense and overlapping targets. For embedded deployment, filter pruning was combined with structural modification by integrating spatial attention techniques [90]. Studies showed a good complexity reduction of up to 62.04% in GFLOPs [73] and 70.63% in parameters [92] as demonstrated in Table 6. A high pruning rate on an Argoverse-HD dataset, speeded up the FPS by 435.9% at cost of about 5.75% drop in $mAP_{50}$ [90]. The goal is not only eliminating accuracy loss but to find the best balanced performance for real-time system on an edge device like the Jetson Nano [76]. MP-YOLO [92] showed an effective hybrid optimization strategy for autonomous driving in densely scenarios, combining structural enhancement like hierarchical and multi-scale feature fusion, a small-scale detection head, WIOU loss, with LAMP pruning. The model achieved a high $AP_{50}$ of 84.4% and 294.12 FPS, while significantly reducing model size, parameters, and GFLOPs, offering an optimal balance between accuracy, efficiency, and real-time performance on edge devices.

**Table 6.** Transportation Safety and Autonomous Vehicles Applications Summary (↑: Increment, ↓: Decrement)

| Ref | Baseline Model | Dataset | Evaluation setup | Results | | |
|---|---|---|---|---|---|---|
| | | | | Accuracy | Detection Speed | Model Size and Complexity |
| [73] | YOLOv5 | Haulu | GPU: Nvidia GeForce RTX 3090 GPU with 24 GB, & OS: Ubuntu 20.04 | mAP: 0.4%↑ | FPS: 7.41%↑ | Params: 66.51%↓ GFLOPs: 62.04%↓ |
| [76] | YOLOX | BDD100k | NVIDIA Jetson Nano, CPU: ARM Cortex-A57 MPCore, RAM: 2GB GPU: 128-core NVIDIA Maxwell, & OS: Ubuntu 20.04 | mAP: 2.1%↑ | FPS: 39.8%↑ | Params: 49.8%↓ GFLOPs: 51.76%↓ |
| [90] | YOLOv4 | Argoverse-HD | NVIDIA Jetson TX-2, CPU: Dual-core Denver2 (64-bit) + Quad-core ARM A57, & RAM: 8 GB 128-bit LPDDR4, & GPU: NVIDIA Pascal | $mAP_{50}$: ~5.75%↓ | FPS: 435.9%↑ | Params: 70.1%↓ |
| [92] | YOLOV8n | DAIR-V2X | CPU: Intel Xeon(R) Bronze 3204@ 1.90GHz*6, & GPU: GeForce RTX 3080 with 10G | $AP_{50}$: 4.7%↑ | FPS: 11.76%↓ | Size= 63.33%↓ Params: 70.63%↓ GFLOPs: 13.58%↓ |

The agriculture and aquaculture domain involves detecting biological objects like crops, pests, fishes, or flowers, in often uncontrolled environments. The deployment devices are commonly mobile or edge devices, such as handheld scanners, Raspberry Pis, or drones.

Due to their balance of efficiency and robustness, reliable lightweight modules specifically

MobileNetv2 and v3 [63, 64, 65, 67, 97] and ShuffleNetv2 [55, 91] were integrated into the models' designs. These studies often included attention methods like CBAM [67, 97] and CA [65] to better recognize targets from complex uncontrolled backgrounds. Model compression techniques were also used such as advanced knowledge distillation to enhance the performance [97, 98], neck channel pruning [91], and quantization for final deployment on low power hardware [98]. Similar to the industrial domain, the researchers used highly specialized collected datasets. They achieved high accuracy which exceeded 94% mAP [64, 91, 98] with good reductions in model size up to 77.99% and in parameters up to 92.87% [55], making them practical for real-world management on resource-limited devices as shown in Table 7. In this domain, lightweight YOLO models with structural or compression-based optimizations achieved a consistent balance of accuracy and efficiency providing multiple optimization strategies, allowing developers to select approaches based on specific crop types, dataset characteristics, or deployment constraints.

Finally, the goal of the general-purpose domain is not to solve a specific problem but to advance the optimization strategies themself. These models are tested on large diverse public benchmarks, like COCO and PASCAL VOC, to demonstrate the general effectiveness of new compression or architectural techniques. Researchers focused on advanced quantization methods based on unilateral histogram [83] and mixed-precision [85], innovated pruning methods like target capacity [89] and dynamic [93] filter pruning methods. They also investigated combining lightweight modules based on GhostNet with attention techniques [74] and the architecture search with knowledge distillation [96]. As summarized in Table 8, this domain achieved efficiency enhancement by up to 97.5% complexity reduction [96] and 81.1% size drop [74], while the accuracy drops were larger and more significant, reaching up to about 8% mAP decrement with mixed precision [85]. The success of a method is defined by the efficiency gains with minimal trade-offs in challenging testing conditions. Ghost-YOLONet [74] illustrated the most effective trade-off for general-purpose applications, including structure modifications, which were a GhostNet backbone, decoupled fully connected attention, and SimSPPF with depthwise separable convolutions, achieved 81.4% $mAP_{50}$, reduced model size by over 81%, and increased FPS by 9.12%. The techniques used in DD-YOLO [96], combining DARTS search with knowledge distillation, can potentially be integrated with Ghost-YOLONet to achieve further reductions in GFLOPs while improving detection accuracy, creating more efficient YOLO model for real-time, resource-constrained deployments.

