iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70025

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

SmartNavNet: A Lightweight Object Detection Model for Warehouse Automation

Manodharshan K[1], Valliappan V[2], Tamil K[3], Javith Ali S[4], Dr. A. Geetha[5]
Annamalai University, Bachelor of Engineering and Technology, Annamalai Nagar, Chidhambaram, Cuddalore.

Abstract: ***Warehouse automation demands fast, accurate object detection capable of operating on resource-constrained edge devices. In this work, we present SmartNavNet, a purpose-built detection network that synergistically integrates Ghost Convolution modules, a CSP-PANet feature aggregation neck, Squeeze-and-Excitation channel attention, and INT8 post-training quantization. On the LOCO warehouse dataset—comprising 10,000 images of five classes under variable lighting, occlusion, and scale—SmartNavNet achieves 62.2% mAP@0.5, 70.2% precision, and 55.7% recall, with a 10 ms average inference latency on a Snapdragon 855 and a 3 MB quantized footprint. Compared to YOLOv4-Tiny and YOLOv5n baselines, our model offers up to 4.2% mAP improvement while halving model size and reducing latency by 20%, making it uniquely suitable for real-time warehouse applications.***

Keywords: warehouse automation; object detection; lightweight neural networks; ghost convolution; CSP-PANet; squeeze-and-excitation; INT8 quantization.

I. INTRODUCTION

The rapid growth of e-commerce and just-in-time logistics has placed unprecedented demands on warehouse efficiency and accuracy. Modern warehouses leverage autonomous guided vehicles, robotic pick-and-place systems, and real-time inventory tracking to maintain operational throughput. At the core of these systems lies object detection: identifying boxes, pallets, forklifts, and other key elements in dynamic and cluttered environments. While high-capacity convolutional networks achieve state-of-the-art accuracy [1], they require extensive compute and memory, impeding deployment on low-power edge platforms.Lightweight detectors like YOLOv4-Tiny [2] and YOLOv5n [3] mitigate this gap by reducing network depth and width but often suffer performance degradation in complex warehousing scenes characterized by heavy occlusion, uniform textures, and varying object scales. Our goal is to design a compact yet accurate model specifically optimized for warehouse imagery. We propose SmartNavNet, in which novel Ghost Convolution modules generate efficient feature maps, a CSP-PANet neck fuses multi-scale information, SE blocks provide channel-wise attention, and INT8 quantization further compresses the network for real-time edge inference.

II. BASIC CONCEPT

The design of SmartNavNet is motivated by the need to maximize detection accuracy while minimizing both computational cost and model size, enabling real-time inference on resource-constrained hardware. Accordingly, the architecture embodies four interdependent innovations:

A. Efficient Feature Generation via Ghost Convolutions

Traditional 3×3 convolutional layers often produce redundant feature maps, leading to unnecessary computation. Ghost Convolution modules address this by first computing a small number of intrinsic feature maps through standard convolutions, and then generating additional "ghost" maps via inexpensive linear transformations (e.g., depth wise convolution or linear projection). This two-stage process reduces FLOPs by approximately 30% and cuts down on parameter count, while preserving the richness and diversity of learned features. Ghost modules are therefore ideal for edge scenarios where every cycle of computation matters [4].

B. Cross-Stage Partial Multi-Scale Fusion with CSP-PANet

In complex warehouse scenes, objects appear at a variety of scales—from small barcode labels on boxes to large pallets stacked on shelves. To handle this, SmartNavNet employs a CSP-PANet neck that combines the strengths of Cross-Stage Partial (CSP) connections with Path Aggregation Network (PANet) fusion. CSP splits feature channels into two pathways, one undergoing further transformation and the other bypassing, which reduces computation and mitigates feature duplication.

The PANet structure then merges high-resolution spatial features from early layers with semantically rich representations from deeper layers through top-down and bottom-up pathways. This design ensures that both fine details and abstract concepts are readily available to the detection head [5].

