EFFICIENT SMALL OBJECT DETECTION YOLO MODEL FOR EDGE COMPUTING

A PREPRINT

Jinze Huang*
Department of Computer Science
New York University
New York, USA
jh9108@nyu.edu

Ning Miao†
Courant Institute of Mathematical Sciences
New York University
New York, USA
nm4543@nyu.edu

ABSTRACT

This research focuses on enhancing real-time small object detection (SOD) on edge devices by modifying the YOLOv8s architecture. We address the challenges of detecting small objects, particularly traffic signs in the TT100K dataset, by integrating a Global Attention Module (GAM) and optimizing the detection heads for shallow features (P2/3/4) and removing P5. Our proposed model achieves a mAP50 of 0.91226, a 58.8% improvement over the baseline. Model compression techniques, including pruning and INT8 quantization, are also investigated, yielding up to a sixfold increase in inference speed on an NVIDIA A100 GPU with minimal accuracy loss, which reduce model from 10.0M to 5.46M, GFLOPS from 102.4 to 50.7, and inference speed up to 6 times faster with acceptable metrics loss. However, these reductions are less than 10%. The proposed model outperforms other lightweight models and demonstrates significant potential for robust and efficient SOD in resource-constrained environments.

Keywords YOLO, Small Object Detection, Edge Computing, Attention Mechanism, Model Compression

1 Introduction

With the rapid development of the computer vision field in recent years, object detection technology has become increasingly efficient and accurate. However, deploying efficient and accurate real-time small object detection algorithms in edge devices remains a challenge task. Concretely, small objects contain very few pixels, making feature extraction difficult. Also, many object detection algorithms need to resize the original input to a lower resolution form (e.g 640×640 pixels), When resizing high resolution images, small objects may lose critical information. Moreover, there is a technical contradiction, small objects detection need deeper networks but edge devices require lightweight models, so we need to do a trade-off. Before neural networks were widely used, the mainstream of SOD algorithms was based on traditional image processing. It was highly relied on manual feature extraction[1] such as Haar features, HOG [2], and etc. In the post-Neural Nerwork era, object detection algorithms based on CNN (convolution neural network) dominated the development of the object detection field. With diverse datasets and additional semantic information, the performance of SOD has been improved.

For SOD, an individual object usually only occupies a tiny part of the picture with low resolution, occlusion, and background noise. These characteristics make normal object detection models hard to identify tiny individuals, therefore, we need specific SOD technology to detect and localize tiny objects accurately. SOD focuses on addressing the object with low resolution and weak feature representation which is easily influenced by background noise. In contrast, normal object detection models are applied when an object occupies a relatively big part of the picture. Hence, SOD faces problems arising from higher rates of missed and false detection, and it needs more sophisticated feature extraction and multi-scale adaptation techniques to detect accurately[3, 4].

*Supervise the research project, conceptualize the methodology, and lead the experimental design and implementation

†Contribute to experimental design and implementation, improve the model architecture, and handle the documentation

YOLO is a fast and efficient object detection algorithm widely used for real-time detection. By modifying its feature extraction and pyramid network modules, we can improve accuracy while keeping it lightweight enough for edge devices[5]. The model's innovative backbone structure enables high-accuracy real-time detection even with limited computing resources. Recent improvements like YOLO-RET have shown promising results in edge computing scenarios by enhancing system responsiveness through localized processing[6]. With my research deadline approaching rapidly, I chose YOLO as it provides a good balance between performance and development time, while still allowing room for novel improvements in small object detection through careful architectural modifications.

In this research, we focus on the following works:

1. Modify the YOLO architecture by adjusting transformer-based and CNN-based backbone modules, neck and detection heads and feature map fusion to create a light-weight model balancing performance and speed.
2. Attention module is compared and used to improve the detection accuracy.
3. Train the modified YOLO model on the TT100K dataset[7], which contains small traffic signs in diverse conditions.
4. Compare different modified YOLOv8 model's performance and conduct ablation experiments to validate the proposed changes.
5. Lightweight model by quantization and pruning and test on V100, A100 and Orin AGX X.

2 Related Work

2.1 Small Object Detection

In image processing, "small objects" are typically defined by their pixel area or relative size compared to the entire image. In this research, we define small objects as those with a pixel area less than 32×32 pixels or occupying less than 0.1% of the total image area[8]. When the input image resolution is high, we need to downsample the high-resolution image to the network input size (e.g., 640×640 pixels) with a downsampling factor s. For an input image with dimensions $\boldsymbol{x} = (x_1, x_2)$, $\boldsymbol{x} \in \mathbb{R}^{H \times W}$, it becomes $\acute{\boldsymbol{x}} = (\acute{x_1}, \acute{x_2})$, $\acute{\boldsymbol{x}} \in \mathbb{R}^{\frac{H}{s} \times \frac{W}{s}}$. During this process, small objects may lose significant high-frequency information, making their features less distinct and harder to detect. The loss of high-frequency information during downsampling can be understood through Fourier transform. Images can be represented as a superposition of sine and cosine waves of different frequencies. High-frequency components correspond to fine details and small objects in the image. Downsampling causes these high-frequency components to be discarded or aliased, resulting in information loss.

$$\acute{\boldsymbol{x}} = \text{Downsample}(\boldsymbol{x}, s) \tag{1}$$

Due to the value of industry and society safety, small object detection(SOD) is always a hot spot in the object detection area. Boshra Khalili and Andrew W.Smyth proposed an improved YOLOv8 by improving GFPN structure, adding c2-EMA modules, and modifying PIoU loss, achieving significant improvements in mAP0.5 from 40.6% to 45.1% in traffic CCTV.[9] Shi et al. proposed FocusDet in 2024, a specialized small object detection framework, integrating STCF-EANet backbone and Bottom FocusPAN, which improved mAP@.5 by 13.1% and 6.2% on VisDrone2021-DET and CCTSDB2021 datasets, respectively.[10] Remote sensing images also present significant challenges in object detection due to the prevalence of small-scale targets. Wang et al. introduced a novel arbitrary-oriented object detection framework for remote sensing images, incorporating a periodic pseudo-domain angle representation and ratio-balanced loss, achieving competitive results on multiple datasets.[11] Comparasion of different object detection algorithms is shown in Fig. 1.

2.2 Backbone

Backbone in deep learning serves as the primary feature extractor network, responsible for processing input data and computing a hierarchical representation of features, which can then be utilized by subsequent task-specific modules for various applications such as object detection, segmentation, or classification. Apple published a new Backbone called MobileOne[12] in 2023. It reduced memory access cost(MAC) by forcing synchronization like global pooling used in Squeeze-Excite block, using skip connections and squeeze-excite blocks and different structure of training and inference. GhostNetV2 captured the dependence between long-range pixels by fully-connected(FC) layers to aggregate local and long-range information simultaneously.