**Table 7.** Agriculture and Aquaculture Applications Summary (↑: Increment, ↓: Decrement)

| Ref. | Baseline Model | Dataset | Evaluation setup | Results | | |
|---|---|---|---|---|---|---|
| | | | | Accuracy | Detection Speed | Model Size and Complexity |
| [55] | YOLOv5x | Collected | CPU: two Intel Xeon 5218R, GPU: four NVIDIA GeForce TESLA T4, & OS: Windows 10 | mAP: 2.24%↓ | FPS: 928.57%↑ | Params: 92.87%↓ GFLOPs: 95.51%↓ |
| [63] | YOLOv5s | Collected | CPU: Intel 5218*2 @203Hz, RAM: 64GB, GPU: NVIDIA GeForece GTX 2080Ti with 12GB, & OS: Windows 10 P | AP: 0.8%↓ | Time; 66.15%↓ | Size: 34.53%↓ Params: 35.73%↓ GFLOPs: 40.24%↓ |
| | | | NVIDIA Jetsom Nano (TensorRT), CPU: 4 core ARM A57 @1.43GHz, GPU: NVIDIA128 core Maxwell, & RAM: 4 GB 64-bit LPDDR4 @ 25.6 GB/s | Precision: 5.27%↑ | Time: 62.4%↓ | RT Params: 35.65%↓ |
| [64] | YOLOv4 | Collected | CPU: Intel Core i5-8400 @ 2.80 GHz, GPU: NVIDIA GeForce GTX 2080 Ti, & OS: Ubuntu 16.04 | AP= ~5.275%↑ mAP: 5.28%↑ | FPS: 99.29%↑ | Size: 77.99%↓ |
| [65] | YOLOv7 | Collected | CPU: Inter Core i3 12100, RAM: 16 GB, GPU: GeForce GTX 3060-12G, & OS: Windows 11 | $mAP_{50}$: 0.9%↓ | FPS: 62.71%↑ | Params; 81.32%↓ |
| [67] | YOLOv5s | Combine collected | CPU: 4-core processor, & GPU: Tesla V100 with 32 GB on the Baidu PaddlePaddle platform. | 1.07% ↑ | FPS: 21.32%↑ | GFLOPs: 59.75%↓ |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | & pubic sets | | | | |
| [91] | YOLOv5 | Collected | Raspberry Pi 4b, RAM: 4 GB, Storage: 16 GB (plug-in SD card), & OS[78**:** Official Raspbian OS | mAP: 1.75%↓ | FPS: 218.52%↑ | Size: 73.02%↓ |
| [97] | YOLOv3 | Collected | NVIDIA RTX 3070 | $mAP_{50}$: 6.11%↑ | FPS: 50.49%↑ | Size: 62.32%↓ GFLOPs: 73.52%↓ |
| [98] | Tiny YOLOv4 | Collected | FPGA, OS: Linux, & SoC: Xilinx Zynq UltraScale+ MPSoC EV (Ultra96 SoC) | mAP: 0.47%↓ | - | Size: 73.85%↓ |

**Table 8.** General Purpose Detection Applications Summary (↑: Increment, ↓: Decrement)

| **Ref #** | **Baseline Model** | **Dataset** | **Evaluation setup** | **Results** | | |
|---|---|---|---|---|---|---|
| | | | | **Accuracy** | **Detection Speed** | **Model Size and Complexity** |
| [74] | YOLOv4 | PASCAL VOC 2007 & 2012 | - | $mAP_{50}$: 4.2%↓ | FPS: 9.12%↑ | Size: 81.1%↓ |
| [83] | YOLOv5s | COCOval2017 | CPU: Intel i7-12700H(x86) (OPenVINO) | AP: 0.1%↓ | Time: 31.02%↑ | Size=75%↓ |
| [85] | YOLOv5 | COCO & PASCAL VOC | GPU: NVIDIA GeForce RTX 3090 | ~8%↓ | - | Size= 54%↓ GFLOPs: 78%↓ |
| [89] | YOLOv5 | PASCAL VOC | NVIDIA Jetson Xavier NX | $mAP_{50}$: 2.9%↓ | Time: 34.47%↓ | Params: 29.98%↓ GFLOPs: 46.16%↓ |
| [93] | YOLOv3 | COCO | NVIDIA 3090 GPUs | mAP: 1.2%↓ | - | Params: 46.63%↓ GFLOPs; 58.85↓ |
| [96] | YOLOv4 | COCO2017 test-dev | GPU: TITAN XP 12G graphics card, & OS: Linux | AP: 2.4%↑ | - | Params: 61.4% GFLOPs: 97.5% |