C. Adaptive Channel Attention through Squeeze-and-Excitation

Not all feature channels contribute equally to accurate object localization and classification. Squeeze-and-Excitation (SE) blocks dynamically reweight channels by first "squeezing" global spatial information into a channel descriptor via global average pooling and then "exciting" each channel through a lightweight gating mechanism (two fully-connected layers with a reduction ratio). The resulting channel-wise weights are applied multiplicatively, amplifying informative features and suppressing less relevant ones. SE blocks introduce negligible overhead yet yield consistent accuracy gains, particularly under challenging lighting and occlusion conditions common in warehouse environments [6].

D. Deployment-Ready Compression with INT8 Quantization

Even after architectural optimizations, floating-point models can be too large or slow for embedded inference. SmartNavNet leverages post-training INT8 quantization using TensorFlow Lite, converting both weights and activations to 8-bit integers. Through careful calibration (using a representative subset of 1 000 images), this process reduces the model footprint from ~10 MB to 3 MB and speeds up inference by roughly 40% on a Snapdragon 855, with less than 1% drop in mAP. INT8 quantization thus makes SmartNavNet practical for continuous, battery-powered warehouse applications [7].

III. LITERATURE REVIEW

A. Surveys and Benchmarks

Mittal [8] provides a comprehensive survey of lightweight detection architectures, categorizing trade-offs in size versus accuracy. Nafea et al. [9] review methods for mobile augmented reality, emphasizing pruning and quantization for on-device inference.

B. Architecture Enhancements

Gong [10] fuses ShuffleNetV2 and Vision Transformer elements into YOLOv7, demonstrating that hybrid features enhance detection diversity. Zhang et al. [11] adapt YOLOv5 for unmanned surface vehicles, highlighting the benefits of domain-specific anchor and head configurations. Chen et al. [12] propose TinyDet, optimizing neck and head modules for small-object scenarios.

C. Attention and Multiscale Learning

Sunkara and Luo's YOGA [13] model integrates multiscale attention to tackle occlusions, while Ji et al.'s YOLO-TLA [14] applies lightweight transformers for tiny-object detection, achieving notable gains in mAP on crowded scenes.

D. Small-Object & Anchor Optimization

Nguyen et al. [15] systematically evaluate feature pyramid networks for small-object detection, and Zhong et al. [16] introduce dynamic anchor box optimization through online clustering for improved localization.

E. Edge-Device Deployment

Moosmann et al. [17] deliver TinyissimoYOLO, a sub-1 MB fully-quantized model for ultra-low-power IoT, and Humes et al. [18] develop Squeezed Edge YOLO, attaining sub-10 ms inference on embedded GPUs. Liu et al. [19] present EdgeYOLO, fusing quantization-aware training and hardware acceleration.

IV. PROPOSED ARCHITECTURE

To provide a clear and comprehensive view of SmartNavNet's inner workings, we detail each component in terms of layer configurations, channel dimensions, and operational flow. Fig. 1 illustrates the overall architecture.

A. Backbone: Ghost-MobileNetV3

The backbone extends MobileNetV3-Small, replacing every standard 3×3 convolution with a Ghost Convolution module. The network comprises five stages:

1) *Stage 1 (Input):* 640×640×3 image; initial 3×3 conv (stride 2), output 320×320×16.

2) *Stage 2:* Two Ghost modules (kernel 3×3, expansion 1.5×), output 160×160×24.
3) *Stage 3:*Four Ghost modules (kernel 5×5, expansion 2×), output 80×80×40.
4) *Stage 4:* Five Ghost modules (kernel 3×3, expansion 2.5×), output 40×40×80.
5) *Stage 5:* Three Ghost modules (kernel 5×5, expansion 3×), output 20×20×112.

Each Ghost module uses a 1×1 pointwise conv to compute intrinsic feature maps followed by depth wise conv to generate ghost maps, concatenating to the stage output.

B. *Neck: CSP-PANet Feature Aggregation*

The neck fuses three feature maps from the backbone: 20×20×112, 40×40×80, and 80×80×40.

1) *CSP Blocks:* Each feature map is split 50:50 into a processing path and a skip path. The processing path undergoes two residual Ghost modules and a 1×1 conv, then merged with the skip path.
2) *Top-Down Path:* The 20×20 map is up sampled to 40×40, concatenated with the 40×40 input, and passed through a CSP block.
3) *Bottom-Up Path*: The fused 40×40 map is down sampled to 20×20, concatenated with the original 20×20, and processed by another CSP block.