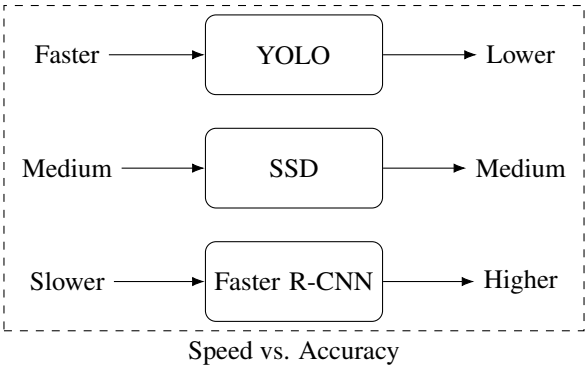


Figure 1: Comparison of Object Detection Algorithms (Speed vs. Accuracy)

2.3 Edge Computing

With the proliferation of the Internet of Things (IoT), edge computing has emerged to meet the demand for faster and more reliable data processing. The core principle of edge computing is to deploy computational resources closer to devices, reducing transmission latency and enhancing data processing efficiency. Qiu et al.[13] proposed LD-YOLOv10, a lightweight drone target detection algorithm that incorporates a novel RGELAN feature extraction structure, an AIFI module with multi-head attention, a DR-PAN Neck structure, and a combination of Wise-IoU and EIoU loss functions. Liu et al.[14] proposed a lighter decoupled head with fewer channels and convolutional layers, incorporating implicit representation layers and parameterization techniques to improve regression performance while reducing inference costs and enhancing parallel computation capabilities.

3 Datasets and Augmentation

3.1 Tsinghua-Tencent 100K

The Tsinghua-Tencent 100K (TT100K) dataset is a large-scale traffic sign detection and classification dataset collected from more than 100,000 images across China(Fig. 2). According to our analysis(Table 1), the dataset contains 24,052 images in total, with 20,454 for training and 3,598 for validation, maintaining approximately a 5.8:1 train/validation ratio. The dataset encompasses 74,008 annotated objects, including 63,185 objects in the training set and 10,823 objects in the validation set. These objects are categorized into 50 different classes of traffic signs.

The size distribution analysis(Fig. 3) reveals that most than 60% objects are less than 4096 pixels compared to the image size of 2048x2048. TT100K has an average width of 45.5 pixels and height of 49.4 pixels, making it suitable for small object detection research. The spatial distribution heatmap(Fig. 3a) indicates that traffic signs are predominantly located in the central and upper regions of the images, which aligns with typical traffic scene compositions. This comprehensive dataset provides a robust benchmark for evaluating traffic sign detection algorithms, especially those focusing on small object detection in real-world scenarios.

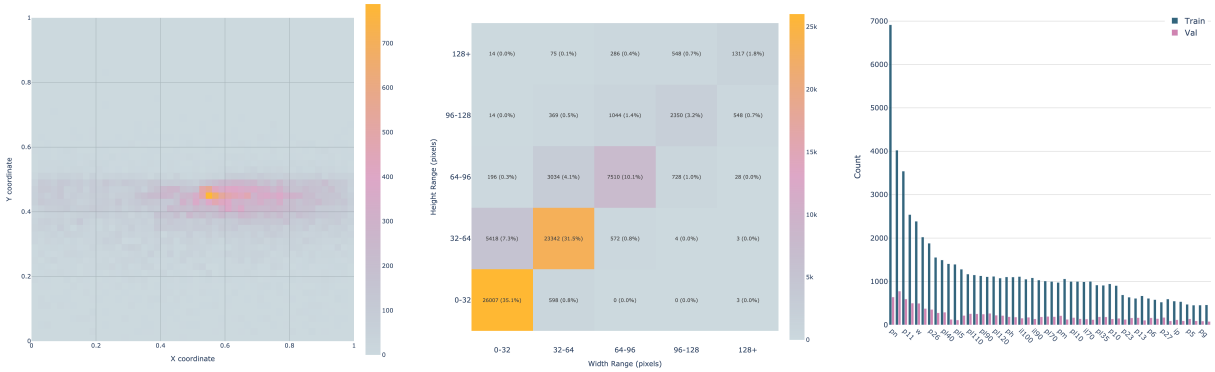
(a) Example image 1

(b) Example image 2

Figure 2: Examples from TT100K dataset

Metric	Value
Total Images	24,052
Training Images	20,454
Validation Images	3,598
Total Objects	74,008
Training Objects	63,185
Validation Objects	10,823
Number of Categories	50
Train/Val Ratio	5.84
Mean Width	45.5
Mean Height	49.4
Mean Area	3191.5
Image Size	2048x2048

Table 1: Dataset Statistics



(a) Spatial distribution (b) Size distribution (c) Category distribution

Figure 3: Distribution analysis of TT100K dataset

3.2 Augmentation

The normal augmentation methods used in researches can be divided into two categories: **Basic Augmentations** and **Advanced Augmentations**.

- **Basic Augmentations**:
  - Non-Geometric: Flip, Crop, Color, Noise, Kernel
  - Geometric: Rotate, Translate, Shear
  - Erasing: Random, Hide-and-Seek, GridMask
- **Advanced Augmentations**:
  - Multi-Image Mixing: Mixup, CutMix, SaliencyMix
  - Single Image Mixing: Local Aug, Self-Aug
  - Auto Augment: Random/ADA, RL-based
  - Feature Level: Feature Space Aug, Style Transfer

4 Methods

4.1 YOLO

YOLO series is a one-stage end-to-end SOTA object detection network. Since the first YOLO was published, it has been the SOTA model among the object detection models. With the help of many computer vision researchers, the YOLO family has been upgrading to YOLOv11 [15] published in Sep 2024. For now, the YOLO network mainly consists of 3 parts, Backbone, Neck, and Detection Head(Fig. 12). Figure 13 shows the performance of different YOLO models.

4.2 Global Attention Module

Attention mechanism is a key component in deep learning now, it enables models to focus on relevant features and ignore irrelevant ones by assigning different weights to different features. In this research, we used two Global Attention Module (GAM) that integrates global information into the YOLO model. GAM enhances feature representation capabilities by introducing channel attention and multi-scale spatial attention mechanisms. For a given input feature map $F \in \mathbb{R}^{C\times H\times W}$, GAM first uses the channel attention sub-module to calculate importance weights $M_c \in \mathbb{R}^{C\times 1\times 1}$ for each channel, then uses the spatial attention sub-module to calculate importance weights $M_s \in \mathbb{R}^{1\times H\times W}$ for different spatial positions. The final output feature map F'' can be expressed as:

$$F' = M_c(F) \otimes F \tag{2}$$

$$F'' = M_s(F') \otimes F' \tag{3}$$

where $\otimes$ represents element-wise multiplication.

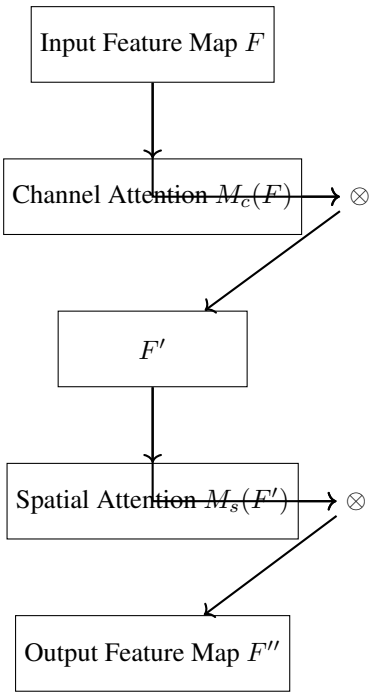


Figure 4: Global Attention Module (GAM) Structure

Compared with Convolutional Block Attention Module(CBAM), GAM can capture more global information and improve the detection accuracy of small objects. The sequential architecture of GAM(Fig. 5) demonstrates superior efficiency by reducing computational overhead while maintaining detection accuracy. When combined with P2/3/4 detection heads, its hierarchical processing better preserves fine-grained spatial information critical for small objects. Additionally, the streamlined memory footprint makes it particularly suitable for edge deployment scenarios.

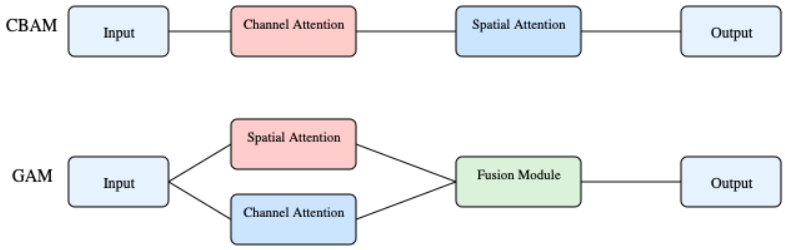
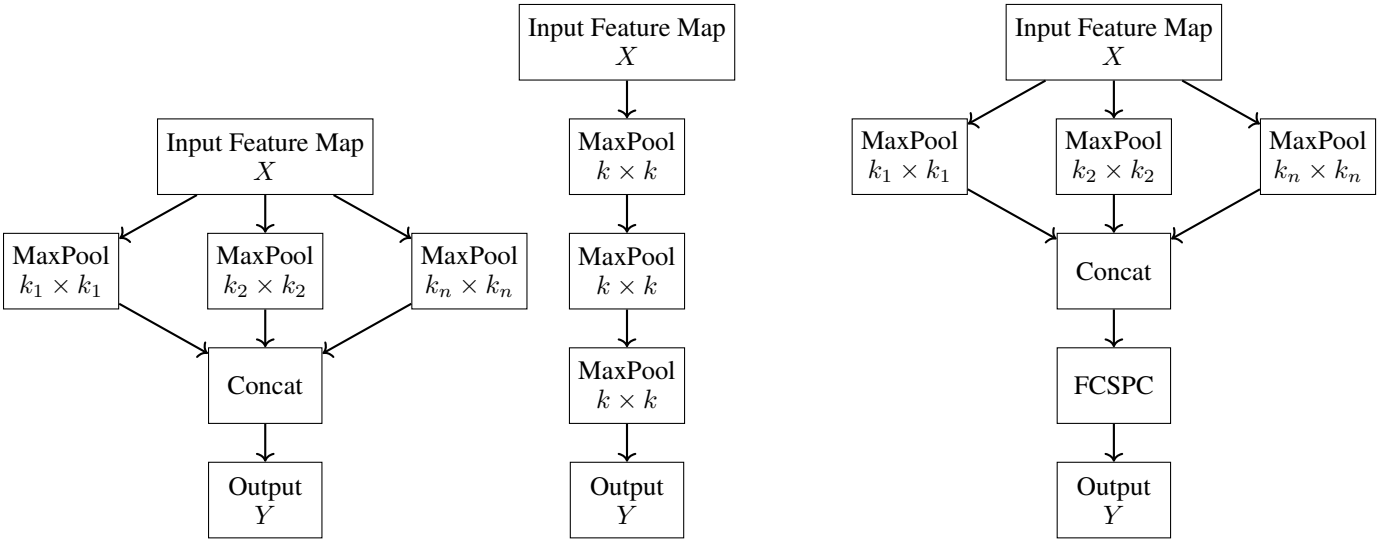


Figure 5: CBAM vs GAM

4.3 SPP family

The SPP (Spatial Pyramid Pooling)(6a) module performs pooling operations on feature maps using different sized pooling kernels and concatenates the results to obtain multi-scale feature representations. For an input feature map X, the output Y of SPP can be expressed as:

$$Y = \text{Concat}(\text{MaxPool}(X, k_1), \text{MaxPool}(X, k_2), ..., \text{MaxPool}(X, k_n)) \tag{4}$$



(a) SPP Module (b) SPPF Module (c) SPPFCSPC Module

Figure 6: Comparison of different SPP modules

SPPF (Spatial Pyramid Pooling - Fast) (6b) is an improved version of SPP that reduces computational cost and increases speed by using multiple cascaded small-sized pooling kernels instead of the large pooling kernels in SPP. For a given input feature map X, the output Y of SPPF can be expressed as:

$$Y = \text{MaxPool}(\text{MaxPool}(\text{MaxPool}(X, k), k), k) \tag{5}$$

SPPFCSPC (Spatial Pyramid Pooling - Fully Connected Spatial Pyramid Convolution) (6c) combines SPP with Fully Connected Spatial Pyramid Convolution (FCSPC) to enhance the network's ability to extract features of different scales and effectively mitigate the impact of object scale variations caused by different factors such as shooting angles and image segmentation. The expression can be written as:

$$Y = \text{FCSPC}(\text{Concat}(\text{MaxPool}(X, k_1), \text{MaxPool}(X, k_2), ..., \text{MaxPool}(X, k_n))) \tag{6}$$

4.4 Pruning

Pruning is a compression technique that compresses a model by removing channels that contribute little to feature extraction in a neural network, with the goal of reducing computational resource consumption (reduce GFLOPS) and model size (reduce model parameters) while maintaining model performance. In this research, we use a pruning method called LAMP (LAYER-ADAPTIVE SPARSITY FOR THE MAGNITUDE-BASED PRUNING) [16], which determines the pruning order by calculating the relative importance score of each weight (weight squared divided by the sum of the squares of all surviving weights in the layer (Eq. 7)). After the pruning weight files were obtained, map, precision and recall of the model were reduced, and these important metrics were improved again by fine-tuning.

$$\text{LAMP\_score}(w_i^{(l)}) = |w_i^{(l)}|\sqrt{n_l} \text{ where } n_l \text{ is the number of neurons in layer } l \tag{7}$$

4.5 Quantization

Quantization is a technique that converts model weights and activation values from floating-point representation to low-bit integer representation. For a given floating-point number x_{fp32}, the process of quantizing it to an INT8 integer q_{int8} can be expressed as:

$$q_{\text{int8}} = \text{round}\left(\text{clip}\left(\frac{x_{\text{fp32}}}{s}, -128, 127\right)\right) \tag{8}$$

where s is the scaling factor, $\text{clip}(x, a, b)$ represents limiting the value of x to the range $[a, b]$, and $\text{round}(x)$ represents rounding x to the nearest integer. The dequantization process can be expressed as:

$$x_{\text{fp32}} = s \cdot q_{\text{int8}} \tag{9}$$

The choice of scaling factor s affects the precision and dynamic range of quantization. Common methods include scaling based on maximum absolute value and scaling based on KL divergence[17].

5 Experiment and Analysis

5.1 Our Model

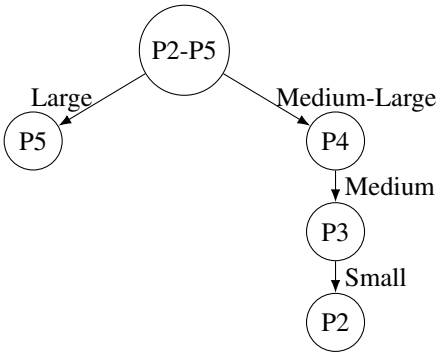


Figure 7: Feature Pyramid Levels and Object Sizes

Figure 8: Comparison of Models with and without P5 Layer

Model	Precision	Recall	mAP50	mAP50-95
With P5	0.89641	0.80526	0.90652	0.73618
Ours	0.9154	0.81115	0.91226	0.7259

In the feature pyramid network (FPN), different layers of feature maps are responsible for detecting objects of different sizes. As shown in Figure 7, the P2 layer has the highest spatial resolution and is suitable for detecting small objects; the P3 and P4 layers are responsible for detecting medium-sized objects; and the P5 layer has lower spatial resolution and is suitable for detecting large objects. Since the TT100K dataset mainly contains small objects, the P5 layer is limited in its effectiveness, and removing the P5 layer can reduce the model's computational load and improve inference speed, while also avoiding excessive downsampling that leads to loss of small object information. To verify this, we conducted a comparison experiment (Table 8). Our model (Fig. 9) is a modified YOLOv8s model with GAM attention and P2/3/4 detection heads. It achieves the best performance in all evaluation indicators, among which mAP50 reached 0.91226, an increase of 58.8% compared with the benchmark model (yolov8s). This result shows the effectiveness of our proposed improved strategy in small object detection tasks. We add two GAM modules, one is after 1st conv layer,

and the other is before SPPF-CSPC layer. We also removed the P5 layer and use P2 instead. The features of small objects are more concentrated in the shallow layers, and the features of large objects are more concentrated in the deep layers. So we add two GAM modules to capture the features of small objects and large objects. Also, we replaced the SPPF with SPPF-CSPC, which is a modified SPPF that uses CSPC instead of C3. CSPC is a new structure that is more suitable for small object detection tasks and faster than SPPF.

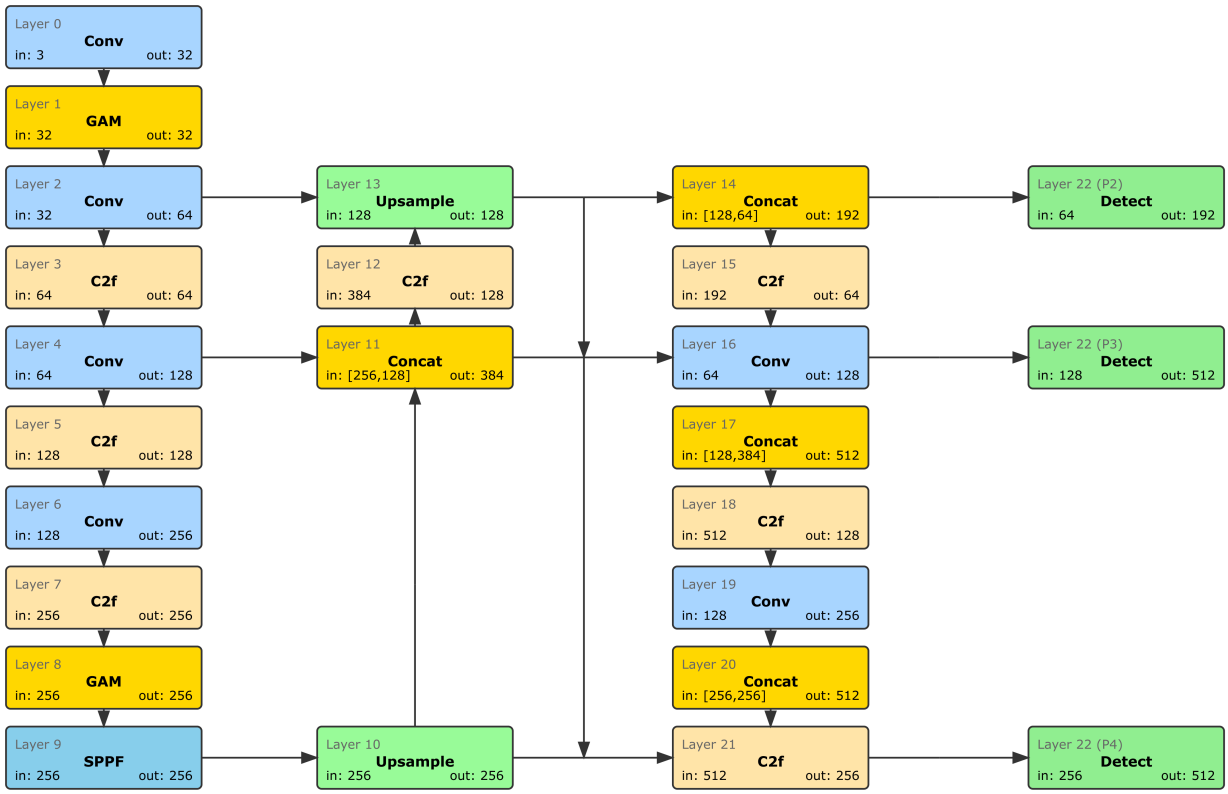


Figure 9: Our Modified YOLO with GAM attention and P2/3/4 detection heads

5.2 YOLO Series Comparison

Table 2: Performance Comparison: YOLOv8, YOLOv9, and YOLOv10

Model Name	Precision (B)	Recall (B)	mAP50 (B)	mAP50-95 (B)
yolov8s	0.65109	0.50092	0.5777	0.46824
yolov9	0.62836	0.5042	0.57608	0.46669
yolov10s	0.58348	0.48222	0.53559	0.43603

The comparison of YOLOv8s, YOLOv9, and YOLOv10s reveals relatively similar performance metrics, with YOLOv8s demonstrating a slight edge in overall performance. However, these relatively small differences may become more pronounced in specific application domains, such as small object detection, highlighting the need for further investigation.

5.3 Backbone Comparison

Table 3 compares several lightweight backbones integrated with YOLOv8s, evaluating their effectiveness for edge computing applications, particularly for small object detection. The baseline model, YOLOv8s, exhibits lower performance metrics than the other models that utilize more efficient backbones. Notably, the C2f-GhostBlock backbone demonstrates significant improvements across all metrics. Our proposed model, incorporating the Global Attention Mechanism (GAM) with a P2/3/4 output layer configuration, achieves superior performance compared to the

Table 3: YOLOv8s Baseline and Lightweight Models for Edge Computing

Model Name	Precision (B)	Recall (B)	mAP50 (B)	mAP50-95 (B)
baseline(yolov8s)	0.65109	0.50092	0.5777	0.46824
FasterNet	0.81183	0.59062	0.71633	0.47712
C2f-GhostBlock	0.89491	0.76432	0.87806	0.69342
MobileNetv4	0.81696	0.59044	0.7147	0.51869
GhostNetv2	0.83853	0.64894	0.77163	0.63156
MobileNeXt-xxs	0.72358	0.4831	0.59016	0.31933
ours(GAM_P2/3/4)	**0.9154**	**0.81115**	**0.91226**	**0.7259**

Table 4: Transformer-based Models: MobileViT and EfficientViT

Model Name	Precision (B)	Recall (B)	mAP50 (B)	mAP50-95 (B)
MobileNetv4	0.81696	0.59044	0.7147	0.51869
EfficientViT	0.73639	0.59134	0.68897	0.55251

other models, with the highest precision, recall, mAP50, and mAP50-95 values. These results suggest that our model has considerable potential for accurate and robust small object detection in resource-constrained edge environments.

5.4 Transformer-based Models

We also adopt two different light weight transformer based models as backbone, they are Mobilenetv4[18] and EfficientVIT[19] . As you can see from the table 4, Although their internal network structure also has attention mechanism and feature extraction ability, their performance in detection metric is relatively mediocre.

5.5 Ablation Experiment

Table 5 and Figure 10 present the results of an ablation study investigating the impact of different attention modules and configurations on the performance of our model. Our full model, which incorporates two GAM modules along with the SPPCSPC layer and modified output layers (P2/3/4), achieves the highest overall performance, demonstrating superior precision, recall, mAP50, and mAP50-95. Replacing two GAMs and SPPCSPC with a single GAM and SPPF layer ("GAM with SPPF") results in a slight decrease in precision, recall and mAP50, but a notable increase in mAP50-95. Introducing only a single GAM module to the baseline model ("singleGAM") leads to a more modest drop in overall performance compared to our full model. Replacing the GAM modules with spatial attention ("SAM") or a coordinate attention module across P2/3/4 output layers ("CAM") further diminishes performance across all metrics, particularly mAP50-95. These findings underscore the effectiveness of the GAM module, suggesting that it plays a crucial role in enhancing the model's ability to focus on relevant features and improve detection accuracy. Furthermore, the superior performance of our full model suggests a synergistic effect between using two GAMs and the SPPCSPC, and removing P5 enhances performance.

In order to further analyze the influence of different modules on the model performance, we studied the contribution of each modification through the control variable method. Therefore, our ablation experiments is conducted as shown in Figure 10

- **base (11.17M)** : The original YOLOv8s model as the baseline
- **MobileNeXt (7.41M)** : Use lightweight backbone instead

Table 5: Ablation Study with Different Modules

Model Name	Precision (B)	Recall (B)	mAP50 (B)	mAP50-95 (B)
GAM with SPPF	0.89641	0.80526	0.90652	0.73618
singleGAM	0.90827	0.78842	0.89547	0.7001
SAM	0.89077	0.7662	0.87621	0.67961
CAM	0.90306	0.76122	0.87979	0.69293
ours	**0.9154**	**0.81115**	**0.91226**	**0.7259**

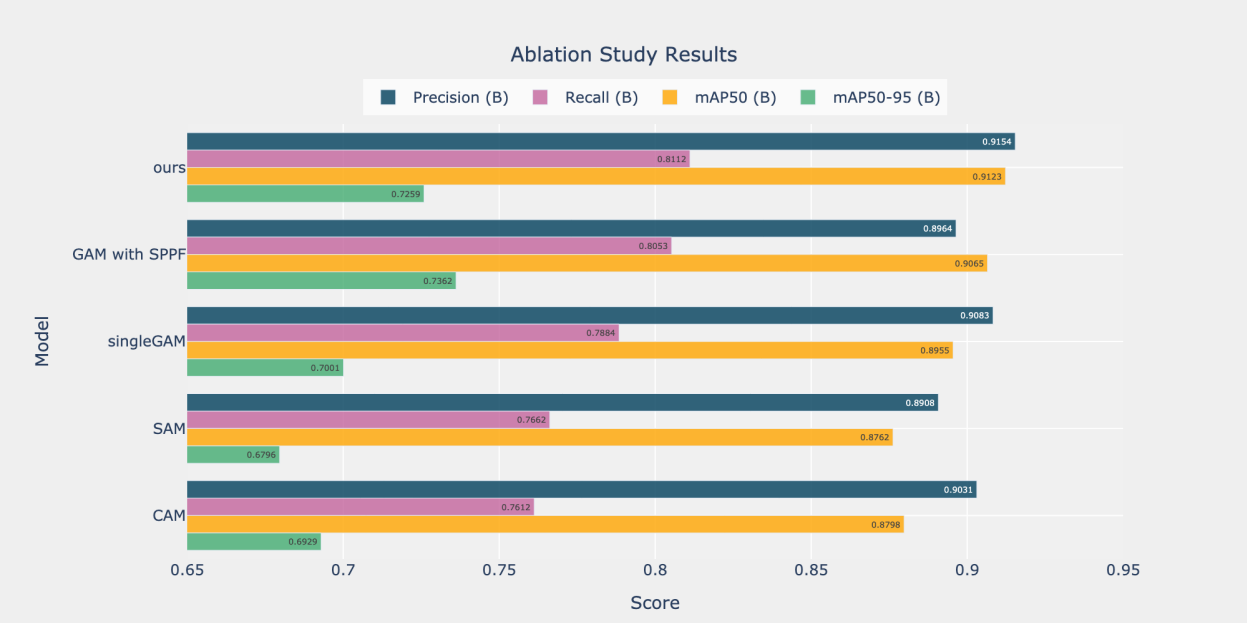


Figure 10: Ablation Experiment

- **GAM P2-5 (23.87M)** : Added GAM attention mechanism and P2 detect head
- **base with P2-5 (10.86M)** : Adds P2 detect head to the base model
- **GAM P2-4 SPPF (11.75M)** : Our complete improvement but keep SPPF
- **CBAM_smalltarget (5.14M)** : A lightweight version that uses the CBAM attention mechanism
- **single GAM (8.14M)** : Only a single GAM module is used
- **ours (10.01M)** : Our final modification

From the experimental results, we can observe that the GAM attention mechanism brings a significant improvement, compared to the original base model, precision improved from 0.65 to 0.87, and mAP50 also improved from 0.58 to 0.85. In terms of detection head optimization, the addition of P2 detection head significantly improves the detection ability of the model for small objects, which can be seen in the base with P2-5 configuration, whose mAP50-95 increases from 0.47 to 0.61.

More importantly, when we removed the P5 detection head, the model not only maintained high performance, but also reduced the number of parameters from 11.7M to 10.01M, and reaches a nice detection metrics where precision 0.92, recall 0.81, mAP50 0.91. For attention mechanism, the experiment proves that GAM has obvious advantages over CBAM. This ablation experiment not only validate the effectiveness of each modification we add, but also demonstrate that the combination of these components leads to an ideal performance improvements, which assists our design decisions for SOD optimization.

5.6 Model compression

In Figure 11 we compare the number of channels before and after pruning. We can see that in the shallow and middle layers of the network, more channels are pruned. In particular, a large number of channels are eliminated in the conv layers near to the neck. Acording to table 6, after pruning, our model parameters and GFLOS have been reduced by nearly half from 10M to 5.46M and from 102.4 to 50.7, which is more suitable for our edge deployment. However, dection metrics including recall, precision and mAP50 only reduced within 1%. After pruning and quantization, our model inference speed is up to 6 times faster, and the FPS is increased from 104.17 to 666.67. Although pruning plus int8 quantization does the same number of operations as pruning alone, we get a significant speedup compared to floating-point arithmetic because of the int8 type. Honestly, while gaining speed improvement, we also encountered some loss in detection metrics, we sacrificed recal,l precision and mAP50, but these reductions were within 10%.

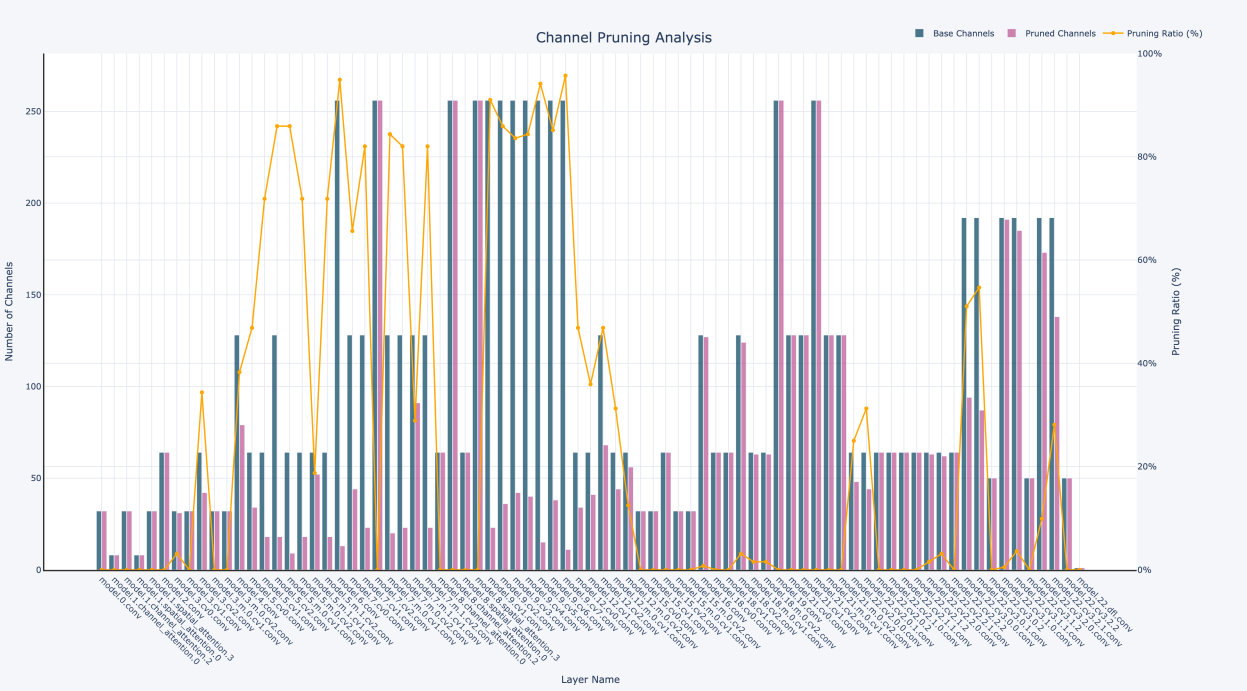


Figure 11: Channel pruning analysis

Metrics		Original	Pruning	Pruning+TensorRT INT8
FPS	V100	91.15	114.98	229.97
	A10	104.17	131.87	666.67
	A100	117.19	148.36	853.40
Parameters (Millions)		10.0	5.46	5.46
GFLOPS or GOPS		102.4	50.7	50.7
Precision		0.91413	0.90891	0.83381
Recall		0.81045	0.80400	0.69251
mAP50		0.91183	0.90552	0.82779
mAP50-95		0.73862	0.78170	0.68005

Table 6: Inference Performance Comparison

6 Limitations and Discussion

While our model achieves strong performance, several limitations should be noted:

1. The model's performance may degrade when dealing with extremely small objects (< 16×16 pixels) or under severe occlusion conditions.

2. The dual GAM modules introduce additional computational overhead, which could be problematic for devices with very limited computational resources.

3. The current pruning strategy is based on magnitude, which may not be optimal for all deployment scenarios. More sophisticated pruning methods could potentially achieve better compression-accuracy trade-offs.

These limitations suggest directions for future research, including developing more efficient attention mechanisms and exploring hardware-aware compression strategies.

7 Conclusion and Future Work

This research demonstrates the effectiveness of integrating a Global Attention Module (GAM) and optimizing detection heads (P2/3/4, without P5) within the YOLOv8s architecture for improved small object detection, particularly for traffic

signs in the TT100K dataset. Our model achieves state-of-the-art performance, evidenced by a mAP50 of 0.91226. Ablation studies confirm the significant contributions of the dual GAM integration and SPPCSPC layer. Furthermore, model compression via pruning and quantization substantially reduces model size and computational requirements, leading to a significant boost in inference speed (up to sixfold) while maintaining acceptable accuracy. The results underscore the feasibility of deploying advanced deep learning models on edge devices for real-time SOD.

Future work will focus on several key areas:

1. **Model Refinement:** Further refining the model architecture by exploring alternative attention mechanisms and optimizing the balance between accuracy and efficiency.
2. **Advanced Data Augmentation:** Investigating advanced data augmentation techniques to enhance model robustness and generalizability across diverse scenarios.
3. **Generalization Ability:** Exploring more datasets about traffic object detection and adapt our model to these datasets to improve the generalization ability.
4. **Edge Device Deployment:** Evaluating the optimized model's performance on various edge computing devices, including the NVIDIA Orin AGX X, to validate its real-world applicability and efficiency.
5. **Tracking New Methods:** Continuously monitoring advancements in image processing and deep learning for potential integration into the SOD pipeline.

References

[1] Xiuling, Zheng et al. "Starting from the structure: A review of small object detection based on deep learning". In: *Image and Vision Computing* 146 (2024), p. 105054. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2024.105054. URL: https://www.sciencedirect.com/science/article/pii/S0262885624001586.

[2] Dollar, Piotr et al. "Pedestrian detection: A benchmark". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pp. 304–311. DOI: 10.1109/CVPR.2009.5206631.

[3] Li, Bo, Huang, Shengbao, and Zhong, Guangjin. "LTEA-YOLO: An Improved YOLOv5s Model for Small Object Detection". In: *IEEE Access* 12 (2024), pp. 99768–99778. DOI: 10.1109/ACCESS.2024.3429282.

[4] Han, Guang et al. "Small Object Detection Based on Microscale Perception and Enhancement-Location Feature Pyramid". In: *IEEE Transactions on Cognitive and Developmental Systems* (2024), pp. 1–15. DOI: 10.1109/TCDS.2024.3397684.

[5] Ni, Jianjun et al. "A Small-Object Detection Model Based on Improved YOLOv8s for UAV Image Scenarios". In: *Remote Sensing* 16.13 (2024). ISSN: 2072-4292. DOI: 10.3390/rs16132465. URL: https://www.mdpi.com/2072-4292/16/13/2465.

[6] Ganesh, P. et al. "YOLO-ReT: Towards High Accuracy Real-time Object Detection on Edge GPUs". In: *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 1311–1321. DOI: 10.1109/WACV51458.2022.00138. URL: https://doi.ieeecomputersociety.org/10.1109/WACV51458.2022.00138.

[7] Zhu, Zhe et al. "Traffic-Sign Detection and Classification in the Wild". In: June 2016, pp. 2110–2118. DOI: 10.1109/CVPR.2016.232.

[8] Lin, Tsung-Yi et al. "Microsoft COCO: Common objects in context". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[9] Khalili, Boshra and Smyth, Andrew W. *SOD-YOLOv8 – Enhancing YOLOv8 for Small Object Detection in Traffic Scenes*. 2024. arXiv: 2408.04786 [cs.CV]. URL: https://arxiv.org/abs/2408.04786.

[10] Shi, Yanli, Jia, Yi, and Zhang, Xianhe. "FocusDet: an efficient object detector for small object". en. In: *Scientific Reports* 1 (May 2024), p. 10697. (Visited on 10/01/2024).

[11] Wang, Minghui et al. "Highly Efficient Anchor-Free Oriented Small Object Detection for Remote Sensing Images via Periodic Pseudo-Domain". en. In: *Remote Sensing* 15.15 (Aug. 2023), p. 3854. ISSN: 2072-4292. DOI: 10.3390/rs15153854. URL: https://www.mdpi.com/2072-4292/15/15/3854 (visited on 10/01/2024).

[12] Lee, Youngwan et al. *An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection*. 2019. arXiv: 1904.09730 [cs.CV]. URL: https://arxiv.org/abs/1904.09730.

[13] Qiu, Xiaoyang et al. "LD-YOLOv10: A Lightweight Target Detection Algorithm for Drone Scenarios Based on YOLOv10". In: *Electronics* 13.16 (2024). ISSN: 2079-9292. DOI: 10.3390/electronics13163269. URL: https://www.mdpi.com/2079-9292/13/16/3269.

[14] Liu, Shihan et al. *EdgeYOLO: An Edge-Real-Time Object Detector*. 2023. arXiv: 2302.07483 [cs.CV]. URL: https://arxiv.org/abs/2302.07483.

[15] Jocher, Glenn and Qiu, Jing. *Ultralytics YOLO11*. Version 11.0.0. 2024. URL: https://github.com/ultralytics/ultralytics.

[16] Lee, Jaeho et al. "Layer-adaptive Sparsity for the Magnitude-based Pruning". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=H6ATjJ0TKdf.

[17] Nowak, Grzegorz et al. "On the relaxation of the 8-bit quantization constraint on the weights of neural networks". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 969–977.

[18] Daquan, Zhou et al. *Rethinking Bottleneck Structure for Efficient Mobile Network Design*. 2020. arXiv: 2007.02269 [cs.CV]. URL: https://arxiv.org/abs/2007.02269.

[19] Liu, Xinyu et al. *EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention*. 2023. arXiv: 2305.07027 [cs.CV]. URL: https://arxiv.org/abs/2305.07027.

A Figures

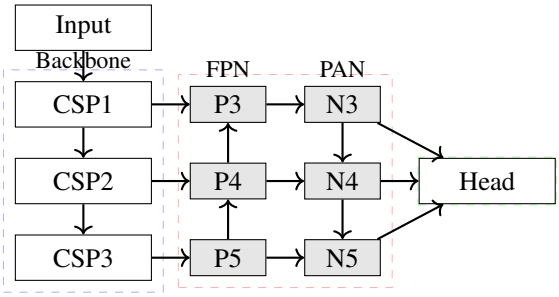


Figure 12: YOLO network architecture

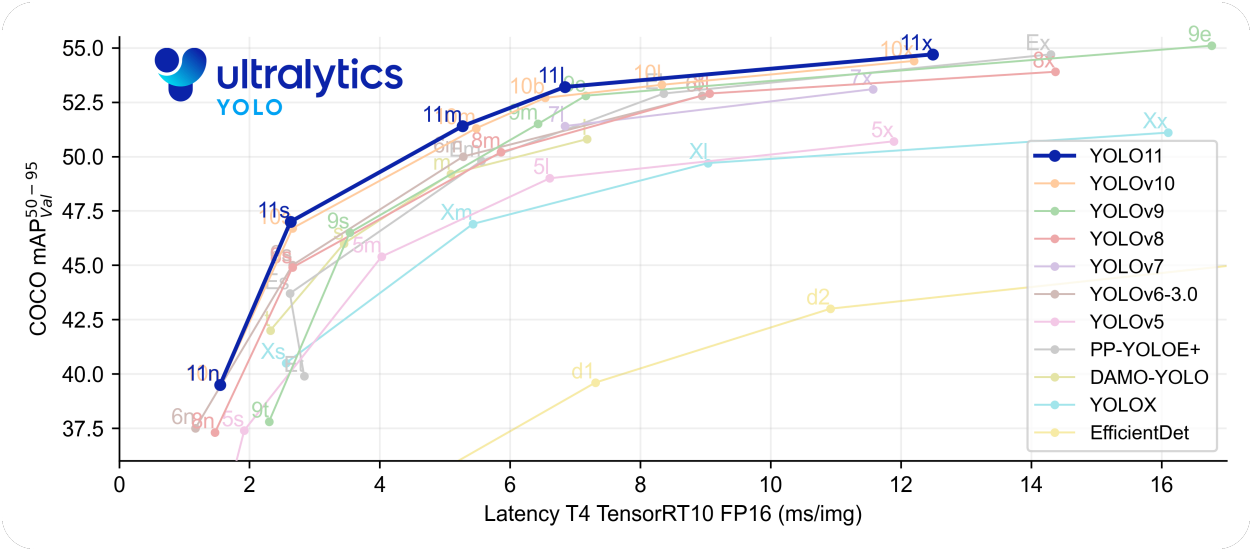


Figure 13: YOLO performance comparison

A.1 Appendix 1 - Our Model Yaml Configuration

```
    # Ultralytics YOLO, AGPL-3.0 license
# YOLOv8 object detection model with P3-P5 outputs. For Usage examples see https://docs.ul

# Parameters
nc: 1  # number of classes
scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml' will call yolov8.yam
  # [depth, width, max_channels]
  n: [0.33, 0.25, 1024]  # YOLOv8n summary: 225 layers,  3157200 parameters,
3157184 gradients,   8.9 GFLOPs
  s: [0.33, 0.50, 1024]  # YOLOv8s summary: 225 layers, 11166560 parameters, 11166544 grad
28.8 GFLOPs
  m: [0.67, 0.75, 768]   # YOLOv8m summary: 295 layers, 25902640 parameters, 25902624 grad
79.3 GFLOPs
  l: [1.00, 1.00, 512]   # YOLOv8l summary: 365 layers, 43691520 parameters, 43691504 grad
  x: [1.00, 1.25, 512]   # YOLOv8x summary: 365 layers, 68229648 parameters, 68229632 grad

# YOLOv8.0n backbone
backbone:
   # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]]  # 0-P1/2
  - [-1, 1, GAM_Attention, []]
  - [-1, 1, Conv, [128, 3, 2]]  # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]]  # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, Conv, [512, 3, 2]]  # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 3, GAM_Attention, []]
  - [-1, 1, SPPFCSPC, [512, 5]]  # 8

# YOLOv8.0n head
head:
  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 4], 1, Concat, [1]]  # cat backbone P3
  - [-1, 3, C2f, [256]]  # 11

  - [-1, 1, nn.Upsample, [None, 2, 'nearest']]
  - [[-1, 2], 1, Concat, [1]]  # cat backbone P2
  - [-1, 3, C2f, [128]]  # 14 (P2/4-xsmall)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 11], 1, Concat, [1]]  # cat head P3
  - [-1, 3, C2f, [256]]  # 17 (P3/8-small)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 8], 1, Concat, [1]]  # cat head P4
  - [-1, 3, C2f, [512]]  # 20 (P4/16-medium)

  - [[14, 17, 20], 1, Detect, [nc]]  # Detect(P2, P3, P4)
```

B Appendix 2 - Training Strategy and Hyperparameters

```
model=model_cfg,
data='/workspace/mydata/YOLOv8_TT100K.yaml',
epochs=200,
batch=batch_size,
imgsz=640,
save=True,
save_period=50,
cache=True,
device=[0,1,2,3],
workers=16,
project=f'runs/{model_name}',
name='train_optimized',
exist_ok=False,
pretrained="/workspace/yolov8-tricks/yolov8s.pt",
optimizer='Adam',
seed=3407,
deterministic=False,
single_cls=False,
rect=False,
cos_lr=True,
close_mosaic=10,
amp=True,
fraction=1.0,
freeze=None,
lr0=0.001 * (batch_size/ 64),
lrf=0.01,
momentum=0.937,
weight_decay=0.0005,
warmup_epochs=5.0,
warmup_momentum=0.8,
warmup_bias_lr=0.1,
box=7.5,
cls=0.5,
dfl=1.5,
plots=True,

# Data augmentation settings
hsv_h=0.015,
hsv_s=0.7,
hsv_v=0.4,
degrees=15,
translate=0.1,
scale=0.5,
shear=5.0,
perspective=0.0005,
flipud=0.0,
fliplr=0.0,
mosaic=1.0,
mixup=0.1,
copy_paste=0.3,
```

C Appendix 3 - Referenced Github Repo

1. YOLOv10: THU-MIG/yolov10
2. YOLOv8 & YOLOv11: ultralytics
3. YOLOv9: WongKinYiu/yolov9
4. Training Framework: chaizwj/yolov8-tricks
5. MobileOne: apple/ml-mobileone
6. MobileSAM: ChaoningZhang/MobileSAM
7. RepViT-SAM: THU-MIG/RepViT
8. GHOST Face Swap: ai-forever/ghost
9. Huawei GhostNet: huawei-noah/Efficient-AI-Backbones
10. MobileViT: yangyucheng000/MobileViT
11. SAHI: obss/sahi
12. External Attention: External-Attention-pytorch

D Appendix 4 - Dataset Usage Disclaimer

1. TT100K official website: https://cg.cs.tsinghua.edu.cn/traffic-sign/
2. TT100K for YOLOv8: https://www.kaggle.com/datasets/braunge/tt100k

The TT100K dataset used in this research is used solely for academic and research purposes. Specifically:

1. This dataset is used exclusively for non-commercial academic research.
2. No commercial use or redistribution of the dataset is permitted.
3. All rights to the original dataset remain with Tsinghua-Tencent Joint Laboratory.
4. Any publication using this dataset must cite the original dataset paper.
5. The authors make no warranties regarding the dataset and accept no liability for its use.

For questions about dataset usage rights, please contact the original dataset authors at: https://cg.cs.tsinghua.edu.cn/traffic-sign/

Contact

GitHub Link

The GitHub repo for this project is https://github.com/GrayMiao123/NYU-CV-Fall2024-Project–Group10.