A quantitative synthesis of the collected studies showed clear trends in accuracy, speed, and model complexity. Structural modification approach achieved about 35–90% reductions in parameters and FLOPs, with average mAP decreases within 1–4%, while the compression approach offered additional computational gains, with pruning reducing parameters by up to 70% and quantization improving inference time, especially on CPUs and embedded GPUs which support low precision operations. Knowledge distillation methods further enhanced accuracy retention, ensuring that smaller student models maintained performance levels close to the original networks. Detection speeds improved by 30–400%, particularly when models were deployed on platforms such as Jetson Nano, TX2, or FPGA, confirming the feasibility of optimized YOLO variants for real-time low-power applications.

Although this review provides a comprehensive analysis of lightweight YOLO optimization strategies for edge computing, several limitations should be highlighted. First, the diversity of datasets and hardware platforms used in the reviewed studies makes direct comparison of performance metrics difficult. Second, some studies lacked detailed implementation settings or comprehensive metrics, limiting the consistent benchmarking. Third, this review focuses on YOLO-based models and does not include other computer vision lightweight architectures that may show different optimization behaviour at constrained conditions. Finally, while efforts were made to include recent research, the rapid evolution of YOLO variants and optimization techniques means that new advancements may be beyond the scope of this review. These limitations highlight the need for updated standardized edge-specific benchmarks, guided evaluation frameworks, and continued investigation into hybrid optimization strategies for future research.

## 6. CONCLUSION

This paper presents a comprehensive review of optimization strategies that are used Sto enhance YOLO models for efficient operation and suitable

deployment in resource-constrained environments. It systematically classifies and analyzes existing YOLO optimization approaches including structure modification, model compression, and hybrid strategies. Furthermore, it provides a comparative framework that shows performance trade-offs across different application domains, datasets, and hardware platforms. Finally, it identifies the main principles that improve computational efficiency while preserving high detection accuracy, providing guidance for developing lightweight object detection models suitable for real-time edge deployment.

The review highlights that structural modifications ensure better model stability and generalization, while compression-based methods such as pruning and quantization provide further compactness and inference speed. Hybrid optimization is the most balanced solution, achieving high detection accuracy, significant size and GFLOP reductions, and good inference speed. Across existing studies, the average reductions are 35%-90% in model size, 30%-93% in parameter count, and 15%–98% in GFLOP. Optimization strategies often lead to increased inference speed and maintained or improved accuracy, however, in some cases accuracy degradation and inference slowdown occur as a trade-off for higher efficiency and compactness. These results demonstrate the high potential of optimized YOLO architectures to enable real-time detection in embedded, IoT, and edge-AI devices with limited computational resources.

In practical terms, the optimized YOLO models offer advantages for industrial, agricultural, smart surveillance, and autonomous systems. These models provide faster decision making and longer device lifetimes in edge deployments due to their reductions in memory usage and energy consumption while maintaining accuracy. The reviewed strategies support scalable and energy-efficient computer vision systems that can operate reliably on various environmental and hardware constraints.

Future research should focus on hardware-aware and adaptive optimization, where platform specific characteristics are considered during training and deployment to ensure efficient inference across edge devices. Additionally, exploring federated edge learning could enhance distributed model optimization while preserving data privacy. Another promising direction is to develop dynamic and hybrid quantization methods that enable real-time adaptation of model precision according to task complexity and available computational resources. Finally, integrating neural architecture search with energy-driven training strategies can automate the finding of architectures that balance accuracy, latency, and power consumption, ensuring scalable and efficient deployment on IoT, UAV, and mobile edge platforms.