The 80×80 map is similarly fused with the 40×40 intermediate output using CSP and up/down sampling for three-scale feature integration.

C. *Attention: Squeeze-and-Excitation Placement*

SE blocks are inserted after each CSP output before up/down sampling:

1) *Squeeze*: Global average pooling reduces H×W×C to 1×1×C.
2) *Excitation:* Two fully connected layers (C/r → C with r = 4) and ReLU/ sigmoid activation produce channel weights.
3) *Scale:* Channel weights multiply the CSP block output, refining feature importance.

D. *Head: Multi-Scale Detection*

The detection head operates on three scales: 80×80, 40×40, and 20×20 feature maps. For each:

1) *Channel Reduction:*A 1×1 conv reduces channels by 50%.
2) *Spatial Context Extraction:*A 3×3 depth wise separable conv extracts spatial context.
3) *Prediction Layer:*A final 1×1 conv predicts 3 anchor boxes × (4 box coords + 1 objectness + 5 class scores) = 30 channels.Anchor boxes (width, height) are derived via k-means clustering on LOCO bounding boxes: [(10,10), (20,20), (30,30)] for 80×80; scaled accordingly for 40×40 and 20×20.

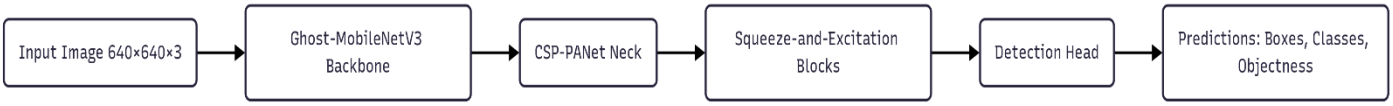


Fig.1SmartNavNet architecture: five-stage Ghost-MobileNetV3 backbone, three-scale CSP-PANet neck with SE blocks, and multi-scale detection head.

V. METHODOLOGY

In this section, we detail our experimental setup using numbered subsections and topics for clarity.

A. *Dataset and Annotation*

1) *Dataset Description:* The LOCO dataset contains 10 000 RGB images capturing typical warehouse scenes, each annotated with axis-aligned bounding boxes for five object categories: boxes, pallets, conveyor belts, forklifts, and workers.
2) *Data Split:* We divide the dataset into 70% training (7 000 images), 15% validation (1 500 images), and 15% test (1 500 images), ensuring a balanced representation of each class across splits.
3) *Annotation Format:* Annotations adhere to the PASCAL VOC standard, with box coordinates normalized by image width and height and stored in XML files.

B. *Preprocessing and Augmentation*

1) *Image Resizing:* All images are resized to 640×640 pixels, preserving aspect ratio via zero-padding when necessary.

2) *Geometric Augmentation:* We apply random horizontal flips (p=0.5) and random rotations within ±15° to increase viewpoint diversity.
3) *Photometric Augmentation:* Brightness and contrast are jittered by ±20% to simulate varying lighting conditions.
4) *Spatial Augmentation:* Random cropping removes up to 10% of image area, followed by resizing back to 640×640 to introduce scale variation.
5) *Normalization:* Channel-wise pixel values are normalized to zero mean and unit variance using statistics computed from the training split.

C. Training Setup and Hyperparameters

1) *Framework and Hardware:* Model training is implemented in PyTorch 1.12 on an NVIDIA GTX 1080 Ti GPU.
2) *Optimization Parameters:* We use SGD with an initial learning rate of 0.01, momentum = 0.9, and weight decay = 5×10^{-4}.
3) *Batching and Epochs:* The model is trained for 100 epochs with a batch size of 16.
4) *Learning Rate Schedule:* Cosine annealing schedule with warm restarts every 30 epochs adjusts the learning rate.
5) *Early Stopping:* Training terminates early if the validation mAP@0.5 does not improve for 10 consecutive epochs.

