TADQ: Efficient Edge Deployment of Tiny Attention-Enhanced Quantization Model for Real-time Object Detection in Autonomous Systems

B.Tech Project Report Submitted to IIT Tirupati

submitted in partial fulfillment of the requirements

for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINNEERING

by

G.HARSHITHA CS21B021

K.SAI VIGNESH CS21B028

Supervisor(s)

Dr.C.VISHNU

DEPARTMENT OF COMPUTER SCIENCE AND ENGINNEERING

INDIAN INSTITUTE OF TECHNOLOGY TIRUPATI

May 2025

DECLARATION

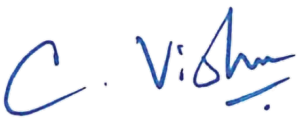
We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission to the best of my knowledge. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 12-05-2025

Signature
G.Harshitha (CS21B021)
K.Sai Vignesh (CS21B028)

BONA FIDE CERTIFICATE

This is to certify that the report titled **TADQ: Efficient Edge Deployment of Tiny Attention-Enhanced Quantization Model for Real-time Object Detection in Autonomous Systems**, submitted by **G.Harshitha and K.Sai Vignesh**, to the Indian Institute of Technology, Tirupati, for the award of the degree of **Bachelor of Technology**, is a bonafide record of the project work done by him [or her] under our or [my] supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.



Place: Tirupati
Date: 12-05-2025

Dr.C.Vishnu
Guide
Assistant Professor
Department of Computer Science and Engineering
IIT Tirupati - 517619

ACKNOWLEDGMENTS

We express our sincere gratitude to our supervisor, Dr. C. Vishnu, for his invaluable guidance, mentorship, and constant encouragement throughout this project. His insights and expertise have been instrumental in shaping the direction and outcome of our work.

We also extend our heartfelt thanks to the Department of Computer Science and Engineering, Indian Institute of Technology Tirupati, for providing a supportive academic environment and the necessary resources to carry out this research.

Finally, we are deeply appreciative of the contributions of researchers whose works we have referenced in our study. Their research and knowledge have greatly enriched our understanding and strengthened the foundation of this project.

ABSTRACT

KEYWORDS: Quantization, PTQ , PyTorch , Attention , Tiny-Attention-Detection-Quantization, Object-Detection , YoloV8, Tensorflow, CBAM, ECA, CARLA

Recent advancements in autonomous driving systems demand robust object detection models that can operate efficiently on edge devices without compromising accuracy. While attention mechanisms have significantly improved detection performance, their integration with quantized models for edge deployment remains largely underexplored. In this paper, we propose Tiny Attention Detection Quuantization, a novel framework that systematically integrates lightweight attention modules - CBAM and ECA into the YOLOv8 architecture.Our pipeline, TADQ, combines these attention modules with post-training static quantization using TensorFlow Lite to optimize for both FP16 (16-bit) and INT8 (8-bit) inference. We evaluate the performance on the CARLA autonomous driving dataset, demonstrating that our TADQ approach effectively preserves the benefits of attention mechanisms even under aggressive quantization while maintaining real-time capabilities, highlighting its suitability for real-world deployment in resource-constrained environments.

- **Baseline YOLOv8** achieves a mean Average Precision (mAP) of **69.0%**, serving as the performance benchmark.
- **YOLOv8 + ECA attention module** improves detection accuracy to **75.0% mAP**, showing enhanced feature sensitivity with minimal overhead.
- **YOLOv8 + ResCBAM (TADQ)** further boosts performance to **84.2% mAP**, significantly outperforming the baseline and other variants.
- **Quantized TADQ model (INT8)** retains high accuracy with **80.1% mAP**, demonstrating robustness against quantization-induced degradation.
- **Model size is reduced by approximately 50%** post quantization, enabling memory-efficient deployment on edge devices.

TADQ offers a compelling balance between detection accuracy and computational efficiency. The integration of attention mechanisms with quantization enables efficient deployment of object detectors in real-world autonomous driving applications.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CUDA	Compute Unified Device Architecture
FPS	Frames Per Second
GPU	Graphics Processing Unit
mAP	Mean Average Precision
mAR	Mean Average Recall
NLP	Natural Language Processing
PTQ	Post-Training Quantization
YOLO	You Only Look Once
CBAM	Convolutional Block Attention Module
ECA	Efficient Channel Attention

CHAPTER 1

INTRODUCTION

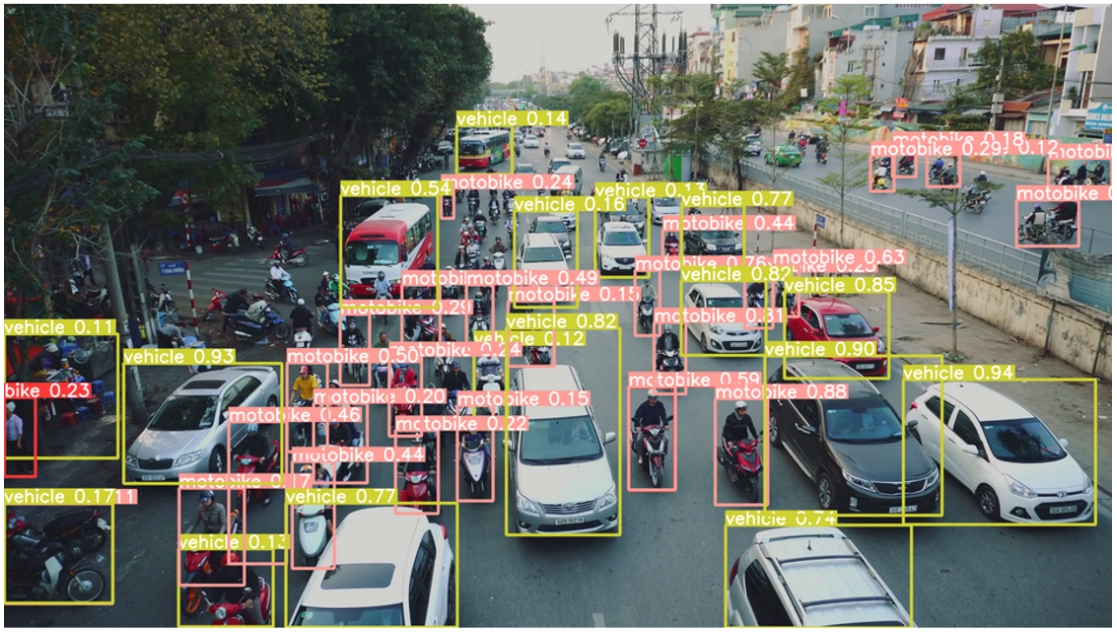


Figure 1.1: Object detection in traffic scenarios from the results of BTP1

Object detection is a fundamental task in computer vision, with applications ranging from autonomous driving and surveillance to robotics and healthcare. Modern object detection models like YOLOv8 have achieved state-of-the-art accuracy. However, their computational complexity and resource-intensive nature make them less suitable for deployment on edge devices such as drones, IoT devices, and mobile platforms.

Edge devices typically have limited computational power, constrained memory, and stringent real-time processing requirements. These constraints, added to the intensive power requirement of sophisticated models, represent a great challenge to real-time object detection. Guaranteeing both low latency and great accuracy under these restrictions calls for novel ways of optimizing.

Emerging as a bright answer to these problems is model quantization. Quantization dramatically lowers. The model size and computational overhead allow for quicker inference and lowered Energy use: Though these benefits exist, perfect performance without Still a major topic of investigation is cutting back on detection accuracy.

TADQ, a fresh approach incorporating light attention methods inside The YOLOv8 design's middle layers. The strategy's approach embeds neck network attention modules stress important characteristics while depressing others. The lesser important ones. This improvement lets the quantized model keep precision. Furthermore used is hybrid quantization approach whereby one transitions from FP32 to FP16 and INT8 to maximize computational efficiency while preserving performance much.

The CARLA autonomous driving dataset is thoroughly tested in several experiments. showing that the suggested approach saves model size by 50% while keeping excellent accuracy relative to conventional quantization methods. Insights are given regarding how post-training quantization performance depends on attention mechanisms. enabling the effective implementation of object recognition systems in actual independent systems.

CHAPTER 2

Literature Review

2.1 Object Detection

Object detection, a fundamental task in computer vision, has gained significant attention. Given its applications in several areas including surveillance systems, autonomous cars, robotics then. Localizing relates to not only object detection in an image but also them, therefore, calls for strong models able to manage many different real-world situations. Object detection has developed over time from early approaches like sliding window. classifiers for state-of-the-art architectures like YOLO (You Only Look Once) (4), SSD (Single Shot Multibox Detector), and Faster R-CNN. kim et al. (11) and Deepa et al. (12) offer a thorough comparative study on these systems, especially By balancing speed and accuracy, YOLO has shown remarkable results. Making them appropriate for applications in real time.

Zhou et al. (15) suggest a way to improve object recognition accuracy by changing asymmetrical pooling levels, the YOLOv5 architecture's receptive field can be enlarged. to the head section of YOLOv5. Zhong et al. (16) presents an optimized YOLOv8 model with reduced parameters and improved accuracy using techniques such as Faster-C2f, Rep-Fasterblock, and advanced neck networks. Niranjan et al. (6) introduce an object detection model for autonomous driving and simulate it with CARLA simulator.