## NOMENCLATURE

| | |
|---|---|
| 2D | Two-Dimensional |
| AI | Artificial Intelligence |
| AP | Average Precision |
| ARM | Advanced Reduced instruction set computer Machine |
| C3 | Cross Stage Pirtial with 3 convolutions |
| CA | Coordinate Attention |
| CBAM | Convolutional Block Attention Module |
| CCTV | Closed-Circuit Television |
| CIoU | Complete Intersection over Union |
| CNN | Convolutional Neural Network |
| Conv | Convolution |
| Conv-GN | Convolution and Group Normalization |
| CSPDarknet53 | Cross Stage Pirtial Darknet53 |
| DARTS | Differentiable Architecture Search |
| DGSM | Dynamic Group Convolution Shuffle Module |
| DGST | Dynamic Grouped Convolution Shuffle Transformer |
| EIoU | Efficient Intersection over Union |
| FLOPS | Floating Point Operations |
| FP16 | Floating Point 16-bit |
| FPGA | Field-Programmable Gate Array |
| FPN | Feature Pyramid Network |
| FPS | Frame Per Second |
| GFLOPs | Giga Floating Point Operations |
| GIoU | Generalized Intersection over Union |
| GPU | Graphics Processing unit |
| HFF | Hierarchical Feature Fusion |
| HRank | High-Rank feature map pruning |
| H-SiLU | Hard Sigmoid-Weighted Linear Unit |
| IIoT | Industrial Internet of Things |
| img/s | Image per second |
| IoT | Internet of Things |
| LAMP | Layer Adaptive sparsity for Magnitude-based Pruning |
| LCA | Lightweight Coordinate Attention |
| LCP | Layer-compensated Pruning |
| LFPN | Lightweight Feature Pyramid Network |
| mAP | mean Average Precision |
| $mAP_{50}$ | mean Average Precision at intersection over union threshold 0.5 |
| MCA | Multidimensional Collaborative Attention |
| MPDIoU | Minimum Point Distance Intersection over Union |
| MPFD | Multi-teacher Pre-activation Feature Distillation |
| MQA | Mobile Multi-Query Attention |
| ms | Millisecond |
| MSE | Quantization Mean Square Error |
| MSFB | Multi-Scale Feature Fusion Block |

| | |
|---|---|
| NAS | Neural Architecture Search |
| PAN | Path Aggregation Network |
| PL | Programmable Logic |
| PNQ | Progressive Network Quantization |
| PS | Programmable System |
| PTQ | Post-Training Quantization |
| QAT | Quantization Aware Training |
| R-CNN | Region-Based Convolutional Neural Network |
| R-Pruning | Recovery Pruning |
| SE | Squeeze-and-Excitation |
| SENET | Squeeze-and-Excitation Natework |
| SiLU | Sigmoid-Weighted Linear Unit |
| SimAM | Simple Attention Module |
| SimSPPF | Simple Spatial Pyramid Pooling Fast |
| SIoU | Squared Intersection over Union |
| SPP | Spatial Pyramid Pooling |
| SPPF | Spatial Pyramid Pooling Fast |
| TCFP | Target Capacity Filter Pruning |
| TensorRT | Tensor RunTime |
| TIR | Thermal Infrared |
| TPU | Tensor Processing Unit |
| UAV | Unmanned Aerial Vehicles |
| UH | Unilateral Histogram-based |
| UIB | Universal Inverted Bottleneck |
| ViT | Vision Transformer |
| WIoU | Weighted Intersection over Union |
| YOLO | You Only Look Once |
| $\gamma$ | Saliency Score |

## References


[1] H. Feng, G. Mu, S. Zhong, P. Zhang, and T. Yuan, "Benchmark Analysis of YOLO Performance on Edge Intelligence Devices," *Cryptography*, vol. 6, no. 2, p. 16, Apr. 2022, doi: 10.3390/cryptography6020016.

[2] S. A. Mostafa et al., "A YOLO-based deep learning model for Real-Time face mask detection via drone surveillance in public spaces," *Information Sciences*, vol. 676, p. 120865, Aug. 2024, doi: 10.1016/j.ins.2024.120865.

[3] S. Kang, Z. Hu, L. Liu, K. Zhang, and Z. Cao, "Object Detection YOLO Algorithms and Their Industrial Applications: Overview and Comparative Analysis," *Electronics*, vol. 14, no. 6, p. 1104, Mar. 2025, doi: 10.3390/electronics14061104.

[4] G. Xue, S. Li, P. Hou, S. Gao, and R. Tan, "Research on lightweight Yolo coal gangue detection algorithm based on resnet18 backbone feature network," *Internet of Things*, vol. 22, p. 100762, Jul. 2023, doi: 10.1016/j.iot.2023.100762.

[5] Z. Lyu, T. Yu, F. Pan, Y. Zhang, J. Luo, D. Zhang, Y. Chen, B. Zhang, and G. Li, "A survey of model compression strategies for object detection," *Multimedia Tools and Applications*, vol. 83, no. 16, pp. 48165–48236, 2023, doi: 10.1007/s11042-023-17192-x.

[6] M. H. Mir, J. A. Kovilpillai J, S. S. Mohamed, Pragya, T. A. Mir, and B. Paul, "Deep learning based Crop Monitoring for effective Agricultural-IoT Management," *Procedia Computer Science*, vol. 258, pp. 332–341, 2025, doi: 10.1016/j.procs.2025.04.270.

[7] P. Mittal, "A comprehensive survey of deep learning-based lightweight object detection models for edge devices," *Artificial Intelligence Review*, vol. 57, no. 9, p. 242, Aug. 2024, doi: 10.1007/s10462-024-10877-1.

[8] M. A. Burhanuddin, "Efficient Hardware Acceleration Techniques for Deep Learning on Edge Devices: A Comprehensive Performance Analysis," *KHWARIZMIA*, vol. 2023, pp. 103–112, Aug. 2023, doi: 10.70470/KHWARIZMIA/2023/010.