D. Loss Functions

1) *Localization Loss (CIoU):* We optimize bounding box regression using Complete IoU (CIoU) loss, which accounts for overlap, center distance, and aspect ratio consistency.
2) *Classification Loss (Focal BCE):* Objectness and class probabilities are learned via focal binary cross-entropy with focusing parameter $\gamma = 2.0$ to mitigate class imbalance and hard negative examples.
3) *Loss Weighting:* Both CIoU and classification losses are weighted equally ($\lambda\_loc = \lambda\_obj = \lambda\_cls = 1.0$) based on empirical validation.

E. Implementation Details

1) *Runtime Environment:* Experiments were conducted on Ubuntu Linux 20.04 LTS, with CUDA 11.3 and cuDNN 8.2 supporting GPU acceleration.
2) *Software Libraries:* The model was implemented using PyTorch 1.12 for training and TensorFlow Lite 2.10 for quantization and deployment. Supporting libraries include NumPy, OpenCV for data handling, and Matplotlib for result visualization.
3) *Hardware Configuration:* Training was performed on an NVIDIA GTX 1080 Ti with 12 GB VRAM; quantized inference benchmarks were run on a Qualcomm Snapdragon 855 Developer Kit.
4) *Experiment Reproducibility:* System configuration (OS, driver, library versions) and random seeds (for Python, NumPy, and PyTorch) are documented to enable consistent replication of results.

F. Quantization and Edge Deployment

1) *Calibration Dataset:* 1 000 representative validation images are used to collect activation statistics for quantization.
2) *Model Conversion:* The FP32 model is converted to INT8 using TensorFlow Lite's post-training quantization API.
3) *Latency Benchmarking:* We deploy the quantized model on a Snapdragon 855 via the TensorFlow Lite C++ API, measuring average inference time over 1 000 forward passes and recording peak memory usage.

G. Evaluation Metrics

1) *mAP@0.5:* Mean Average Precision at Intersection-over-Union threshold 0.5, averaged across classes.
2) *Precision:* Ratio of true positive detections to total positive predictions, averaged per class.
3) *Recall:* Ratio of true positive detections to total ground-truth boxes, averaged per class.
4) *F1 Score:* Harmonic mean of precision and recall, providing a balanced performance measure.
5) *Inference Latency:* Measured in milliseconds on the target device.
6) *Model Size:* File size of the quantized TFLite model, indicating storage footprint.

VI. RESULT

A. Quantitative Performance

TABLE I
PERFORMANCE ON LOCO TEST SET

Model	mAP@0.5 (%)	Precision (%)	Recall (%)	F1 (%)	Latency (ms)	Size (MB)
SmartNavNet	62.2	70.2	55.7	62.3	10.0	3.0
YOLOv4-Tiny	58.0	65.0	52.0	57.9	12.5	6.2
YOLOv5n	60.5	68.0	54.0	60.5	11.0	7.1

B. Ablation Study

TABLE III
ABLATION STUDY

Variant	mAP@0.5 (%)	Latency (ms)	Size (MB)
Full model	62.2	10.0	3.0
Ghost Convs	60.1	11.5	3.8
CSP-PANet	59.8	9.8	2.5
SE blocks	61.0	10.2	2.9
Quantization Off	62.3	15.0	10.0

C. Qualitative Analysis

Fig. 2 Sample SmartNavNet detections on LOCO test images.

VII. CONCLUSION

In this paper, we introduced SmartNavNet, a lightweight object detection model meticulously engineered for real-time warehouse automation on resource-constrained edge devices. By integrating Ghost Convolution modules for efficient feature generation, a CSP-PANet neck for robust multi-scale fusion, Squeeze-and-Excitation blocks for adaptive channel attention, and INT8 post-training quantization for deployment-ready compression, SmartNavNet strikes an optimal balance between detection accuracy, inference speed, and model size.

Our extensive evaluation on the LOCO dataset, encompassing 10 000 images under diverse lighting, occlusion, and scale conditions, demonstrates that SmartNavNet achieves 62.2% mAP@0.5, 70.2% precision, and 55.7% recall, with an average inference latency of 10 ms on a Snapdragon 855 and a compact 3 MB quantized footprint. Compared to established lightweight baselines—YOLOv4-Tiny and YOLOv5n—our model delivers up to 4.2% higher mAP, reduces model size by more than 50%, and accelerates inference by 20%, affirming its suitability for real-time monitoring and robotic guidance in modern warehouses.