2.2 Autonomous Systems

Recent advancements in object detection algorithms have revolutionized various domains, including autonomous vehicles, surveillance systems, and industrial automation. Despite progress, several challenges remain unaddressed, particularly in balancing computational efficiency, accuracy, and energy consumption in resource-constrained environments like edge computing and UAV-based systems.

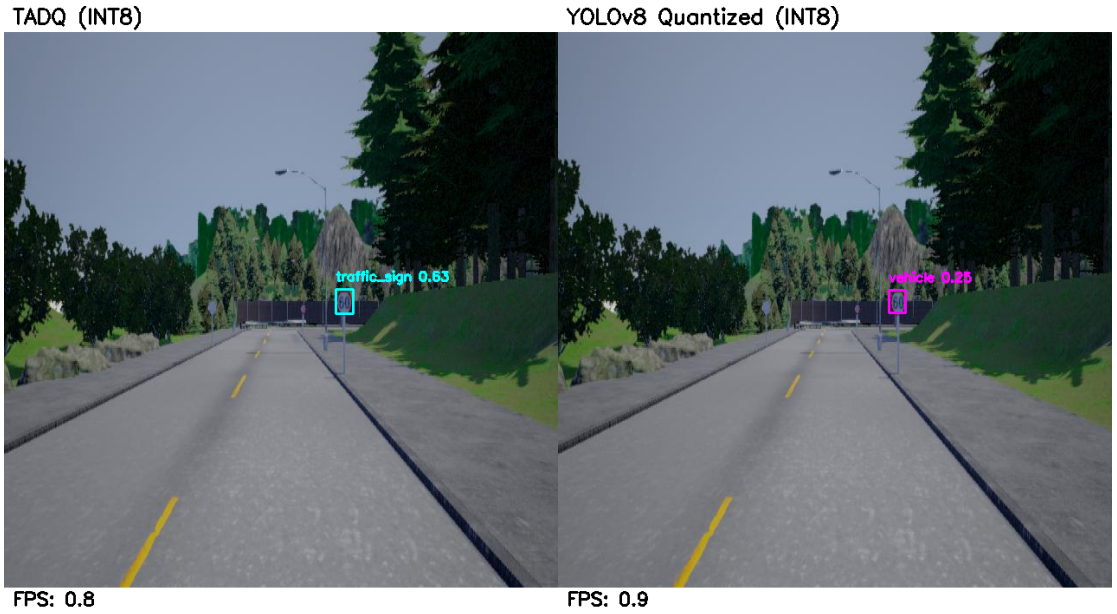


Figure 2.1: YOLOv8 Quantized model misclassifies a traffic sign as a vehicle.

In autonomous systems, the integration of object detection plays a pivotal role. Autonomous systems, whether they are self-driving cars or drones, are heavily dependent on real-time object recognition and tracking for navigation, collision avoidance, and decision making, where a single mistake can cost us a huge loss or damage, so every percent of accuracy matters in these critical systems. Yang et al. (14) propose that the integration of quantization-aware training (QAT) and reparameterization optimizers (RepOpt) has resulted in significant reductions in energy consumption, allowing extended UAV flight times while maintaining performance.

Zhou et al. (17) overcome the challenge of detecting objects at multiple scales in autonomous driving scenarios by introducing an efficient multiscale detection framework, which effectively handles small object detection while maintaining real-time performance. Maaz et al. (3) propose a mixed architecture of CNNs and Transformers along with a split depth-wise transpose attention (STDA) encoder using self-attention across channel dimensions which showcases better accuracy even with fewer parameters, for efficient deployment on edge devices and mobile platforms.

2.3 Attention Mechanism

The attention mechanism has increasingly become a very situational deep learning development, especially for computer vision and natural language processing. Derived from the cognitive functioning of humans, attention lets neural networks fix emphasis on vital parts of input data, thus improving performance and interpretability. In relation to object detection, attention mechanisms assume major roles by focusing on specific salient regions or features, thus aiding in increased precision or efficiency for various detection tasks.

2.3.1 Working Principle of Attention

The crux of attention is to give a weighted sum of input features where the weight describes the importance of the said feature within the context. More formally, the attention mechanism can be described as a query-key-value (QKV) model:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where Q (query), K (key), and V (value) are linear projections of the input features, and d_k is the dimension of the key vectors. The softmax function computes attention weights that determine the relevance of each feature.

2.3.2 Channel Attention

Channel attention mechanisms focus on learning the interdependencies between different feature channels. The main idea is that not all feature maps are equally informative; some channels may contribute more to the task than others. One widely adopted method is the Squeeze-and-Excitation (SE) block (21), which computes channel-wise attention by first performing global average pooling to squeeze spatial information into a channel descriptor and then using fully connected layers to learn channel-wise dependencies.

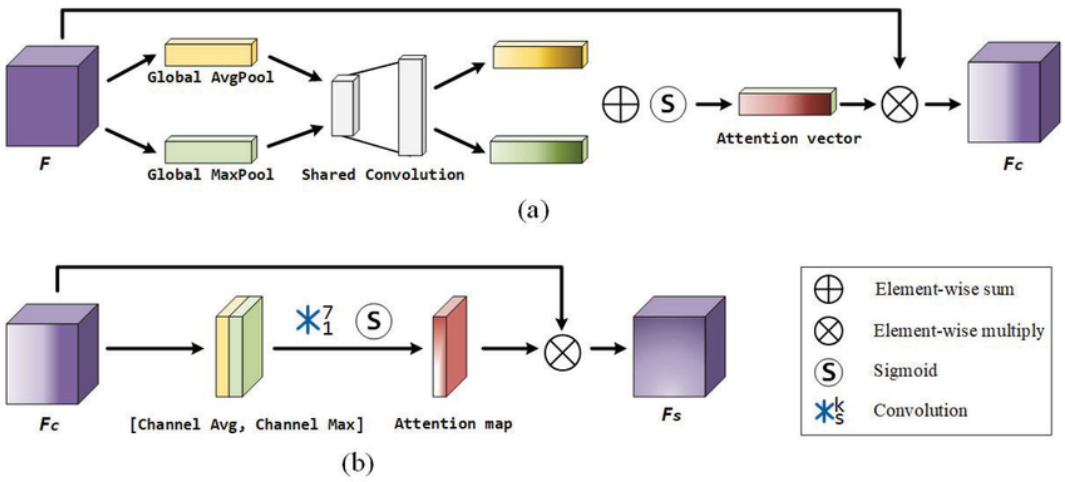


Figure 2.2: The channel attention mechanism and spatial attention mechanism. (a) and (b) respectively denote the channel attention module and spatial attention module.

2.3.3 Spatial Attention

While channel attention emphasizes the importance of feature maps, spatial attention focuses on identifying the most informative spatial locations within a feature map. This is particularly useful for tasks like object detection, where the location of the object is crucial. The Convolutional Block Attention Module (CBAM) (1) combines both channel and spatial attention sequentially to refine feature representations.

Types of Attention Mechanisms

Several types of attention mechanisms have been proposed, each with different characteristics and computational implications:

- **Self-Attention:** A mechanism where a sequence attends to itself, widely used in transformers and Vision Transformers (ViTs).
- **Global Attention:** Considers all positions in the input when computing attention, ensuring comprehensive context understanding.
- **Local Attention:** Focuses on a window of nearby features, reducing computational cost while capturing local dependencies..

Relevance to Object Detection

In object detection projects, adding attention mechanisms improves the accuracy of both the precision and strength of feature extraction. Focus on discriminative is aided by channel attention. Features like texture or color; spatial attention guarantees the model's

precise object placement.

By Combining spatial and channel attention, these systems let lightweight Still, significant improvements ideal for implementation on restricted resources. particularly when followed by quantization techniques for optimization, edge devices.

Integrating attentional systems into YOLO-based object recognition algorithms has become a strong strategy to improve detection accuracy. Model focus on pertinent characteristics is made possible by attention mechanisms, which also eliminate noise, therefore making them very good in complicated settings.

In recent times, Efficient Channel Attention (ECA) module (8) and the Convolution Block Attention Module (CBAM) (1) demonstrated to offer great potential in improving the performance of deep convolutional neural networks (CNNs) using channel and spatial attention. Introducing these modules in object detection models like YOLO (9), (19), (20) shown great results. Cai et al. (18) build upon YOLOX by introducing attention mechanism in path aggregation feature pyramid network (PAFPN) to make the network focus more on important information.

2.4 Diffusion

Diffusion models, originally developed for generative tasks, operate by progressively denoising data from a random noise distribution to a structured output. This iterative denoising helps the model embrace the finer nuances of a given data distribution, making it apt for any task that requires a finer resolution to understand its target, for instance: object detection.

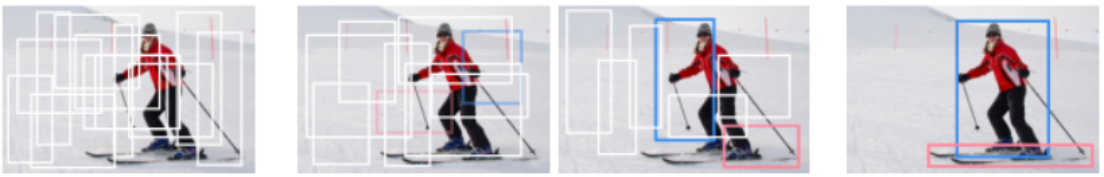


Figure 2.3: Object detection as a denoising diffusion process from noisy boxes to object boxes.

Generating object proposals using diffusion models in order to give a better representation to the features acts as an intermediate step toward synthetic data training. By assuming the process of detection as a special denoising phenomenon, these models can

refine bounding box predictions and classification scores iteratively.

Relavance to Object Detection

Diffusion blocks integrated into object detection architectures around attention modules provide more benefits:

Iterative Refinement: Diffusion processes enable the model to iteratively refine object proposals, leading to more accurate detections.

Robust Feature Representation: The denoising nature of diffusion models helps in learning robust features that are less sensitive to noise and variations in the input data.

Improved Generalization: By simulating various noise conditions during training, diffusion models can enhance the model's ability to generalize to unseen scenarios.

Recent studies have explored the integration of diffusion models into object detection frameworks. DiffusionDet: This approach formulates object detection as a denoising diffusion process, refining randomly generated boxes to accurate object detections through iterative steps (22). DiffusionEngine: A scalable data generation engine that utilizes diffusion models to produce high-quality, diverse training data for object detection tasks, improving performance in data-scarce scenarios (23)

2.5 Quantization Techniques

Quantization has become an integral technique in optimizing deep learning models for deployment on resource-constrained devices. It involves reducing the precision of the numerical representation of model parameters and operations, transitioning from high-precision formats (e.g., 32-bit floating point) to lower-precision formats (e.g., 8-bit integers). This method lowers memory consumption considerably, computation expense, and energy usage while holding tolerable standards of precision.

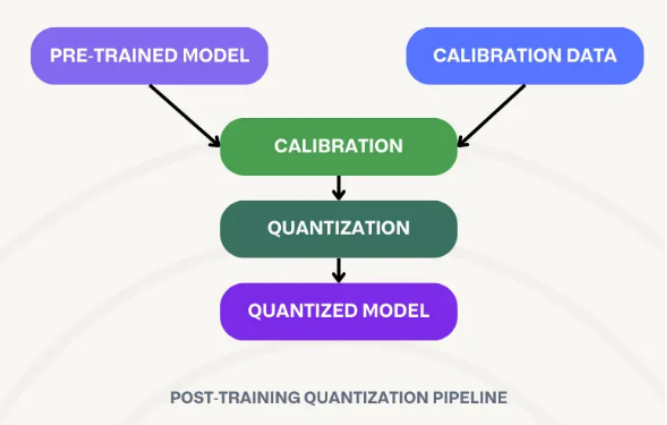


Figure 2.4: Architecture of Post-training quantization

2.5.1 Post-Training Quantization (PTQ)

Post-Training Quantization is a technique applied after the model has been trained. Because it doesn't require extra retraining or model fine-tuning, it's a desirable choice in situations where retraining resources are scarce. PTQ involves:

- **Weight Quantization:** Reducing the precision of model weights.
- **Activation Quantization:** Quantizing activations during inference.

Even though PTQ is computationally effective, model performance may slightly deteriorate as a result, especially for tasks requiring high numerical precision.

2.5.2 Static and Dynamic Quantization

Quantization can be implemented statically or dynamically, depending on the nature of the workload and operational constraints.

Static Quantization

- In static quantization, the quantization parameters (scale and zero-point) are determined during the calibration phase, before inference.
- Calibration involves running a subset of the dataset through the model to collect statistics.
- Static quantization is particularly useful for models deployed in environments with predictable workloads.

Dynamic Quantization

- Dynamic quantization computes the quantization parameters at runtime, based on the input data.
- It is typically used for NLP and RNN-based models where inputs vary significantly during inference.
- This approach incurs additional computational overhead but adapts well to diverse inputs.

‘ Post-training quantization (PTQ) is particularly advantageous as it enables model compression without requiring retraining, making it highly practical for real-world deployment and also in cases where data cant be accessed. Recent advancements in PTQ for object detection have focused on preserving feature integrity and reducing accuracy degradation. Q-YOLO introduced by Wang et al. (13) utilizes a unilateral histogram based activation quantization scheme, in the PTQ pipeline minimizing the mean squared error quantization errors. MPQ-YOLO by Liu et al. (5) propose mixed precision quantization for YOLO which retains better accuracy in low bit representations.

Xie et al. (10) propose a mix of pruning and an improved Parameterized Clipping Activation(PACT) algorithm to quantize pruned model resulting in an great compression of model and improved frames per second(FPS). Huang et al. introduced HQOD, a method that balances classification and localization loss during quantization-aware training (QAT), achieving near-floating-point precision (7).

2.6 YOLO-v8

YOLOv8, represents a significant leap forward in the realm of real-time object detection. Building upon the strong foundation laid by its predecessors, YOLOv8 incorporates several advanced features that address the limitations of earlier models. This makes it an optimal choice for uses where precise object detection is essential, including autonomous systems, which need great speed and accuracy. In our research, The anchor-free split Ultralytics head, which eliminates reliance on predefined anchor boxes and streamlines the detection process, is one of the key innovations in YOLOv8. This design lowers computational overhead in addition to enhancing detection accuracy, particularly for objects of varying scales. The model’s state-of-the-art backbone and neck architectures further improve feature extraction, ensuring robust performance even in challenging

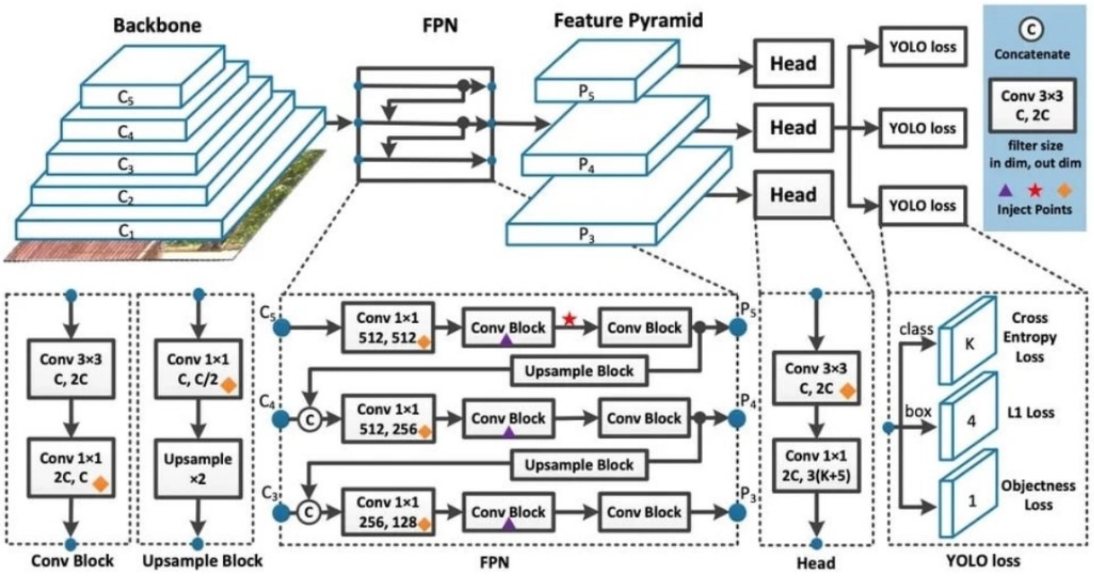


Figure 2.5: Architecture of YOLOv8

scenarios. For example, the model achieves a remarkable mean Average Precision (mAP) of up to 53.9 on the COCO dataset with YOLOv8x, showcasing its ability to outperform previous YOLO versions.

From a mathematical perspective, YOLOv8 optimizes the object detection process through its loss function, which balances localization (L_{loc}), confidence (L_{conf}), and classification losses (L_{cls}):

$$L_{\text{total}} = \lambda_{\text{loc}} \cdot L_{\text{loc}} + \lambda_{\text{conf}} \cdot L_{\text{conf}} + \lambda_{\text{cls}} \cdot L_{\text{cls}} \tag{2.2}$$

Here, $\lambda_{\text{loc}}, \lambda_{\text{conf}},$ and λ_{cls} are weight factors that allow fine-tuning for specific tasks, ensuring adaptability across diverse datasets.

Additionally, YOLOv8 integrates the Information Bottleneck Principle, maximizing mutual information across the layers to retain essential features. This is mathematically represented as:

$$I(X,X) \geq I(X, f_\theta(X)) \geq I(X, g_\phi(f_\theta(X))) \tag{2.3}$$

where f_θ and g_ϕ are parameterized transformation functions for feature encoding and decoding, respectively.

The model also incorporates reversible functions to minimize information loss:

$$X = v_{\zeta}(r_{\psi}(X)) \tag{2.4}$$

where r_{ψ} and v_{ζ} represent the reversible and inverse functions, respectively. This architectural innovation enhances parameter utilization and facilitates efficient gradient updates.

To sum up, YOLOv8's flexible across several computer vision tasks make it a compelling option for our study. Using YOLOv8, we have been able to strike a compromise between accuracy and efficiency, which is essential for the success of our application. With respect to YOLOv8, recent research has concentrated on enhancing accuracy with mechanisms of attentiveness. This method is in line with our research on Tiny Attention Detection.

For YOLOv8, quantization is the process by which we seek to balance accuracy and efficiency by using lightweight attention modules with quantization methods. Through a methodical assessment of quantized variations, we investigate the trade-offs between precision and computational efficiency, so helping to optimize object identification systems for edge. Systems: Reducing model parameters and accelerating inference speed while keeping accuracy, our approach solves the problems of real-time object recognition in Resource-constrained surroundings for autonomous critical decision-making systems. cars and drones.

These developments highlight the need of attention mechanisms and quantization in improving YOLO-based models. Our suggested approach builds on these already in existence. Offering a simplified solution for real-time detection operations while preserving great precision, foundations

CHAPTER 3

Implementation

3.1 Proposed Architecture

The proposed pipeline, TADQ (Tiny Attention-Enhanced Quantization), introduces a novel framework for improving real-time object detection in autonomous systems by incorporating attention mechanisms into quantized models. The pipeline is structured as follows:The proposed pipeline, TADQ (Tiny Attention-Enhanced Quantization), introduces a novel framework for improving real-time object detection in autonomous systems by incorporating attention mechanisms into quantized models. The pipeline is structured as follows:

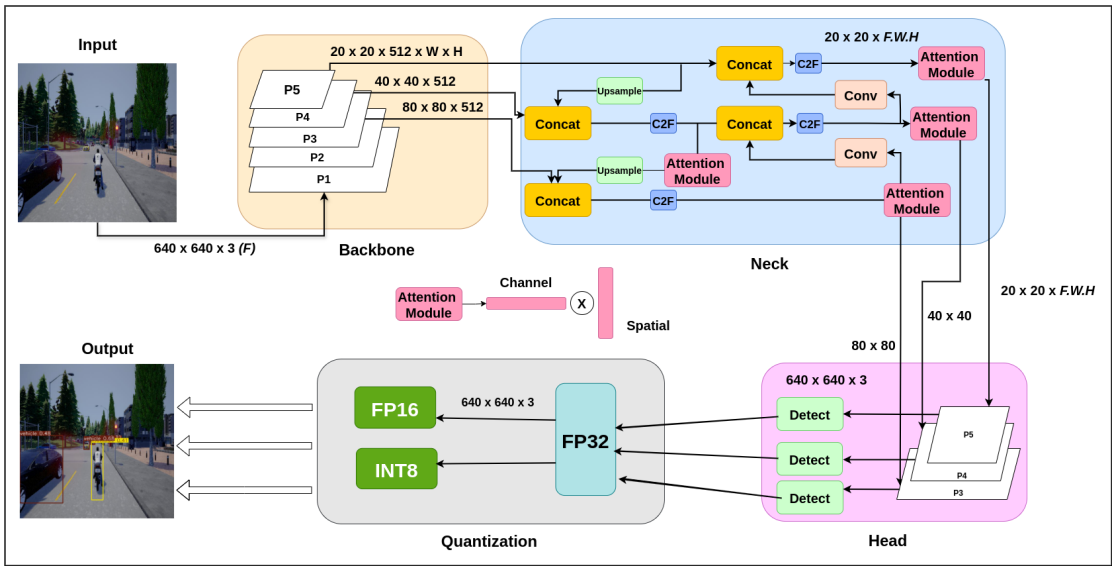


Figure 3.1: Proposed Architecture of the TADQ Pipeline

The pipeline begins with the selection of YOLOv8 as the baseline model, chosen for its state-of-the-art performance in object detection and real-time capabilities. Tiny attention modules are integrated into the intermediate layers, particularly in the neck of the architecture. These attention units help the model to give important characteristics top priority and reject unimportant ones, therefore improving detection accuracy without greatly raising the computational complexity of the model. In two different configurations, the improved model undergoes precise reduction: conversion to FP16 maintains

floating-point accuracy for quick calculations while lowering memory footprint. Using integer precision, conversion to INT8 further compresses the model by greatly lowering computing burden and energy use.

3.2 Methodology and Challenges

3.2.1 Attention modules

Attention mechanisms are primarily categorized into:

Spatial Attention: Captures pairwise pixel-level relationships, emphasizing the spatial locations critical for detection tasks.

Channel Attention: Focuses on dependencies across feature channels, allowing the model to recalibrate channel-wise feature importance.

Several studies have demonstrated the effectiveness of attention mechanisms in improving model performance. For instance, the Convolutional Block Attention Module (CBAM) (1) combines spatial and channel attention to enhance the representation capabilities of convolutional networks. The Efficient Channel Attention (ECA) module (8), on the other hand, achieves competitive performance with fewer parameters by leveraging local cross-channel interactions.

Building on these principles, we introduce an attention-enhanced YOLOv8 architecture tailored for autonomous object detection tasks.

3.2.2 Proposed Integration of Attention Mechanisms

We integrate the following attention modules into the YOLOv8 architecture to evaluate their impact on model performance:

CBAM: Combines spatial and channel attention to improve feature refinement at every convolutional block. The module processes features F sequentially through channel and spatial attention:

$$F' = M_c(F) \otimes F \tag{3.1}$$

$$F'' = M_s(F') \otimes F' \tag{3.2}$$

where M_c and M_s are defined as:

$$M_c(F) = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \tag{3.3}$$

$$M_s(F) = \sigma(f^{7\times7}([AvgPool(F'); MaxPool(F')])) \tag{3.4}$$

Where: $F \in \mathbb{R}^{C\times H\times W}$ is input feature map $M_c \in \mathbb{R}^{C\times1\times1}$ is channel attention map $M_s \in \mathbb{R}^{1\times H\times W}$ is spatial attention map

ECA: Simplifies channel attention by eliminating fully connected layers, reducing computational overhead while retaining performance. The ECA module applies efficient channel attention through:

$$y_c = \sigma(\sum_{k=1}^{K} w_k \cdot \sum_{i=1}^{H}\sum_{j=1}^{W} x_{c+k-\lceil\frac{K}{2}\rceil,i,j}) \tag{3.5}$$

with adaptive kernel size selection:

$$k_{size} = \left|\frac{\log_2(C)}{\gamma} + \frac{b}{\gamma}\right|_{odd} \tag{3.6}$$

Where: $x_{c,i,j}$ is the pixel value at position (i, j) in channel c w_k are the convolution kernel weights K is the kernel size for local cross-channel interaction C is the number of channels γ and b are hyperparameters $||_{odd}$ ensures the kernel size is odd

To further optimize the architecture, we use hybrid modules such as ResBlock + CBAM (ResCBAM), which combine the feature extraction capabilities of ResBlocks with attention modules to improve performance. The ResCBAM operation can be expressed as:

$$Y = F(X, W_i) + X \tag{3.7}$$

$$Output = CBAM(Y) \tag{3.8}$$

Where: X is the input feature map $F(X, W_i)$ represents residual mapping W_i are learnable parameters $CBAM()$ represents the CBAM attention operation.

Figure 3.2 shows the original scene captured from the CARLA simulator, while Figure 3.3 displays attention heatmaps across multiple channels in our ResCBAM-based model. These visualizations demonstrate that different channels selectively focus on important spatial regions such as vehicles, roads, and contextual background elements, validating the effectiveness of attention in guiding the model's focus during detection.

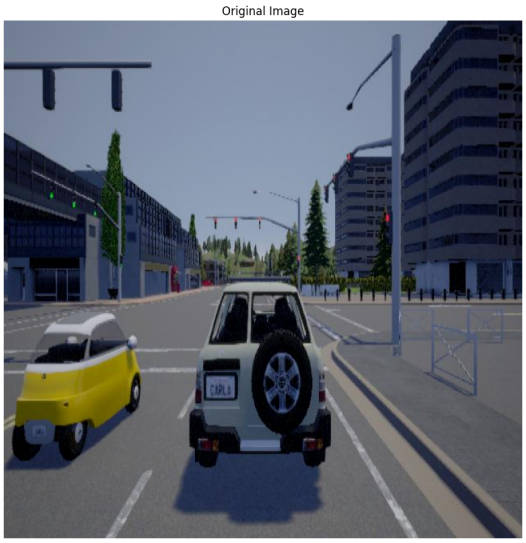


Figure 3.2: Original input image from the CARLA simulator.

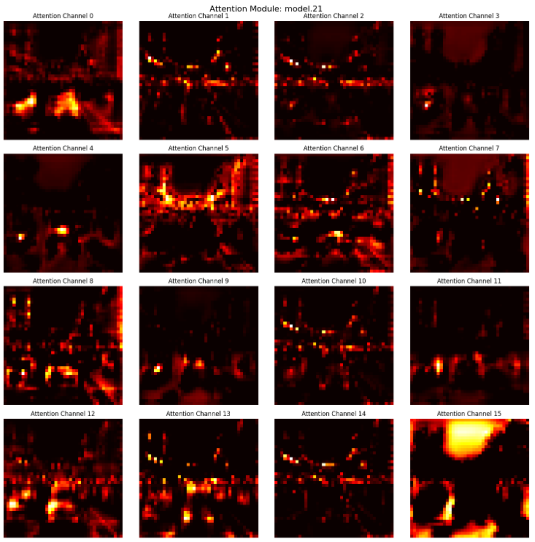


Figure 3.3: Grad-CAM visualizations across 16 attention channels from the attention module.

3.2.3 Attention modules in PyTorch

ECA Attention Module

```python
class ECAAttention(nn.Module):
    def __init__(self, c1, k_size=3):
        super(ECAAttention, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.conv = nn.Conv1d(1, 1, kernel_size=k_size,
                              padding=(k_size - 1) // 2, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        y = self.avg_pool(x)
        y = self.conv(y.squeeze(-1).transpose(-1, -2)) \
            .transpose(-1, -2).unsqueeze(-1)
        y = self.sigmoid(y)
        return x * y.expand_as(x)
```

Figure 3.4: ECAAttention Module in PyTorch

The `ECAAttention` module implements the Efficient Channel Attention mechanism. It first performs global average pooling to summarize spatial information into a channel descriptor. This descriptor is then passed through a 1D convolutional layer that captures local cross-channel dependencies without the need for dimensionality reduction. A sigmoid activation function generates attention weights, which are then multiplied element-wise with the input tensor. This process selectively enhances important channels while suppressing less informative ones.

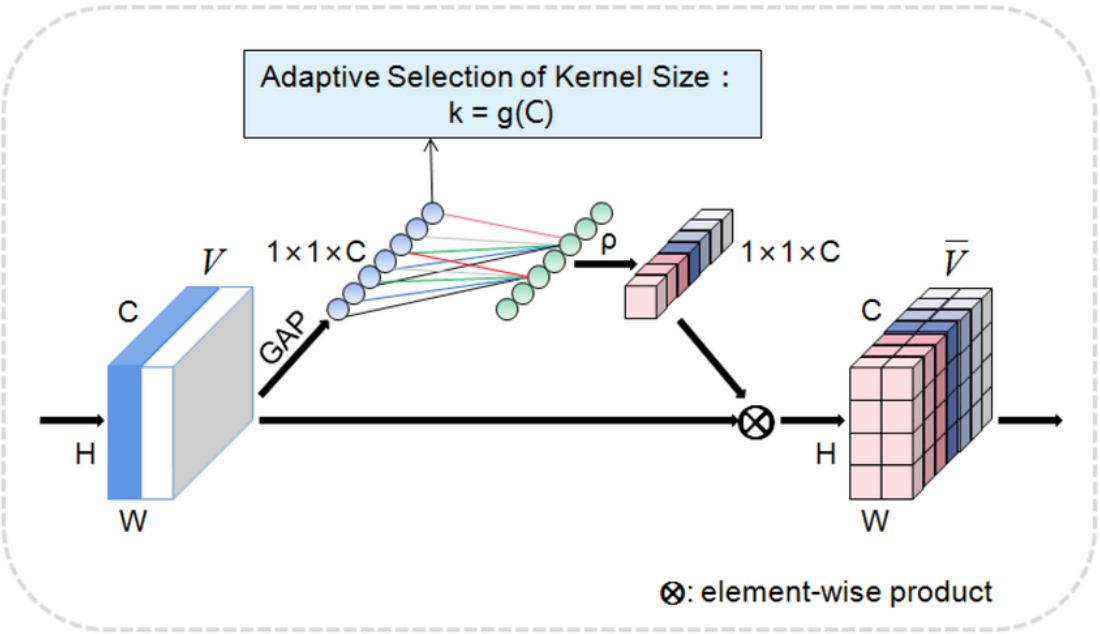


Figure 3.5: ECA Module

ResCBAM Attention Module

The `ResBlock_CBAM` module defines a residual block that integrates the Convolutional Block Attention Module (CBAM) to enhance feature representations. The input first undergoes a bottleneck transformation comprising three convolutional layers with batch normalization and LeakyReLU activation. The output of this transformation is passed through the CBAM, which sequentially applies channel and spatial attention mechanisms to recalibrate the features. If the spatial resolution or channel dimension changes, the residual path is downsampled to match the output. Finally, the recalibrated output is added to the residual connection, and a ReLU activation is applied to introduce non-linearity.

3.2.4 Lightweight Diffusion Module Integration

To investigate further enhancement of feature representation, we experimented with integrating a lightweight *DiffusionBlock* in the attention path. The idea was inspired by the concept of feature denoising and robustness introduced through controlled stochasticity, mimicking aspects of diffusion processes. We hypothesized that injecting small learnable noise and applying a minimal transformation might help the model generalize better by diversifying the learned representations.

The `DiffusionBlock` is a minimal module designed to apply a stochastic perturbation followed by a simple transformation. It introduces a learnable noise component through the `noise_scale` parameter and adds feature-wise Gaussian noise sampled at runtime. This noisy

```python
class ResBlock_CBAM(nn.Module):
    def __init__(self, in_places, places, stride=1, downsampling=False, expansion=1):
        super(ResBlock_CBAM, self).__init__()
        self.expansion = expansion
        self.downsampling = downsampling

        self.bottleneck = nn.Sequential(
            nn.Conv2d(in_places, places, 1, 1, bias=False),
            nn.BatchNorm2d(places),
            nn.LeakyReLU(0.1, inplace=True),
            nn.Conv2d(places, places, 3, stride, 1, bias=False),
            nn.BatchNorm2d(places),
            nn.LeakyReLU(0.1, inplace=True),
            nn.Conv2d(places, places * expansion, 1, 1, bias=False),
            nn.BatchNorm2d(places * expansion),
        )
        self.cbam = CBAM(c1=places * expansion)

        if downsampling:
            self.downsample = nn.Sequential(
                nn.Conv2d(in_places, places * expansion, 1, stride, bias=False),
                nn.BatchNorm2d(places * expansion)
            )
        self.relu = nn.ReLU(inplace=True)

    def forward(self, x):
        residual = x
        out = self.bottleneck(x)
        out = self.cbam(out)
        if self.downsampling:
            residual = self.downsample(x)
        out += residual
        out = self.relu(out)
        return out
```

Figure 3.6: ResBlock CBAM Module in PyTorch

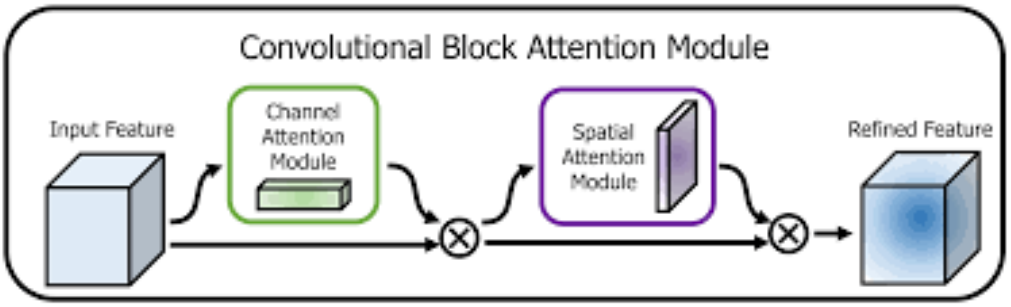


Figure 3.7: CBAM Module

input is then passed through a 3x3 convolutional layer followed by batch normalization and ReLU activation to learn a noise-invariant representation. We tested this module in two configurations: (i) placed before the attention blocks (CBAM and ECA) to perturb the features before weighting, and (ii) after the attention blocks to refine the output representation.

Despite the conceptual motivation, empirical results indicated that the performance did not

```python
class DiffusionBlock(nn.Module):
    def __init__(self, channels):
        super(DiffusionBlock, self).__init__()
        self.noise_scale = nn.Parameter(torch.zeros(1))  # learnable noise magnitude
        self.transform = nn.Sequential(
            nn.Conv2d(channels, channels, kernel_size=3, padding=1, bias=False),
            nn.BatchNorm2d(channels),
            nn.ReLU(inplace=True)
        )

    def forward(self, x):
        noise = torch.randn_like(x) * self.noise_scale
        x_noisy = x + noise
        return self.transform(x_noisy)
```

Figure 3.8: Lightweight DiffusionBlock in PyTorch

improve as expected. In some configurations, accuracy and mAP slightly degraded. This could be due to the shallow and lightweight nature of the diffusion block, which may not be complex enough to introduce meaningful refinements, or due to interference of the injected noise with the precise feature selection enforced by attention mechanisms. While lightweight diffusion-inspired transformations offer a promising direction for future research, our initial integration did not yield measurable improvements. More structured or deeper forms of stochastic attention fusion could be explored in future work.

3.2.5 Model Quantization

In our implementation, we explore Post Training Quantization (PTQ) with static quantization approach to optimize our tiny attention-enhanced YOLOv8 model for edge deployment:

Post Training Quantization

PTQ is applied after model training is complete, converting the model to use lower precision arithmetic for both storage and computation. Within PTQ, we specifically utilize static quantization, which offers a balance between model efficiency and accuracy. This method:

Pre-computes scaling factors using a calibration dataset Converts weights to INT8 format during model conversion Maintains consistent quantization parameters during inference.

Static Quantization

Static quantization involves converting floating-point weights and activations to fixed-point integers during inference. The quantization process can be expressed mathematically as:

$$Q(x) = round(\frac{x}{S}) + Z \tag{3.9}$$

$$S = \frac{x_{max} - x_{min}}{2^n - 1} \tag{3.10}$$

$$Z = round(-\frac{x_{min}}{S}) \tag{3.11}$$

Where: $Q(x)$ is the quantized value, x is the floating-point input, S is the scale factor, Z is the zero point, n is the bit width (8 for INT8), x_{max} and x_{min} are the maximum and minimum values in the calibration dataset. The dequantization process is given by:

$$x = S(q - Z) \tag{3.12}$$

Where: x is the dequantized floating-point value, S is the scale factor,q is the quantized value.

Our suggested methodology strikes a balance between model performance and deployment efficiency by utilising quantisation and attention processes, which makes it appropriate for autonomous object detection tasks in the real world.

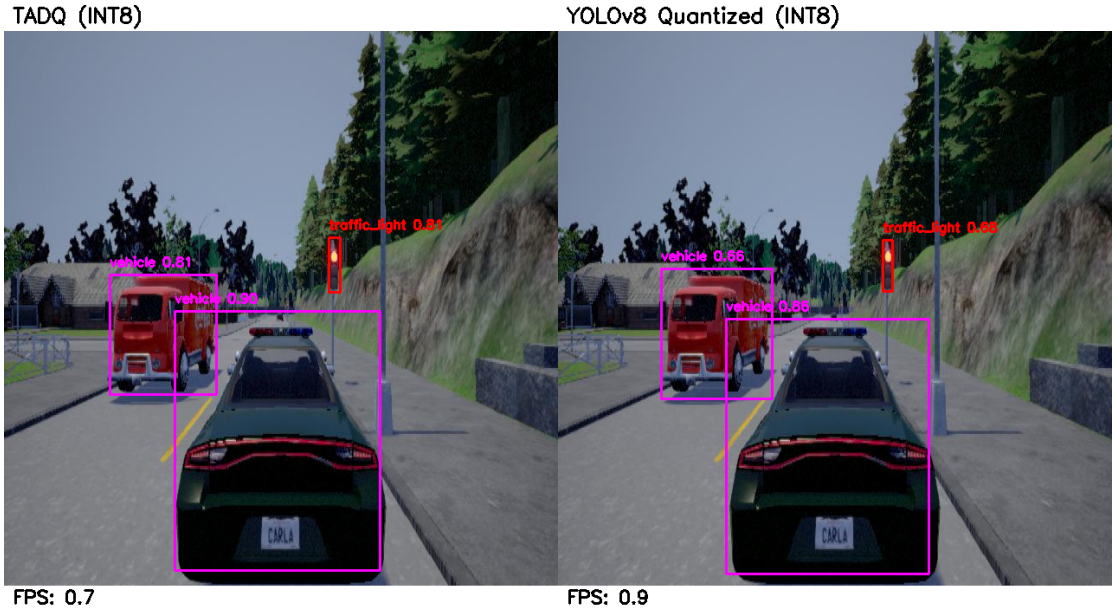


Figure 3.9: TADQ demonstrates higher confidence in detection.

Quantization through TensorFlow

Out of existing deep learning libraries, empirically, we noticed TensorFlow had the best support API for our use case in the pipeline that was proposed, especially finetuning custom layers. After completion of training, the models are exported to TensorFlow and then quantized to TFLite models. Initially, to learn the static quantization weights and activations, we have to calibrate the dataset. So, we used a subset of the CARLA dataset (300 images) to calculate the performance of the trained model for various settings (FP32, FP16, INT8). FP16 quantization lowers memory usage while maintaining near-full precision, providing a good balance between accuracy and efficiency. On the other hand, INT8 quantization further reduces model size and accelerates inference, enabling real-time processing on low-power hardware devices and embedded systems. These optimizations ensure that TFLite models perform efficiently with minimal resource consumption while preserving accuracy.

3.3 Dataset

The CARLA dataset, used for this study, consists of 886 images across five classes: bike , motorbike, vehicle, traffic light and traffic sign, with all images sized at 640×640 pixels. The dataset is divided into 575 training images, 100 validation images, and 211 testing images to ensure robust model evaluation. Designed to replicate realistic traffic scenarios, the CARLA dataset provides a valuable benchmark for developing and testing object detection models in urban environments.

Classes	Training Images	Validation Images	Testing Images	Total Images
Bike	141	4	23	168
Motobike	61	18	11	90
Vehicle	520	67	137	724
Traffic Light	310	56	174	540
Traffic Sign	67	6	10	83

Table 3.1: Distribution of images across different classes for training, validation, and testing.

3.4 Dataset Preprocessing and Augmentation

For dataset preparation, we utilized Roboflow (v1.1.51) for efficient dataset splitting and preprocessing. The dataset was divided into three sets: 65% (575 images) for training, 11% (100 images) for validation, and 24% (211 images) for testing. Preprocessing steps included applying auto-orientation and resizing the images to a fixed resolution of 640x640 using stretching. No additional augmentations were applied during this stage to preserve the original dataset characteristics.

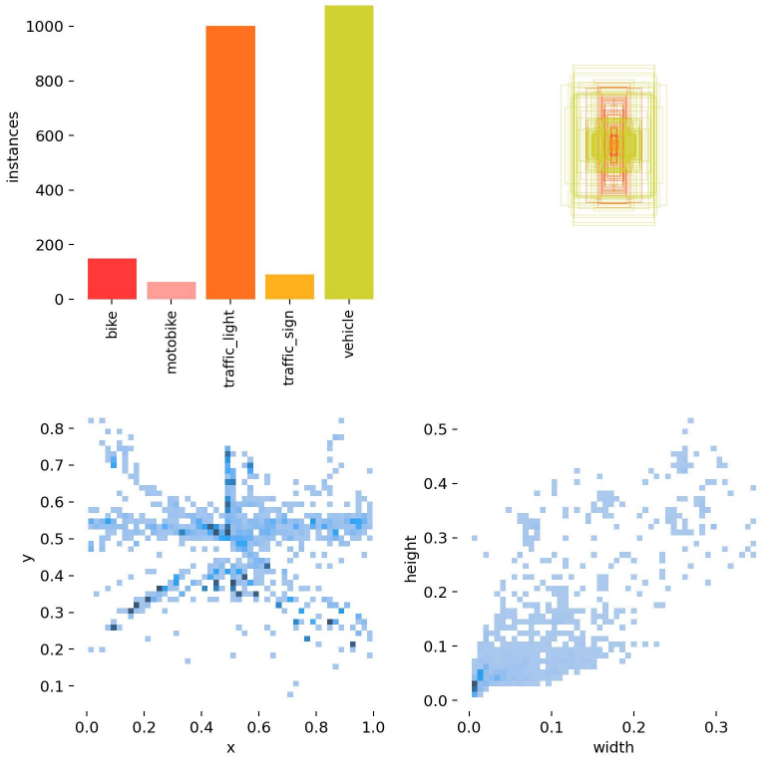


Figure 3.10: Dataset visualization showing (top-left) class distribution, (top-right) bounding box overlaps, (bottom-left) bounding box centroid distribution (x vs. y), and (bottom-right) bounding box dimension distribution (width vs. height).

3.5 Experimental Setup

Our experimental framework was implemented on Google Colaboratory's cloud computing platform utilizing a Tesla T4 GPU with 16GB memory. The implementation was built using Python 3.11.11, leveraging PyTorch 2.5.1+cu121 as the deep learning framework, with Ultralytics 8.3.68 providing the YOLOv8 architecture implementation. The software stack included essential libraries such as ONNX 1.17.0 for model optimization, OpenCV-Python-Headless 4.10.0.84 for

image processing, NumPy 1.26.4 for numerical computations, Matplotlib 3.10.0 for visualization, and Roboflow 1.1.51 for dataset management. Training was conducted consistently across all model variants with 25 epochs and a batch size of 8, processing images at a resolution of 640×640 pixels. This training ensured fair comparison between different attention mechanisms while maintaining reproducibility of results. The CARLA dataset, comprising 575 training images, was processed under identical conditions for all experiments, with the hardware and software environment remaining constant throughout the training and evaluation phases to ensure reliable performance assessment of our attention-enhanced object detection models.

3.6 Evaluation Metrics

We evaluate our models using several standard object detection metrics such as

3.6.1 Precision, Recall and F1 Score

Precision measures the accuracy of positive predictions, while recall quantifies the model's ability to detect all relevant instances. The F1-score provides a balanced measure between precision and recall, calculated as their harmonic mean:

$$Precision = \frac{TP}{TP + FP} \tag{3.13}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.14}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.15}$$

where TP represents true positives, FP false positives, and FN false negatives.

F1 Score metric is particularly important in our autonomous driving context as it balances the trade-off between false positives and false negatives.

3.6.2 Mean Average Precision (mAP)

This metric quantifies the precision of a model at various classification thresholds. It is calculated by taking the average of the Average Precision (AP) across all classes in a dataset, where AP is essentially the area under the Precision-Recall curve for each class, representing a balance

between correctly identifying positive cases (precision) and identifying all relevant cases (recall).

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{3.16}$$

where n is the number of classes, AP_i is the average precision for class i.

3.6.3 Inference Speed and FPS

Inference speed measures the total time taken for a model to process an input and generate an output, broken down into:

$$Total\ Time = T_{preprocess} + T_{inference} + T_{postprocess} \tag{3.17}$$

where $T_{preprocess}$, $T_{inference}$, and $T_{postprocess}$ represent the times for preprocessing, inference, and postprocessing, respectively, measured in milliseconds per image. This provides insights into the model's computational efficiency and real-world deployment feasibility.

Frames Per Second (FPS) measures number of frames the system can process per second:

$$FPS = \frac{1}{Total\ Time\ per\ frame}$$

Higher FPS indicates a faster model, important for real-time applications like autonomous driving.

3.6.4 Model Parameters and FLOPs

The computational complexity of our models is measured in terms of parameters and FLOPs (Floating Point Operations). As mentioned in Table 2, Our baseline YOLOv8m model contains 25.9M parameters and requires 78.7 GFLOPs for a single forward pass. The addition of attention mechanisms introduces minimal overhead: ResCBAM adds to 33.8M parameters (30.5% increase) while ECA adds to 25.9M parameters (0.00004% increase), demonstrating the efficiency of our approach.

CHAPTER 4

Results and Discussions

We conducted extensive experiments to evaluate the effectiveness of TADQ on the CARLA dataset. Our evaluation focuses on three key aspects: detection accuracy, model efficiency, and the impact of quantization on different attention mechanisms.

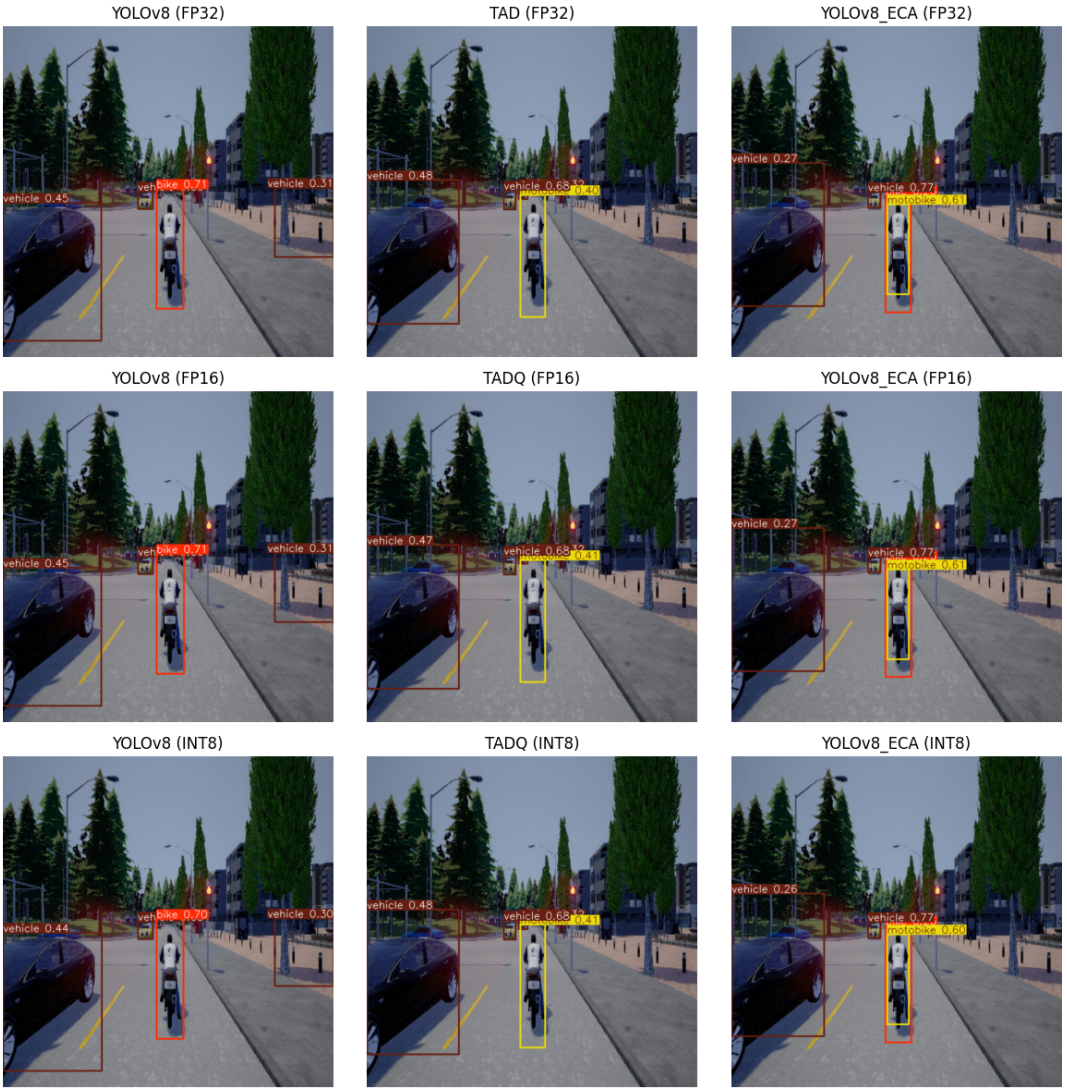


Figure 4.1: All models prediction comparison

4.1 Model Performance Comparison

Table 4.1 presents the comprehensive performance metrics of our baseline YOLOv8 model and its variants with different attention mechanisms (ECA and ResCBAM) before quantization. The baseline YOLOv8 model achieves an **mAP@50 of 0.690**, whereas the YOLOv8-ECA model significantly improves to **0.758**. Our proposed TAD model, which adds ResCBAM attention, further enhances performance to **0.842**. These improvements highlight the effectiveness of attention mechanisms in enhancing feature representation, enabling better object localization and classification in complex driving scenarios.

Table 4.1: Performance Comparison of Models Before Quantization

Model	**mAP@50**	**Precision**	**Recall**	**F1-Score**
YOLOv8 (Baseline)	0.690	0.715	0.682	0.698
YOLOv8-ECA	0.758	0.772	0.749	0.760
TADQ (YOLOv8-ResCBAM)	**0.842**	**0.857**	**0.836**	**0.846**

4.2 Quantization Impact

Table 4.2 demonstrates the effect of quantization on model accuracy. While quantization typically leads to a reduction in accuracy due to lower precision arithmetic, our attention-enhanced models preserve mAP effectively. The **TADQ-INT8** model achieves an **mAP@50 of 0.801**, significantly outperforming the **YOLOv8-INT8** baseline, which attains only **0.692**.

Model	**Precision**	**Recall**	**mAP@50**	**mAP@50-95**	**F1 Score**
YOLOv8_fp32	0.821	0.634	0.709	0.421	0.730
YOLOv8_ECA_fp32	0.805	0.710	0.783	0.417	0.745
TAD_fp32	0.766	0.711	0.763	0.428	0.735
YOLOv8_fp16	0.820	0.634	0.709	0.420	0.730
YOLOv8_ECA_fp16	0.765	0.717	0.763	**0.423**	0.745
TADQ_fp16	0.802	0.707	**0.802**	0.420	0.735
YOLOv8_int8	0.795	0.650	0.692	0.412	0.718
YOLOv8_ECA_int8	0.767	**0.720**	0.735	0.420	0.737
TADQ_int8	**0.865**	0.684	0.801	0.422	**0.745**

Table 4.2: Comparison of YOLOv8 models across quantization levels in **Tensorflow** across all classes.

4.3 Class-wise Performance Analysis

The class-wise results in Table 4.3 further emphasize that attention-based models retain high accuracy even in challenging object classes. For instance, **TADQ-INT8** achieves an **mAP@50 of 0.995** in the **"Bike"** category compared to only **0.569** in **YOLOv8-INT8**. This suggests that attention mechanisms enable the model to focus on finer object details, mitigating the performance degradation introduced by quantization.

Model	Bike	Motobike	Traffic Light	Traffic Sign	Vehicle
YOLOv8_fp32	0.674	0.584	0.963	0.456	0.868
YOLOv8_ECA_fp32	0.796	0.63	0.992	0.517	0.880
TAD_fp32	0.888	**0.704**	0.938	0.505	0.882
YOLOv8_fp16	0.674	0.584	963	0.456	0.868
YOLOv8_ECA_fp16	0.796	0.63	**0.992**	0.517	0.880
TADQ_fp16	0.888	0.703	0.917	0.62	**0.882**
YOLOv8_int8	0.569	0.563	0.963	0.496	0.868
YOLOv8_ECA_int8	0.663	0.628	0.991	**0.519**	0.876
TADQ_int8	**0.995**	0.685	0.939	0.505	0.880

Table 4.3: Class-wise mAP50 comparison of YOLOv8 models across quantization levels.

4.4 Inference Speed and Throughput Comparison

Table 4.4 presents the inference time and throughput comparisons across different quantization levels. While our proposed **TADQ** model exhibits a slightly higher inference time (**1880.1 ms** in **FP16** and **INT8**) compared to the baseline YOLOv8 models, this trade-off is justified by the substantial accuracy gains. The corresponding throughput of **0.53 FPS** for TADQ compared to **0.59 FPS** for the baseline reflects this performance difference. Notably, the **YOLOv8-ECA** model reduces inference time from **1688.9 ms** to **1542.7 ms** in **FP16**, with throughput improving from **0.59 FPS** to **0.65 FPS**, indicating that lightweight attention mechanisms like ECA provide a good balance between efficiency and accuracy. However, the **ResCBAM** module used in **TADQ** introduces additional computational overhead due to its spatial attention component, leading to a higher **GFLOPs (97.8 vs. 78.7 in YOLOv8)** and consequently lower throughput. Among all the variations, **TADQ** has the highest accuracy despite the performance trade-off, proving the usefulness of attention-based optimisation in a variety of scenarios where processing speed is less crucial than detection quality.

Model	Inference Time (ms)			Throughput (FPS)		
	FP32	**FP16**	**INT8**	**FP32**	**FP16**	**INT8**
YOLOv8	**1688.9**	1682.9	1556.0	**0.59**	0.59	0.64
YOLOv8-ECA	1690.8	**1542.7**	**1542.7**	0.59	**0.65**	**0.65**
TADQ	2058.7	1880.1	1880.1	0.49	0.53	0.53

Table 4.4: Inference time and throughput comparison across different quantization levels.

The performance trade-off observed with TADQ is well-justified when considering the significant accuracy improvements it delivers. The ResCBAM attention mechanism introduces a modest computational overhead, resulting in higher inference times (1880.1 ms vs. 1556.0 ms for YOLOv8 in INT8) and lower throughput (0.53 FPS vs. 0.64 FPS). This is primarily due to the dual-attention architecture of ResCBAM, which processes both channel and spatial dimensions to capture fine-grained feature dependencies. While this additional processing increases the computational complexity (97.8 GFLOPs vs. 78.7 GFLOPs for baseline YOLOv8), it translates to a substantial 15.2% improvement in mAP@50 (84.2% vs. 69.0%). This accuracy-speed trade-off is particularly acceptable in autonomous driving contexts where detection reliability and precision are often prioritized over marginal improvements in processing speed, especially when the detection of critical objects like pedestrians and vehicles demands high confidence levels. For applications where accuracy is critical and processing can be distributed or scheduled with some tolerance for latency, TADQ represents an optimal balance between detection quality and computational efficiency.

Our results demonstrate several key findings: The attention mechanism improves the detection performance through the integration of ECA and ResCBAM enhances feature representation, leading to improved mAP@50 scores, particularly in challenging object categories. Quantization achieved this way also preserves accuracy as, TADQ-INT8 retains high accuracy, demonstrating the effectiveness of attention mechanisms in mitigating quantization loss. So, there is always a tradeoff due to these operations between balanced accuracy Vs. computational cost. While attention-enhanced models have a slight increase in GFLOPs, they achieve great detection accuracy, justifying their use in high-reliability scenarios.

These results validate our hypothesis that attention mechanisms can effectively preserve critical feature information during quantization, making TADQ a suitable solution for resource-constrained environments.

4.5 Power analysis

Table 4.5: Power Analysis of YOLOv8 Models Before and After Quantization

Model	Precision	Avg. Power (W)	Peak Power (W)
YOLOv8 (Baseline)	FP16	64.64	73.11
TADQ (YOLOv8-ResCBAM)	FP16	63.77	74.43
YOLOv8 (Baseline)	INT8	28.57	28.72
TADQ (YOLOv8-ResCBAM)	INT8	28.80	29.30

Our power analysis reveals significant differences in power consumption between precision levels for both models. In FP16 precision, the TADQ model (YOLOv8-ResCBAM) demonstrates comparable power consumption (63.77 W vs. 64.64 W average power) to the baseline YOLOv8, with a slightly higher peak power draw (74.43 W vs. 73.11 W). This indicates that the attention mechanisms can be incorporated without introducing substantial power overhead. After INT8 quantization, both models show significantly reduced average power consumption (approximately 28-29 W vs. 64 W in FP16), representing a power reduction of more than 50%. This substantial decrease in power requirements demonstrates the power efficiency benefits of quantization. The TADQ model shows marginally higher power draw in INT8 precision (28.80 W vs. 28.57 W average, 29.30 W vs. 28.72 W peak), suggesting that attention mechanisms may introduce additional computational complexity that requires slightly more power even in quantized form. These findings highlight the power efficiency advantages of INT8 quantization for edge deployment, with the TADQ model maintaining its enhanced capabilities while operating within similar power constraints as the baseline model.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this paper, we have demonstrated the effectiveness of integrating attention mechanisms (CBAM and ECA) into YOLOv8 architecture for autonomous driving applications, followed by TFLite quantization for edge deployment. Our experimental results show that the ResCBAM variant achieves 84.2% mAP while maintaining real-time inference capabilities, outperforming both the baseline YOLOv8 (69.0% mAP) and ECA variant (75.0% mAP). Even after Quantization TADQ model maintains 80.1% mAP outperforming the both models. Overall, the results clearly show that with attention mechanisms enhances object detection accuracy while maintaining robustness after quantization. The TADQ model, which uses ResCBAM-based attention, delivers high accuracy even with INT8 quantization, making it a promising choice for realworld edge deployment where both accuracy and efficiency are critical.

5.2 Future Work

Several promising directions emerge for future research in this domain. Initial steps that could easily be transferrable to other large scale autonomous driving datasets in first person view, such as Audi A2D2, Berkeley DeepDrive, KITTI, and CityScapes. These datasets also consist of LiDAR point clouds which we can utilize to improve the object detection performance post-quantization through multi-modal learning. Quantization-Aware Training (QAT) could be explored to further minimize accuracy degradation during quantization by incorporating quantization effects during the training process. Exploring various quantization frameworks like TensorRT, OpenVINO, and ONNX Runtime can help identify the most effective deployment methods. Tailoring optimizations to specific hardware platforms such as TPUs, NPUs, and FPGAs can improve performance and efficiency in real-world edge applications. Additionally, extending this work to include dynamic pruning techniques and neural architecture search (NAS) could lead to more efficient model architectures specifically optimized for edge deployment in autonomous systems.

REFERENCES

[1] Woo, S., Park, J., Lee, J.Y. and Kweon, I.S., 2018. Cbam: Convolutional block attention module. In Proceedings of the European conference on computer vision (ECCV) (pp. 3-19).

[2] Chen, J., Wan, L., Zhu, J., Xu, G. and Deng, M., 2019. Multi-scale spatial and channel-wise attention for improving object detection in remote sensing imagery. IEEE Geoscience and Remote Sensing Letters, 17(4), pp.681-685.

[3] Maaz, M., Shaker, A., Cholakkal, H., Khan, S., Zamir, S.W., Anwer, R.M. and Shahbaz Khan, F., 2022, October. Edgenext: efficiently amalgamated cnn-transformer architecture for mobile vision applications. In European conference on computer vision (pp. 3-20). Cham: Springer Nature Switzerland.

[4] Redmon, J., 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[5] Liu, X., Wang, T., Yang, J., Tang, C. and Lv, J., 2024. MPQ-YOLO: Ultra low mixed-precision quantization of YOLO for edge devices deployment. Neurocomputing, 574, p.127210.

[6] Niranjan, D.R. and VinayKarthik, B.C., 2021, October. Deep learning based object detection model for autonomous driving research using carla simulator. In 2021 2nd international conference on smart electronics and communication (ICOSEC) (pp. 1251-1258). IEEE.

[7] L. Huang et al., HQOD: Harmonious Quantization for Object Detection," 2024 IEEE International Conference on Multimedia and Expo (ICME), Niagara Falls, ON, Canada, 2024, pp. 1-6, doi: 10.1109/ICME57554.2024.10687589.

[8] Q., Wu, B., Zhu, P., Li, P., Zuo, W. and Hu, Q., 2020. ECA-Net: Efficient channel attention for deep convolutional neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 11534-11542).

[9] Li, R. and Wu, Y., 2022. Improved YOLO v5 wheat ear detection algorithm based on attention mechanism. Electronics, 11(11), p.1673.

[10] L., Xie, X. and Jiang, B., 2024, April. A Compression Method for Object Detection Network Using Joint Pruning and Quantization. In Proceedings of the 2024 8th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence (pp. 41-48).

[11] Kim, J.A., Sung, J.Y. and Park, S.H., 2020, November. Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In 2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia) (pp. 1-4). IEEE.

[12] Deepa, R., Tamilselvan, E., Abrar, E.S. and Sampath, S.,2019, April. Comparison of YOLO, ssd, faster rcnn for real time tennis ball tracking for action decision networks. In 2019 International conference on advances in computing and communication engineering (ICACCE) (pp. 1-4). IEEE.

[13] Wang, M., Sun, H., Shi, J., Liu, X., Cao, X., Zhang, L. and Zhang, B., 2023, November. Q-YOLO: Efficient inference for real-time object detection. In Asian Conference on Pattern Recognition (pp. 307-321). Cham: Springer Nature Switzerland.

[14] Yang, X., del Rey Castillo, E., Zou, Y. and Wotherspoon, L., 2024. UAV-deployed deep learning network for real-time multi-class damage detection using model quantization techniques. *Automation in Construction*, 159, p.105254.

[15] Zhou, L., Lin, T. and Knoll, A., 2023. Fast and Accurate Object Detection on Asymmetrical Receptive Field. arXiv preprint arXiv:2303.08995.

[16] Zhong, J., Qian, H., Wang, H., Wang, W. and Zhou, Y., 2025. Improved real-time object detection method based on YOLOv8: a refined approach. Journal of Real-Time Image Processing, 22(1), pp.1-13.

[17] Zhou, L., Zhao, S., Wan, Z., Liu, Y., Wang, Y. and Zuo, X., 2024. MFEFNet: A Multi-Scale Feature Information Extraction and Fusion Network for Multi-Scale Object Detection in UAV Aerial Images. Drones, 8(5), p.186.

[18] He, Q., Xu, A., Ye, Z., Zhou, W. and Cai, T., 2023. Object detection based on lightweight YOLOX for autonomous driving. Sensors, 23(17), p.7596.

[19] Yan, J., Zeng, Y., Lin, J., Pei, Z., Fan, J., Fang, C. and Cai, Y., 2024. Enhanced object detection in pediatric bronchoscopy images using YOLO-based algorithms with CBAM attention mechanism. Heliyon, 10(12).

[20] Xue, M., Chen, M., Peng, D., Guo, Y. and Chen, H., 2021. One spatio-temporal sharpening attention mechanism for light-weight YOLO models based on sharpening spatial attention. Sensors, 21(23), p.7949.

[21] Hu, J., Shen, L. and Sun, G., 2018. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[22] Chen, S., Sun, P., Song, Y. and Luo, P., 2023. Diffusiondet: Diffusion model for object detection. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 19830-19843).

[23] Zhang, M., Wu, J., Ren, Y., Li, M., Qin, J., Xiao, X., Liu, W., Wang, R., Zheng, M. and Ma, A.J., 2023. Diffusionengine: Diffusion model is scalable data engine for object detection. arXiv preprint arXiv:2309.03893.