[9] H. Xue, B. Huang, M. Qin, H. Zhou, and H. Yang, "Edge Computing for Internet of Things: A Survey," *in 2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics), IEEE*, Nov. 2020, pp. 755–760. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00130.

[10] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for IoT: A survey on current research activities & future directions," *Computer Communications*, vol. 180, pp. 210–231, Dec. 2021, doi: 10.1016/j.comcom.2021.09.003.

[11] X. Kong, Y. Wu, H. Wang, and F. Xia, "Edge Computing for Internet of Everything: A Survey," *IEEE Internet Things Journal*, vol. 9, no. 23, pp. 23472–23485, Dec. 2022, doi: 10.1109/JIOT.2022.3200431.

[12] E. Fazeldehkordi and T.M. Grønli, "A Survey of Security Architectures for Edge Computing-Based IoT," *IoT*, vol. 3, no. 3, pp. 332–365, Jun. 2022, doi: 10.3390/iot3030019.

[13] S. Iftikhar et al., "AI-based fog and edge computing: A systematic review, taxonomy and future directions," *Internet of Things*, vol. 21, p. 100674, Apr. 2023, doi: 10.1016/j.iot.2022.100674.

[14] 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.

[15] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.

[16] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," *in Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, UT, USA, 18–22 June 2018; Volume 1804, pp. 1–6. in Computer vision and pattern recognition, 2018, pp. 1–6.

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

[18] G. Jocher, "Ultralytics YOLOv5," 2020, doi: 10.5281/zenodo.3908559.

[19] C. Li et al., "YOLOv6: A single-stage object detection framework for industrial applications," *arXiv preprint*, arXiv:2209.02976, 2022.

[20] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, 2023, pp. 7464-7475, doi: 10.1109/CVPR52729.2023.00721.

[21] G. Jocher, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

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

[23] A. Wang et al., "Yolov10: Real-time end-to-end object detection," 2024, *arXiv preprint*, arXiv:2405.14458, 2024.

[24] G. Jocher and contributors, "YOLOv11 by Ultralytics," 2023. doi: 10.5281/zenodo.1234567.

[25] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-Centric Real-Time Object Detectors," *arXiv preprint*, arXiv:2502.12524, 2025.

[26] I. S. Gillani et al., "Yolov5, Yolo-x, Yolo-r, Yolov7 Performance Comparison: A Survey," *in Artificial Intelligence and Fuzzy Logic System*, Sep. 2022, pp. 17–28. doi:10.5121/csit.2022.121602.

[27] M. Hussain, "YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO," *IEEE Access*, vol. 12, pp. 42816–42833, 2024, doi: 10.1109/ACCESS.2024.3378568.

[28] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo Algorithm Developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022, doi: 10.1016/j.procs.2022.01.135.

[29] M. G. Ragab et al., "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)," *IEEE Access*, vol. 12, pp. 57815–57836, 2024, doi: 10.1109/ACCESS.2024.3386826.

[30] 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.

[31] M. L. Ali and Z. Zhang, "The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection," *Computers*, vol. 13, no. 12, p. 336, Dec. 2024, doi: 10.3390/computers13120336.

[32] U. Sirisha, S. P. Praveen, P. N. Srinivasu, P. Barsocchi, and A. K. Bhoi, "Statistical Analysis of Design Aspects of Various YOLO-Based Deep Learning Models for Object Detection," *International Journal of Computational Intelligence Systems*, vol. 16, no. 1, p. 126, Aug. 2023, doi: 10.1007/s44196-023-00302-w.

[33] M. Hussain, and R. Khanam, "In-Depth Review of YOLOv1 to YOLOv10 Variants for Enhanced Photovoltaic Defect Detection," *Solar*, vol *4. no.* (3), pp. 351–386, 2024, doi: 10.3390/solar4030016.

[34] Z. Hua et al., "A Benchmark Review of YOLO Algorithm Developments for Object Detection," *IEEE Access*, vol. 13, pp. 123515–123545, 2025, doi: 10.1109/ACCESS.2025.3586673.

[35] J. Wei, A. As'arry, K. Anas Md Rezali, M. Zuhri Mohamed Yusoff, H. Ma, and K. Zhang, "A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection," *IEEE Access*, vol. 13, pp. 93688–93711, 2025, doi: 10.1109/ACCESS.2025.3573376.

[36] S. Liang et al., "Edge YOLO: Real-Time Intelligent Object Detection System Based on Edge-Cloud Cooperation in Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25345–25360, Dec. 2022, doi: 10.1109/TITS.2022.3158253.

[37] F. A. A.- Ibraheemi et al., "A Cognitive Energy-Driven Routing Strategy for Ultra-Efficient Data Transfer in Wireless Sensor Networks," *Applied Data Science and Analysis*, vol. 2025, pp. 131–143, Apr. 2025, doi: 10.58496/ADSA/2025/011.