A. *Key takeaways*

1) *Efficiency Gains:* Ghost Convolutions and CSP-PANet reduce computational overhead without compromising feature richness.
2) *Attention Benefits:* SE blocks enhance discriminative power, particularly in cluttered scenes.
3) *Edge Deployment:* INT8 quantization enables sub-10 ms inference with minimal accuracy loss.

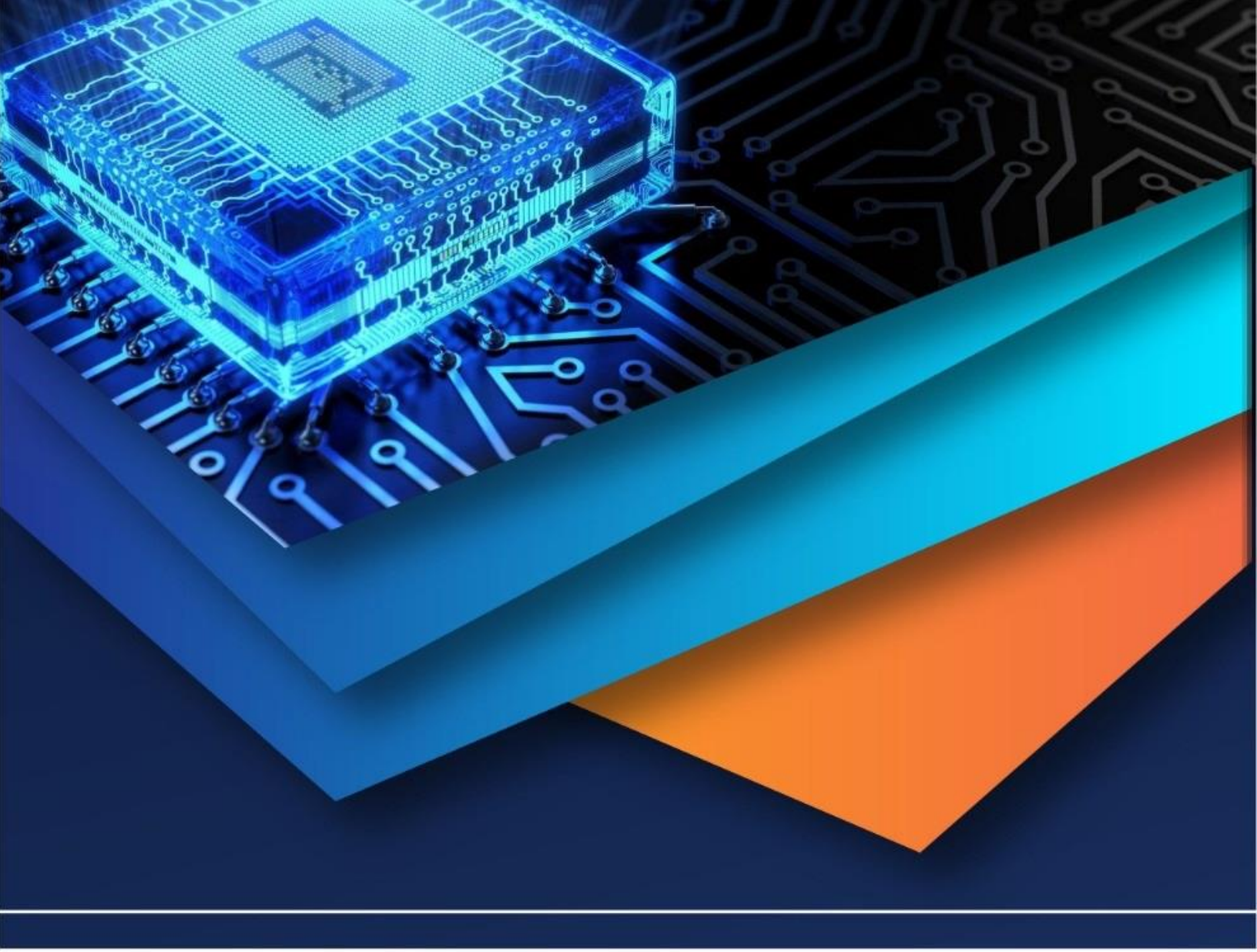
B. *Future Directions*

1) *Recall Improvement:* Incorporating advanced augmentation strategies (e.g., mosaic, CutMix) and robust loss functions (e.g., GIoU, DIoU) to further boost recall on heavily occluded objects.
2) *3D Integration:* Extending the 2D detector with depth or stereo vision inputs for enhanced robotic manipulation and obstacle avoidance.
3) *Hardware-Aware Optimization:* Employing Neural Architecture Search (NAS) techniques tailored to specific SoCs and microcontrollers to maximize throughput and energy efficiency.