[38] J. Li and J. Ye, "Edge-YOLO: Lightweight Infrared Object Detection Method Deployed on Edge Devices," *Applied Sciences*, vol. 13, no. 7, p. 4402, Mar. 2023, doi: 10.3390/app13074402.

[39] 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.

[40] A.E. abdelkareem, "Performance Analysis of Deep Learning based Signal Constellation Identification Algorithms for Underwater Acoustic Communications", *DJES*, vol. 17, no. 3, pp. 1–14, Sep. 2024, doi: 10.24237/djes.2024.17301.

[41] X. Lv, T. Chen, C. Song, C. Yang, and T. Ping, "Application of YOLO Algorithm for Intelligent Transportation Systems: A Survey and New Perspectives," *International Journal of Distributed Sensor Networks*, vol. 2025, no. 1, Jan. 2025, doi: 10.1155/dsn/2859040.

[42] S. Yassine and A. Stanulov, "A COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS FOR THE PURPOSE OF PREDICTING NORWEGIAN AIR PASSENGER TRAFFIC," *International Journal of Mathematics, Statistics, and Computer Science*, vol. 2, pp. 28–43, Jul. 2023, doi: 10.59543/ijmscs.v2i.7851.

[43] M. Flores-Calero et al., "Traffic Sign Detection and Recognition Using YOLO Object Detection Algorithm: A Systematic Review," *Mathematics*, vol. 12, no. 2, p. 297, Jan. 2024, doi: 10.3390/math12020297.

[44] A. B. Raharjo, F. Dumont, and E. Thibaudeau, "Comparative Analysis of YOLO-Based Object," *Advances in Computing and Data Sciences: 8th International Conference (ICACDS 2024)*, Vélizy, France, May 9–10, 2024, Revised Selected Papers, vol. 2194, p. 93, Springer Nature, Oct. 2024.

[45] Z. Yang, X. Lan, and H. Wang, "Comparative Analysis of YOLO Series Algorithms for UAV-Based Highway Distress Inspection: Performance and Application Insights," *Sensors*, vol. 25, no. 5, p. 1475, Feb. 2025, doi: 10.3390/s25051475.

[46] 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

[47] S. Liu, J. Zha, J. Sun, Z. Li and G. Wang, "EdgeYOLO: An Edge-Real-Time Object Detector," *2023 42nd Chinese Control Conference (CCC)*, Tianjin, China, 2023, pp. 7507-7512, doi: 10.23919/CCC58697.2023.10239786.

[48] 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.

[49] C. Hou, Z. Li, X. Shen, and G. Li, "Real-time defect detection method based on YOLO-GSS at the edge end of a transmission line," *IET Image Processing*, vol. 18, no. 5, pp. 1315–1327, Apr. 2024, doi: 10.1049/ipr2.13028.

[50] X. Hu and H. Wen, "Research on Model Compression for Embedded Platform through Quantization and Pruning," *Journal of Physics: Conference Series*, vol. 2078, no. 1, p. 012047, Nov. 2021, doi: 10.1088/1742-6596/2078/1/012047.

[51] A. Lopes, F. Pereira dos Santos, D. de Oliveira, M. Schiezaro, and H. Pedrini, "Computer Vision Model Compression Techniques for Embedded Systems: A Survey," *Computers & Graphics*, vol. 123, p. 104015, Oct. 2024, doi: 10.1016/j.cag.2024.104015.

[52] P. V. Dantas, W. Sabino da Silva, L. C. Cordeiro, and C. B. Carvalho, "A comprehensive review of model compression techniques in machine learning," *Applied Intelligence*, vol. 54, no. 22, pp. 11804–11844, 2024, doi: 10.1007/s10489-024-05747-w.

[53] N. Ma, X. Zhang, H. T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," *in Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., : Springer, Cham,

Switzerland, 2018, vol. 11218, Lecture Notes in Computer Science, doi:10.1007/978-3-030-01264-9_8.

[54] S. Muhammad Yasir and H. Ahn, "Faster Metallic Surface Defect Detection Using Deep Learning with Channel Shuffling," *Computers, Materials & Continua*, vol. 75, no. 1, pp. 1847–1861, 2023, doi: 10.32604/cmc.2023.035698.

[55] F. W. Feng Wang, J. Z. Feng Wang, J. Z. Jing Zheng, X. Z. Jiawei Zeng, and Z. L. Xincong Zhong, "S2F-YOLO: An Optimized Object Detection Technique for Improving Fish Classification," *網際網路技術學刊*, vol. 24, no. 6, pp. 1211–1220, Nov. 2023,doi:10.53106/160792642023112406004.

[56] W. Gong, "Lightweight Object Detection: A Study Based on YOLOv7 Integrated with ShuffleNetv2 and Vision Transformer," *arXiv preprint*, arXiv:2403.0173, Mar. 2024.

[57] J. Su, M. Yang, and X. Tang, "Integration of ShuffleNet V2 and YOLOv5s Networks for a Lightweight Object Detection Model of Electric Bikes within Elevators," *Electronics* (Basel), vol. 13, no. 2, p. 394, Jan. 2024, doi: 10.3390/electronics13020394.

[58] 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.

[59] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint*, arXiv:1704.04861, Apr. 2017.

[60] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. -C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.

[61] A. Howard et al., "Searching for MobileNetV3," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), 2019, pp. 1314-1324, doi: 10.1109/ICCV.2019.00140.

[62] D. Qin et al., "MobileNetV4: Universal models for the mobile ecosystem," *in Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., Springer, Cham, Switzerland, 2025, vol. 15098, Lecture Notes in Computer Science, doi:10.1007/978-3-031-73661-2_5.

[63] K. Yu, G. Tang, W. Chen, S. Hu, Y. Li, and H. Gong, "MobileNet-YOLO v5s: An Improved Lightweight Method for Real-Time Detection of Sugarcane Stem Nodes in Complex Natural Environments," *IEEE Access*, vol. 11, pp. 104070–104083, 2023, doi: 10.1109/ACCESS.2023.3317951.

[64] Y. Li, X. Ma, and J. Wang, "Pineapple maturity analysis in natural environment based on MobileNet V3-YOLOv4," *Smart Agriculture*, vol. 5, no. 2, pp. 35–44, 2023, doi: 10.12133/j.smartag.SA202211007.

[65] L. Jia et al., "MobileNet-CA-YOLO: An Improved YOLOv7 Based on the MobileNetV3 and Attention Mechanism for Rice Pests and Diseases Detection," *Agriculture*, vol. 13, no. 7, p. 1285, Jun. 2023, doi: 10.3390/agriculture13071285.

[66] X. Lang, Z. Ren, D. Wan, Y. Zhang, and S. Shu, "MR-YOLO: An Improved YOLOv5 Network for Detecting Magnetic Ring Surface Defects," *Sensors*, vol. 22, no. 24, p. 9897, Dec. 2022, doi: 10.3390/s22249897.

[67] Y. Qiaomei, C. Tingting, Y. Yongbang, and L. Hua, "Research on the application of lightweight YOLO model in detection of small crop diseases and pests," *Journal of Chinese Agricultural Mechanization*, vol. 45, no. 9, p. 265, 2024.

[68] H. Liu, W. Cheng, C. Li, Y. Xu, and S. Fan, "Lightweight Detection Model RM-LFPN-YOLO for Rebar Counting," *IEEE Access*, vol. 12, pp. 3936–3947, 2024, doi: 10.1109/ACCESS.2024.3349978.

[69] U. K. Altaie, A. E. Abdelkareem, and A. Alhasanat, "YOLO optimization for edge AI: A lightweight approach for deep sky object detection," *Iraqi Journal of Information and Communication Technology*, vol. 8, no. 2, pp. 13–28, 2025.

[70] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: More Features From Cheap Operations," *in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2020, pp. 1577–1586. doi: 10.1109/CVPR42600.2020.00165.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[72] J. Cao, W. Bao, H. Shang, M. Yuan, and Q. Cheng, "GCL-YOLO: A GhostConv-Based Lightweight YOLO Network for UAV Small Object Detection," *Remote Sensing* (Basel), vol. 15, no. 20, p. 4932, Oct. 2023, doi: 10.3390/rs15204932.

[73] 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.

[74] H. Yan, Y. He, C. Cai, and Y. Zhang, "A lightweight target detection network: ghost-YOLONet," *in International Conference on Remote Sensing, Mapping, and Image Processing (RSMIP 2024)*, vol. 13167, pp, 868-872, SPIE, 2024.

[75] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint*, arXiv:2107.08430, Jul. 2021.

[76] W. Wang and W. Yu, "Enhancing Real-Time Road Object Detection: The RD-YOLO Algorithm With Higher Precision and Efficiency," *IEEE Access*, vol. 12, pp. 190876-190888, 2024, doi: 10.1109/ACCESS.2024.3518208.

[77] Y. Lei, D. Pan, Z. Feng, and J. Qian, "Lightweight YOLOv5s Human Ear Recognition Based on MobileNetV3 and Ghostnet," *Applied Sciences*, vol. 13, no. 11, p. 6667, May 2023, doi: 10.3390/app13116667.

[78] B. Jacob et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," *in Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704--2713.

[79] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A White Paper on Neural Network Quantization,", *arXiv preprint*, arXiv:2106.08295, Jun. 2021.

[80] A. K. Al-Zihairy and A. E. Abdelkareem, "Optimizing YOLOv8-cls: A Step Towards Smarter Edge Environments," *2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI), Sana'a, Yemen*, 2024, pp. 1-6, doi: 10.1109/ICETI63946.2024.10777236.

[81] B. Gunay, S. B. Okcu, and H. S. Bilge, "LPYOLO: Low Precision YOLO for Face Detection on FPGA," *in The 8th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'22), Prague*, Czech Republic, Jul. 2022. doi: 10.11159/mvml22.108.

[82] P. Cui, J. Zhang, B. Han, and Y. Wu, "Performance evaluation and model quantization of object detection algorithm for infrared image," *in Seventh Asia Pacific*

*Conference on Optics Manufacture and 2021 International Forum of Young Scientists on Advanced Optical Manufacturing (APCOM and YSAOM 2021)*, J. Tan, X. Luo, M. Huang, L. Kong, and D. Zhang, Eds., SPIE, Feb. 2022, p. 118. doi: 10.1117/12.2614541.

[83] M. Wang et al., "Q-YOLO: Efficient Inference for Real-Time Object Detection," *Pattern Recognition (ACPR 2023)*, H. Lu, M. Blumenstein, SB. Cho, CL. Liu, L. Yagi, and T. Kamiya, Eds., Springer, Cham, 2023, vol. 14408, doi: 10.1007/978-3-031-47665-5_25

[84] X. Yue, H. Li, M. Shimizu, S. Kawamura, and L. Meng, "YOLO-GD: A Deep Learning-Based Object Detection Algorithm for Empty-Dish Recycling Robots," *Machines*, vol. 10, no. 5, p. 294, Apr. 2022, doi: 10.3390/machines10050294.

[85] 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.

[86] H. Cheng, M. Zhang, and J. Q. Shi, "A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, Aug. 2024, doi: 10.1109/TPAMI.2024.3447085.

[87] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint*, arXiv:1710.01878, Oct. 2017.

[88] L. Malihi and G. Heidemann, "Matching the Ideal Pruning Method with Knowledge Distillation for Optimal Compression," *Applied System Innovation*, vol. 7, no. 4, p. 56, Jun. 2024, doi: 10.3390/asi7040056.

[89] J. Jeon, J. Kim, J.-K. Kang, S. Moon, and Y. Kim, "Target Capacity Filter Pruning Method for Optimized Inference Time Based on YOLOv5 in Embedded Systems," *IEEE Access*, vol. 10, pp. 70840–70849, 2022, doi: 10.1109/ACCESS.2022.3188323.

[90] H. Ahn et al., "SAFP-YOLO: Enhanced Object Detection Speed Using Spatial Attention-Based Filter Pruning," *Applied Sciences*, vol. 13, no. 20, p. 11237, Oct. 2023, doi: 10.3390/app132011237.

[91] S. Zhang et al., "Edge Device Detection of Tea Leaves with One Bud and Two Leaves Based on ShuffleNetv2-YOLOv5-Lite-E," *Agronomy*, vol. 13, no. 2, p. 577, Feb. 2023, doi: 10.3390/agronomy13020577.

[92] W. Zhou, J. Wang, J. Wang, M. Xi, Y. Song, and Z. Liu, "Mp-Yolo: Multidimensional Feature Fusion Based Layer Adaptive Pruning Yolo for Dense Vehicle Object Detection Algorithm," 2024. doi: 10.2139/ssrn.4952235.

[93] 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.

[94] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *arXiv preprint*, arXiv:1503.02531, Mar. 2015,

[95] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021, doi: 10.1007/s11263-021-01453-z.

[96] Z. Xing, X. Chen, and F. Pang, "DD-YOLO: An object detection method combining knowledge distillation and Differentiable Architecture Search," *IET Computer Vision*, vol. 16, no. 5, pp. 418–430, Aug. 2022, doi: 10.1049/cvi2.12097.

[97] Y. Liu et al., "An Improved Tuna-YOLO Model Based on YOLO v3 for Real-Time Tuna Detection Considering Lightweight Deployment," *Journal of Marine Science and Engineering*, vol. 11, no. 3, p. 542, Mar. 2023, doi: 10.3390/jmse11030542.

[98] S. Lyu, Y. Zhao, X. Liu, Z. Li, C. Wang, and J. Shen, "Detection of Male and Female Litchi Flowers Using YOLO-HPFD Multi-Teacher Feature Distillation and FPGA-Embedded Platform," *Agronomy*, vol. 13, no. 4, p. 987, Mar. 2023, doi: 10.3390/agronomy13040987.

[99] 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.

[100] Q. Huang, C. Fan, Y. Sun, J. Huang, and W. Jiang, "A Lightweight YOLOv8s Algorithm for Ceiling Fan Blade Defect Detection With Optimized Pruning and Knowledge Distillation," *IEEE Access*, vol. 13, pp. 97392–97408, 2025, doi: 10.1109/ACCESS.2025.3575952.