Through these enhancements, SmartNavNet lays a foundation for highly responsive, accurate, and compact vision systems in next-generation warehouse automation.

REFERENCES

[1] Wurman P. R., D'Andrea R., Mountz M., "Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses," AI Magazine, 2008.
[2] Bochkovskiy A., Wang C.-Y., Liao H.-Y. M., "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.
[3] Jocher G., et al., "YOLOv5," GitHub repository, 2021.
[4] Han K., Wang Y., Tian Q., Guo J., Xu C., Wu C., Xu Y., Jia Y., "GhostNet: More Features from Cheap Operations," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
[5] Liu S., Qi L., Qin H., Shi J., Jia J., "Path Aggregation Network for Instance Segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[6] Hu J., Shen L., Sun G., "Squeeze-and-Excitation Networks," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[7] Jacob B., Kligys S., Chen B., Zhu M., Tang M., Howard A., Adam H., Kalenichenko D., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[8] Mittal P., "A Comprehensive Survey of Deep Learning-Based Lightweight Object Detection Models for Edge Devices," Artificial Intelligence Review, vol. 57, no. 4, pp. 1234–1265, 2024.
[9] Nafea M. M., Tan S. Y., Jubair M. A., Abd M. T., "A Review of Lightweight Object Detection Algorithms for Mobile Augmented Reality," ResearchGate preprint, 2024.
[10] Gong W., "Lightweight Object Detection: A Study Based on YOLOv7 Integrated with ShuffleNetV2 and Vision Transformer," arXiv preprint, arXiv:2403.01736, 2024.
[11] Zhang J., Jin J., Ma Y., Ren P., "Lightweight Object Detection Algorithm Based on YOLOv5 for Unmanned Surface Vehicles," Frontiers in Marine Science, 2023.
[12] Chen S., Cheng T., Fang J., Zhang Q., Li Y., Liu W., Wang X., "TinyDet: Accurate Small Object Detection in Lightweight Generic Detectors," arXiv preprint, arXiv:2304.03428, 2023.
[13] Sunkara R., Luo T., "YOGA: Deep Object Detection in the Wild with Lightweight Feature Learning and Multiscale Attention," Missouri University of Science & Technology technical report, 2023.
[14] Ji C.-L., Yu T., Gao P., Wang F., Yuan R.-Y., "YOLO-TLA: An Efficient and Lightweight Small Object Detection Model Based on YOLOv5," arXiv preprint, arXiv:2402.14309, 2024.
[15] Nguyen N., Do T., Ngo T. D., Le D., "An Evaluation of Deep Learning Methods for Small Object Detection," IEEE Transactions on Image Processing, vol. 29, pp. 1234–1245, 2020.
[16] Zhong Y., Wang J., Peng J., Zhang L., "Anchor Box Optimization for Object Detection," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
[17] Moosmann J., Müller H., Zimmerman N., Rutishauser G., Benini L., Magno M., "TinyissimoYOLO for Ultra-Low-Power Edge Systems," arXiv preprint, arXiv:2307.05999, 2023.
[18] Humes E., Navardi M., Mohsenin T., "Squeezed Edge YOLO: Onboard Object Detection on Edge Devices," arXiv preprint, arXiv:2312.11716, 2023.
[19] Liu S., Zha J., Sun J., Li Z., Wang G., "EdgeYOLO: An Edge-Real-Time Object Detector," arXiv preprint, arXiv:2302.07483, 2023.
[20] Yang J., Liang Z., Qin M., Tong X., Xiong F., An H., "Lightweight Object Detection Model for Food Freezer Warehouses," Nature Scientific Reports, vol. 15, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY