

BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

Squeeze-Aware YOLO: An improved YOLO for Traffic Sign Detection

Mizbaul Haque Maruf

200042125

Syed Tamzid Bakht

200042145

Syed Samin Sadaf

200042163

Department of Computer Science and Engineering

Islamic University of Technology

October, 2025

Squeeze-Aware YOLO: An improved YOLO for Traffic Sign Detection

Mizbaul Haque Maruf

200042125

Syed Tamzid Bakht

200042145

Syed Samin Sadaf

200042163

Department of Computer Science and Engineering

Islamic University of Technology

October, 2025

Declaration of Candidate

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Mizbaul Haque Maruf**, **Syed Tamzid Bakht**, and **Syed Samin Sadaf** under the supervision of **Dr. Md. Hasanul Kabir**, Professor, Department of Computer Science and Engineering and co-supervision of **Sabbir Ahmed**, Assistant Professor, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Dr. Md. Hasanul Kabir

Professor

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Date: October 26, 2025

Mizbaul Haque Maruf

Student ID: 200042125

Date: October 26, 2025

Syed Tamzid Bakht

Student ID: 200042145

Date: October 26, 2025

Sabbir Ahmed

Assistant Professor

Department of Computer Science and Engineering

Islamic University of Technology (IUT)

Date: October 26, 2025

Syed Samin Sadaf

Student ID: 200042163

Date: October 26, 2025

*Dedicated to our supervisor, whose unwavering guidance
and profound insights have been the cornerstone of our
academic journey and the success of this work.*

Contents

1	Introduction	1
1.1	Overview	1
1.2	Motivation	2
1.3	Problem Statement	3
1.4	Research Challenges	3
1.5	Contributions	4
2	Literature Review	5
2.1	Deep Learning Based Object Detection	5
2.2	Two Stage Detector	6
2.2.1	Region Based Convolutional Neural Networks (RCNN)	6
2.3	Single Stage Detector	8
2.3.1	EfficientDet	8
2.3.2	You Only Look Once (YOLO)	10
2.3.3	MHAF-YOLO	18
2.4	Transformer Based Model	22
2.4.1	Detection Transformer(DETR)	22
2.5	Publicly Available Traffic Sign Datasets	24
2.5.1	Early Benchmark Datasets	24
2.5.2	Large-Scale and Diverse Datasets	27
2.5.3	Regionally Specialized Datasets	30
2.5.4	Summary and Observations	33
2.6	Existing Techniques for Traffic Sign Detection	34
3	Proposed Methodology	39
3.1	Overview of Proposed Architecture	39
3.2	Multi-Branch Auxiliary FPN	41
3.3	Re-parameterized Heterogeneous Multi-Scale Module	42

3.4	Re-parameterized Heterogeneous Depthwise Convolution	43
3.5	Squeeze-and-Excitation (SE) Module	44
3.6	SE-Enhanced Bottleneck Layer	46
4	Results and Discussion	50
4.1	Dataset	50
4.1.1	Bangladeshi Traffic Sign Dataset	50
4.1.2	GTSDb	51
4.1.3	CCTSDb2021	52
4.2	Experiment Setup	52
4.3	Evaluation Metrics	53
4.4	Performance Comparison with State of the Art Models	54
4.5	Performance on Bangladeshi Traffic Sign Dataset	60
4.6	Performance on German Traffic Sign Dataset	64
4.7	Performance on CCTSDb2021 Dataset	68
4.8	Error Analysis	70
5	Conclusion	75
5.1	Summary	75
5.2	Limitations	76
5.3	Future Work	77
	References	78

List of Figures

2.1	General pipeline illustrating the step-by-step workflow of deep learning-based object detection [19]	6
2.2	Overview of R-CNN showing how region proposals are combined with CNN features for object detection [30]	7
2.3	EfficientDet architecture highlighting the integration of EfficientNet as the backbone and BiFPN for multi-scale feature fusion [62]	9
2.4	Main stages of the YOLO algorithm, from grid division to bounding-box prediction [29]	11
2.5	Core architecture of the YOLO family showing feature extraction and detection heads [48]	12
2.6	Comparison between Dense Aggregation Network (DAN) and One-Shot Aggregation (OSA), emphasizing feature reuse and computational differences [37]	13
2.7	Structural comparison of CSPNet, ELAN, and GELAN, illustrating different strategies for gradient flow and feature learning [71]	15
2.8	Architectural comparison of the C2f block and the earlier C3 block, focusing on feature fusion efficiency [65]	16
2.9	Key components of YOLOv10: (a) Compact Inverted Block and (b) Partial Self-Attention Block, used to boost accuracy while maintaining efficiency [70]	17
2.10	Comparison between the C2f block and the C3K2 block, demonstrating kernel expansion and multi-scale enhancement on YOLO architecture [72]	18
2.11	Overall architecture of MHAF-YOLO, showing how multi-branch feature aggregation improves detection [77]	19
2.12	Structure of the Superficial Assisted Fusion (SAF) module, designed to preserve shallow spatial details [77]	20

2.13	Structure of the Advanced Assisted Fusion (AAF) module, aimed at enhancing deeper semantic feature interactions [77]	21
2.14	Architecture of RepHMS, illustrating hybrid multi-scale representation with re-parameterizable blocks [77]	21
2.15	Example of re-parameterizing small kernels (3×3 , 5×5) into a larger equivalent kernel (7×7) for inference optimization [77]	22
2.16	Object detection framework of DETR, showing transformer-based end-to-end prediction [6]	23
2.17	Efficient Hybrid Encoder combining CNN and transformer features for improved global-local representation [84]	23
2.18	Sample images from the Belgium Traffic Sign Dataset, illustrating category diversity [67]	25
2.19	Sample images of Swedish Traffic Sign Dataset illustrating situations of Highway [36]	25
2.20	Sample Signs from German Traffic Sign Recognition Benchmark (GT-SRB) [57]	26
2.21	Sample images from German Traffic Sign Detection Benchmark (GTSDb) simulating real road side situations [26]	27
2.22	Sample scenes from the Mapillary Traffic Sign Dataset (MTSD), highlighting real-world complexity [14]	28
2.23	Sample images from LISA Traffic Sign Dataset [46]	28
2.24	Sample images from the TT100K Dataset, showing dense traffic sign clutter [86]	29
2.25	Example images from the CCTSDB2021 Dataset displaying Chinese road conditions [81]	30
2.26	Malaysian Traffic Sign Dataset examples illustrating local sign variations [43]	31
2.27	Sample images from Indian Traffic Sign Dataset [44]	32
2.28	Example scenes from Bangladeshi Traffic Sign Detection Benchmark (BTSD) [3]	32
2.29	Sample images from BanTrafficNet showing lack of real driving circumstances [69]	33
3.1	Overview of the Proposed Architecture	40
3.2	Illustration of the Squeeze-and-Excitation (SE) process used for channel-wise feature recalibration	44
3.3	High-level overview of the proposed block architecture	46

3.4	Comparison of different bottleneck types and their feature transformation patterns	47
3.5	Detailed view of the proposed block, showing internal operations and feature flow	49
4.1	F1-score curve comparison across different models in Bangladeshi Dataset	61
4.2	Precision–Recall (PR) curve comparison across models on Bangladeshi Dataset	62
4.3	Sample images illustrating the superior performance of the model on the Bangladeshi Traffic Sign Dataset.	63
4.4	Sample instances illustrating the superior performance of the model on the Bangladeshi Traffic Sign Dataset	64
4.5	F1-score curves of models evaluated on German Dataset	65
4.6	PR curve of the models evaluated on German Dataset	66
4.7	Sample images demonstrating the improved detection performance and higher confidence scores achieved by the proposed model on the GTSDb dataset	67
4.8	Sample images demonstrating higher confidence scores achieved by the proposed model on the GTSDb dataset	68
4.9	F1 curve comparison of the models in CCTSDb	69
4.10	PR curve comparison of the models in CCTSDb	70
4.11	Confusion matrix of the Squeeze-Aware-YOLO model on the CCTSDb dataset, illustrating its class-wise detection performance	71
4.12	Examples of CCTSDb cases where no improvement is observed	71
4.13	Additional CCTSDb samples showing no significant model improvement	72
4.14	Confusion matrix of the Squeeze-Aware-YOLO model on the Bangladeshi dataset, highlighting class-wise misdetections and overall recognition performance	73
4.15	Confusion matrix of the Squeeze-Aware-YOLO model on the German dataset, highlighting class-wise misdetections and overall recognition performance	74

List of Tables

2.1	Scaling configuration of EfficientDet models (D0–D7), showing depth, width, and resolution coefficients [62].	9
2.2	Structural summary of the Darknet-19 backbone used in earlier YOLO versions [49]	13
2.3	Architectural breakdown of the Darknet-53 backbone used in YOLOv3 [50]	14
4.1	Number of images available for each traffic sign category in Bangladeshi dataset.	51
4.2	Distribution of objects across different traffic sign categories in the German dataset.	52
4.3	List of evaluated models along with their total parameter counts . . .	55
4.4	Performance comparison of all models on the Bangladeshi Traffic Sign Dataset	55
4.5	Class-wise performance metrics for various YOLO models in Bangladeshi Dataset	57
4.6	mAP scores of different models on the German Traffic Sign Dataset (GTSDb)	58
4.7	mAP performance for individual traffic sign classes in the GTSDb dataset	59
4.8	Overall mAP scores of evaluated models on the CCTSDb dataset . . .	59
4.9	Class-specific mAP values for the CCTSDb2021 traffic sign dataset . .	60

List of Abbreviations

CNN	Convolutional Neural Network
GTSDB	German Traffic Sign Detection Benchmark
GTSRB	German Traffic Sign Recognition Benchmark
CCTSDB	CSUST Chinese Traffic Sign Detection Benchmark
MHAF	Multi-Branch Heterogeneous Auxiliary Fusion
YOLO	You Only Look Once
SAF	Superficial Assisted Fusion
AAF	Advanced Assisted Fusion
SE	Squeeze and Excitation
RepHMS	Re-parameterized Heterogeneous Multi-Scale
GHFKS	GlobalH eterogeneous Flexible Kernel Selection
PAFPN	Path Aggregation Feature Pyramid Network
MAFPN	Multi-Branch Auxiliary Feature Pyramid Network

Acknowledgement

We would like to start by expressing our deepest gratitude to Almighty Allah for allowing us to complete our research successfully. We are indeed grateful to be able to submit our thesis work by which are eventually going to end our Bachelor of Science study.

Next we would like to express our honest appreciation to Dr. Md. Hasanul Kabir, Professor, Department of Computer Science and Engineering, Islamic University of Technology and Sabbir Ahmed, Lecturer, Department of Computer Science and Engineering, Islamic University of Technology for being our adviser and mentor. Their inspirations, ideas and observations for this particular research work were invaluable, and this thesis would have never been totally successful without their encouragement and proper guidance. Their valuable opinion, time and input provided throughout the thesis work, from the first phase of thesis topics introduction, research area selection, proposition of algorithm, modification and implementation helped us to do our thesis work in proper way. We are grateful to them for their constant and energetic guidance and valuable advice.

We extend our appreciation to all the respected jury members of the thesis committee for their thoughtful comments and constructive criticism of our research work. They certainly helped us develop this work of science. Finally, we would like to express our sincere gratitude to all the faculty members of the Computer Science and Engineering department of Islamic University of Technology for providing us with a congenial and supportive work environment.

Abstract

Traffic sign detection is a critical component of intelligent transportation systems. Reliable detection under varying lighting, weather, and occlusion conditions is essential for their effective operation. However, many existing real-time detectors struggle to preserve fine-grained spatial details, leading to reduced performance on small or visually degraded signs. To address this gap, we propose Squeeze-Aware-YOLO, an enhanced variant of MHAF-YOLO that integrates a Squeeze-Aware block composed of triple bottlenecks with diverse kernels and a squeeze-and-excitation mechanism. We evaluate Squeeze-Aware-YOLO on the Bangladeshi Traffic Sign Detection Dataset, the German Traffic Sign Detection Benchmark, and CCTSDB2021. On the Bangladeshi dataset, our model achieves a 74.7% mAP@0.5, outperforming the base MHAF-YOLO and surpassing state-of-the-art models including YOLOv8, YOLOv10, and YOLOv11. On the German dataset, Squeeze-Aware-YOLO attains an 84.0% mAP@0.5, outperforming all compared approaches, while on CCTSDB2021 it reaches a 78.3% mAP@0.5, exceeding the performance of MHAF-YOLO. These results validate that the integration of the Squeeze-Aware block with MHAF-style multi-branch fusion and adaptive kernel selection yields a robust real-time detector, particularly well-suited to the diverse and challenging nature of traffic sign recognition tasks.

Chapter 1

Introduction

1.1 Overview

In computer vision, recognition refers to identifying the category of an object present in an image, while object detection extends this task by determining both the class and the precise location of each object through spatial localization. Because detection requires instance-level classification together with bounding box prediction, it is considered a more demanding and fine-grained problem than recognition alone [58]. Over the past decade, deep learning particularly convolutional neural networks (CNNs) has played a central role in advancing the performance of both recognition and detection systems [16]. Among the many application areas of object detection, traffic sign detection holds particular importance for autonomous driving and advanced driver-assistance systems (ADAS). Accurate and timely identification of traffic signs allows intelligent vehicles to interpret road rules, adapt their behavior, and improve overall road safety. A wide range of studies has applied deep learning-based detection frameworks to this task, leveraging CNNs and their modern successors to handle diverse environments and sign variations [3], [15], [28], [66], [81], [86]. Despite steady progress, a significant number of challenges persist: models that achieve strong performance on one dataset often fail to generalize when deployed in different regions or conditions, variations in sign design across countries, changes in lighting, occlusion, weather, and background clutter frequently lead to notable drops in detection accuracy. For instance, a model trained on the German Traffic Sign Recognition Benchmark (GTSRB) may not maintain comparable performance on traffic signs from Bangladesh or the United States due to differences in visual appearance and environmental context. The goal of this research is to design and develop a robust traffic sign detection model capable of delivering consistent performance across diverse datasets

and real-world scenarios. Instead of focusing solely on maximizing accuracy for a single benchmark, this work emphasizes cross-domain generalization. Furthermore, our work evaluates the model on multiple region-specific datasets, including those that reflect the complex and less structured road environments found in countries like Bangladesh. To achieve this, we study existing state-of-the-art detection frameworks and introduce targeted improvements within the YOLO architecture. Experimental results across several datasets show that the proposed model not only matches competitive baselines but also demonstrates enhanced generalization. As a result, our proposed architecture is more suitable for practical deployment in varied traffic conditions.

1.2 Motivation

The motivation for this research originates from the adverse traffic environments of Bangladesh, where road safety remains a persistent concern. Every year, thousands of lives are lost in traffic accidents in the capital city Dhaka alone. The city witnesses approximately 1,640 road accidents over the past five years [11]. These incidents not only cause human suffering but also impose substantial social and economic costs. That includes reduced productivity, strain on healthcare systems, and long-term psychological trauma to affected communities. One significant cause of road accidents is the failure to recognize traffic signs either by human drivers or autonomous systems [3]. In countries like Bangladesh, where roads are extremely congested, traffic signs often appear partially occluded, weathered, or surrounded by visual clutter such as advertisements, posters and non-standard signage. Moreover, traffic signs frequently occupy only a small portion of the visual scene making them harder to distinguish. Also, adverse weather conditions, poor illumination and non-standard placements further degrade recognition performance. These real-world challenges inspired us to explore more reliable and accurate traffic sign detection techniques capable of handling such complexities. While our work is deeply rooted in the need for better detection in Bangladeshi conditions, the problem extends well beyond one region. In fact, most of the existing object detection models tend to overfit to specific datasets. Then, they struggle when applied to different geographical regions or environmental conditions. This observation leads us to a broader goal: to develop a robust traffic sign detection model that generalizes well across multiple datasets, including those collected from diverse and challenging environments. A generalizable model is crucial for ensuring safety and reliability in global autonomous driving systems. Because of its adaptability, it is suitable for varied traffic scenes.

1.3 Problem Statement

The objective of this research is to address the following key problems:

- Existing traffic sign detection models often perform well on the datasets they are trained on but fail to generalize to other datasets.
- With the introduction of varying sign designs, diverse backgrounds, and challenging environmental conditions, detection models often struggle to maintain stable performance.
- Traffic signs often occupy a very small region in the image, leading to a high rate of missed detections, especially in dense urban scenes.

1.4 Research Challenges

Traffic sign detection presents a unique set of challenges that make the development of a robust and generalizable model a highly complex task. One of the most prominent issues is the inherently small size of traffic signs within images. On average, traffic signs occupy only about 0.2% of an image’s area [3], making them difficult to detect accurately. This small object size significantly reduces the effectiveness of anchor-based or grid-based detection methods. This often leads to miss tiny visual cues. Despite the availability of several well-established datasets such as the German Traffic Sign Detection Benchmark (GTSDB) [57], the Belgium Traffic Sign Dataset [67], the Tsinghua-Tencent 100k [86] and the LISA Dataset [46], most models trained on these datasets struggle to perform reliably under real-world conditions. These datasets typically feature controlled environments. Models trained on them often fail to generalize when exposed to environmental variations such as poor lighting, occlusion, motion blur, weathering of sign surfaces and visually cluttered backgrounds. Such real-world variability remains a major obstacle in deploying traffic sign detection systems in diverse geographic and urban contexts. Moreover, we have seen the architectural limitations of existing detection frameworks. Transformer-based models have shown strong capabilities in capturing global contextual information and modeling long-range dependencies. However, they are computationally intensive and often lack the fine-grained local sensitivity required for accurately detecting small and dense objects like traffic signs. Conversely, convolutional neural network based models are more computationally efficient and can adapt by extracting local features. Yet they struggle to detect objects at varying scales and are less effective when signs are distant or partially visible. YOLO-based architectures are popular for their real-time detection capabil-

ities and overall accuracy. However, they often face difficulties in achieving precise localization and detecting small objects within complex or noisy scenes. Another significant challenge is the lack of cross-dataset generalization. Many detection models exhibit strong performance on the specific dataset they are trained on but suffer considerable performance drops when applied to new datasets from different domains. These dataset specific biases highlight the absence of robust domain adaptation in current approaches. Achieving consistent performance across geographically diverse and visually distinct datasets remains an unresolved problem.

1.5 Contributions

The key contributions of this research are summarized as follows:

- We propose a feature enhancement module placed before the backbone, which improves standard bottleneck structures using multiple kernel sizes and a Squeeze-and-Excitation mechanism. This allows adaptive channel recalibration and richer feature representation, enhancing the performance of the following detection stages.
- Experiments on the German, Bangladeshi, and CCTSDB2021 traffic sign datasets demonstrate that the proposed method outperforms existing state-of-the-art approaches, therefore, confirms its effectiveness and generalizability.

This article is structured as follows: Chapter 2 reviews the literature on object detection, covering different approaches and techniques used over the years along with different publically available datasets. Chapter 3 explains the method to address the problem. Chapter 4 describes the datasets, the experiments conducted, and analyzes the results. Finally, Chapter 5 summarizes our findings, discusses the limitations of our work, and outlines our plans for future research.

Chapter 2

Literature Review

Object detection is a computer vision process that finds and labels objects in an image. It has two main steps: first, it draws boxes around the objects, and then it names them. There are two ways to do object detection: using traditional methods or deep learning techniques. In recent times, traditional methods have failed to keep up with the performance of deep learning based approaches [68].

2.1 Deep Learning Based Object Detection

Deep learning based object detectors employ convolutional neural networks (CNNs) to perform end-to-end object detection by automatically learning and extracting increasingly complex feature representations [19], [73]. A Convolutional Neural Network consists of convolutional layers, activation functions, pooling layers, fully connected layers, and a classifier. In these architectures, convolutional layers employ filter kernels that capture characteristic spatial patterns. Furthermore, activation functions introduce the necessary nonlinearity and determine which neuron to fire. Intermediate pooling layers then down-sample the resulting feature maps. The process reduces dimensionality and computational cost without sacrificing essential information. Later, fully connected layers aggregate the distilled features and project them into a lower-dimensional space aligned with the target categories. The classifier is usually implemented using a softmax function. This layer transforms the network's output vector into a normalized probability distribution over the predefined object classes. By unifying these components into a single trainable pipeline, CNN-based detectors jointly optimize feature extraction and decision boundaries. Therefore, it achieves both accuracy and efficiency in detecting multiple object types within an image.

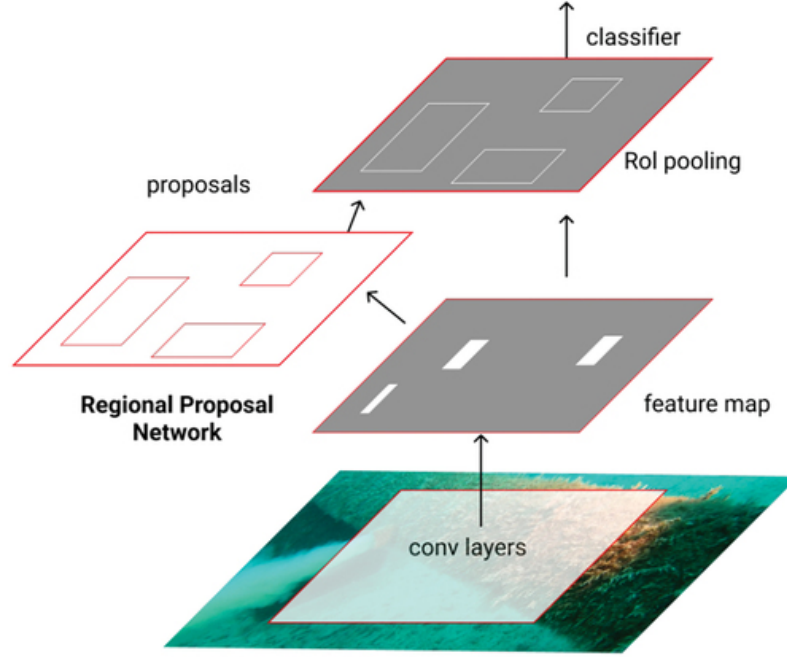


Figure 2.1: General pipeline illustrating the step-by-step workflow of deep learning-based object detection [19]

Object detection algorithms are generally categorized into two primary architectures: single-stage and two-stage detectors. Two-stage detectors such as R-CNN and its variants, first generate region proposals and subsequently refine these regions through classification and localization. In contrast, single-stage detectors including models like EfficientDet and YOLO, perform detection in a unified step. As a result, it achieves faster inference at the expense of reduced accuracy in complex scenarios.

2.2 Two Stage Detector

Two-stage object detectors separate the detection process into distinct phases: region proposal generation followed by region classification and refinement. These models typically achieve higher accuracy than one-stage detectors at the cost of increased computational complexity and slower inference speeds. The region proposal step identifies potential object locations, while the second stage classifies these regions and refines their bounding boxes.

2.2.1 Region Based Convolutional Neural Networks (RCNN)

The introduction of Region-based Convolutional Neural Networks (R-CNN) marked a significant advancement in object detection and localization. Rather than exhaustively searching across the entire image space for candidate windows, R-CNN uses re-

gion proposal function to identify a limited set of candidate regions likely to contain objects [22]. Specifically, approximately 2000 region proposals are generated in the first stage using a selective search algorithm. Each region is then processed through a convolutional neural network to extract discriminative features. After that they are subsequently classified to determine object presence and category.

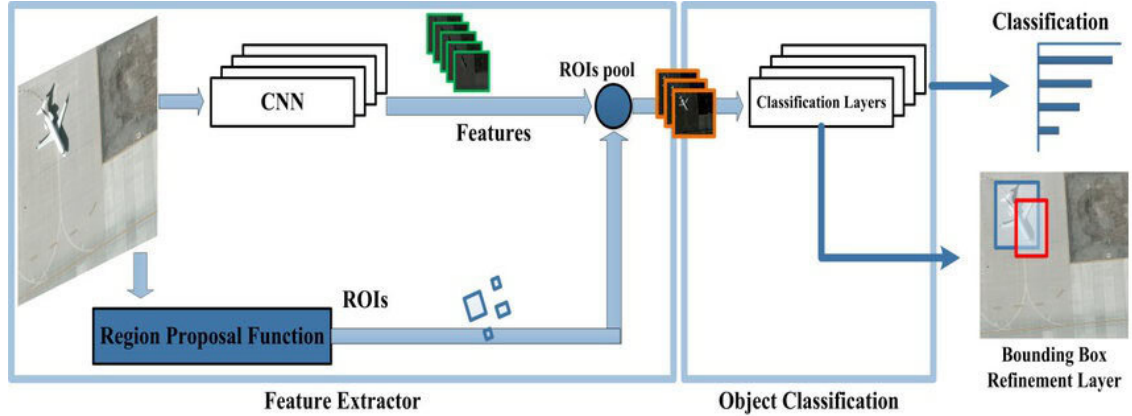


Figure 2.2: Overview of R-CNN showing how region proposals are combined with CNN features for object detection [30]

While R-CNN achieved remarkable improvements in detection accuracy, its multi-stage pipeline comprising region proposal generation, feature extraction, and classification introduced significant computational overhead [53]. It has made real-time inference infeasible. To mitigate this limitation, Fast R-CNN was proposed. Fast R-CNN introduced a convolutional implementation of sliding windows to enhance efficiency in region processing [21]. Instead of processing each region independently, Fast R-CNN applies convolutional feature extraction to the entire image once and subsequently reuses these shared feature maps for all proposed regions [24], [47]. This approach considerably reduces redundancy, enabling faster localization and detection.

Despite these improvements, the dependence on an external region proposal mechanism (e.g., selective search) continued to bottleneck performance. Faster R-CNN addressed this challenge by integrating a Region Proposal Network (RPN) directly within the detection pipeline. Therefore, it enables end-to-end training and significantly improve inference speed [52]. The RPN generates a set of anchor boxes typically nine per spatial location across multiple scales and aspect ratios. Each anchor is assigned an objectness score to indicate the likelihood of containing an object, and only high-confidence anchors are retained for further processing.

In Faster R-CNN, a Region of Interest (RoI) pooling layer converts variable-sized pro-

posals into fixed-size feature vectors through shared convolutional computation [74]. A subsequent classifier, often a Support Vector Machine (SVM) or a fully connected neural network, determines whether each region contains an object and categorizes it as foreground or background [52]. Deep residual networks, such as ResNet-101 or ResNet-152, are frequently employed as backbone feature extractors due to their superior representational capacity. Anchors that exhibit significant overlap with ground-truth bounding boxes (Intersection-over-Union, $\text{IoU} > 0.5$) are considered foreground samples. The model optimizes detection performance using binary cross-entropy loss for classification and a smooth L1 regression loss for bounding box refinement. To suppress redundant detections, Non-Maximum Suppression (NMS) is applied, retains only the highest-scoring bounding boxes. NMS discards overlapping ones below a predefined IoU threshold, thereby improving precision and computational efficiency. Despite that High computational cost and multiple processing stages can not make it suitable for real-time applications.

2.3 Single Stage Detector

One-stage object detectors perform object localization and classification in a single forward pass through the neural network, without generating region proposals. These models are characterized by their high inference speed and efficiency, making them suitable for real-time applications. Two of the most prominent one-stage detectors are EfficientDet and the YOLO (You Only Look Once) family, both have undergone significant evolution since their introduction.

2.3.1 EfficientDet

The EfficientNet architecture was a major milestone in the design of deep convolutional neural networks. It employs neural architecture search (NAS) to optimize both accuracy and efficiency across multiple dimensions [60]. The model introduces a compound scaling method that uniformly scales the network’s depth, width, and resolution using a single scaling coefficient [60]. The underlying intuition of this approach is that larger input resolutions require deeper networks to expand the receptive field and wider layers to effectively capture fine-grained spatial details.

Building upon this foundation, EfficientDet was developed by integrating EfficientNet as the backbone and introducing a Bidirectional Feature Pyramid Network (BiFPN) for multi-scale feature fusion [62]. The original Feature Pyramid Network (FPN) was proposed to enhance the detection of objects at varying scales by combining high-level

semantic features with low-level spatial features [40]. However, BiFPN refines this concept by introducing bidirectional cross-scale connections and learnable weights that allow the model to adaptively prioritize feature contributions across scales. This design enables efficient information flow in both top-down and bottom-up directions, thereby improving feature representation and detection performance.

Table 2.1: Scaling configuration of EfficientDet models (D0–D7), showing depth, width, and resolution coefficients [62].

Model	Input Size	Backbone	BiFPN		
			# Channels	# Layers	Box/Class # Layers
D0	512	B0	64	3	3
D1	640	B1	88	4	3
D2	768	B2	112	5	3
D3	896	B3	160	6	4
D4	1024	B4	224	7	4
D5	1280	B5	288	7	4
D6	1280	B6	384	8	5
D7	1536	B6	384	8	5
D7x	1536	B6	384	8	5

The primary advantage of EfficientDet lies in achieving state-of-the-art performance with substantially fewer parameters and floating-point operations per second (FLOPs) compared to prior detectors. Specifically, the model attains a mean Intersection over Union (mIoU) of 81.74% while utilizing only 18 billion FLOPs and 17 million parameters [62]. Through compound scaling, larger variants such as EfficientDet-D7 and its extended version, EfficientDet-D7x, demonstrate improved detection accuracy while maintaining computational efficiency.

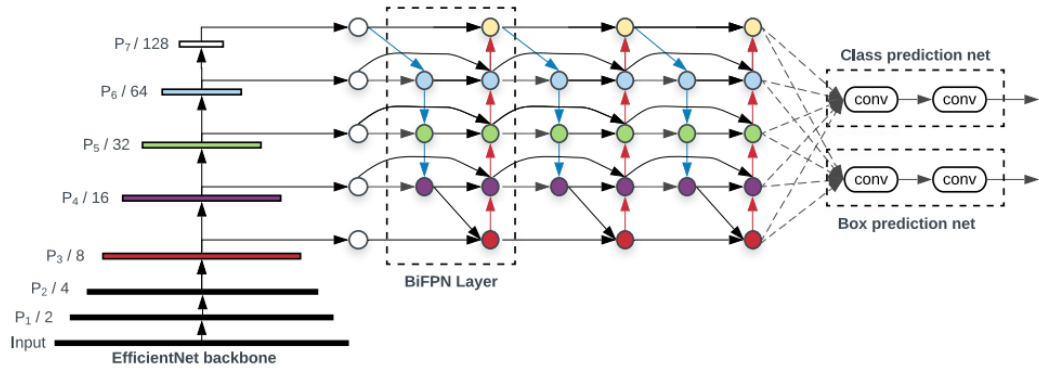


Figure 2.3: EfficientDet architecture highlighting the integration of EfficientNet as the backbone and BiFPN for multi-scale feature fusion [62]

Although the uniform compound scaling of EfficientNet enables a balanced trade-off between performance and complexity, it exhibits certain limitations. Specifically,

not all network stages contribute equally to training speed and accuracy. To address this, EfficientNetV2 was introduced with a non-uniform scaling strategy that incrementally increases layer depth in later stages and adjusts regularization dynamically according to image size [61]. This progressive training approach enhances both convergence speed and overall performance on large-scale datasets.

Recent research has demonstrated that replacing the original EfficientNet backbone with EfficientNetV2 can further improve detection accuracy. For example, the approach proposed by Aldahoul et al. [2] employs EfficientNetV2 as the primary backbone, achieving superior performance with a 94% accuracy rate. The framework further introduces a secondary stage that utilizes EfficientDet-D4 to classify cropped object regions, outperforming several contemporary state-of-the-art detection architectures. However, due to multi-scale feature fusion and heavy backbone dependencies, EfficientDet hinders real-time deployment.

2.3.2 You Only Look Once (YOLO)

YOLO (You Only Look Once) represents a paradigm shift in object detection by introducing a single-stage architecture that predicts bounding boxes and class probabilities simultaneously. Unlike traditional two-stage detectors, YOLO processes images through a single neural network, enabling real-time inference while maintaining competitive accuracy. The fundamental innovation lies in its grid-based approach: the input image is divided into an $S \times S$ grid, where each cell predicts bounding boxes with associated confidence scores and class probabilities [48], [49]. This unified architecture significantly reduces computational overhead compared to region proposal-based methods like Faster R-CNN.

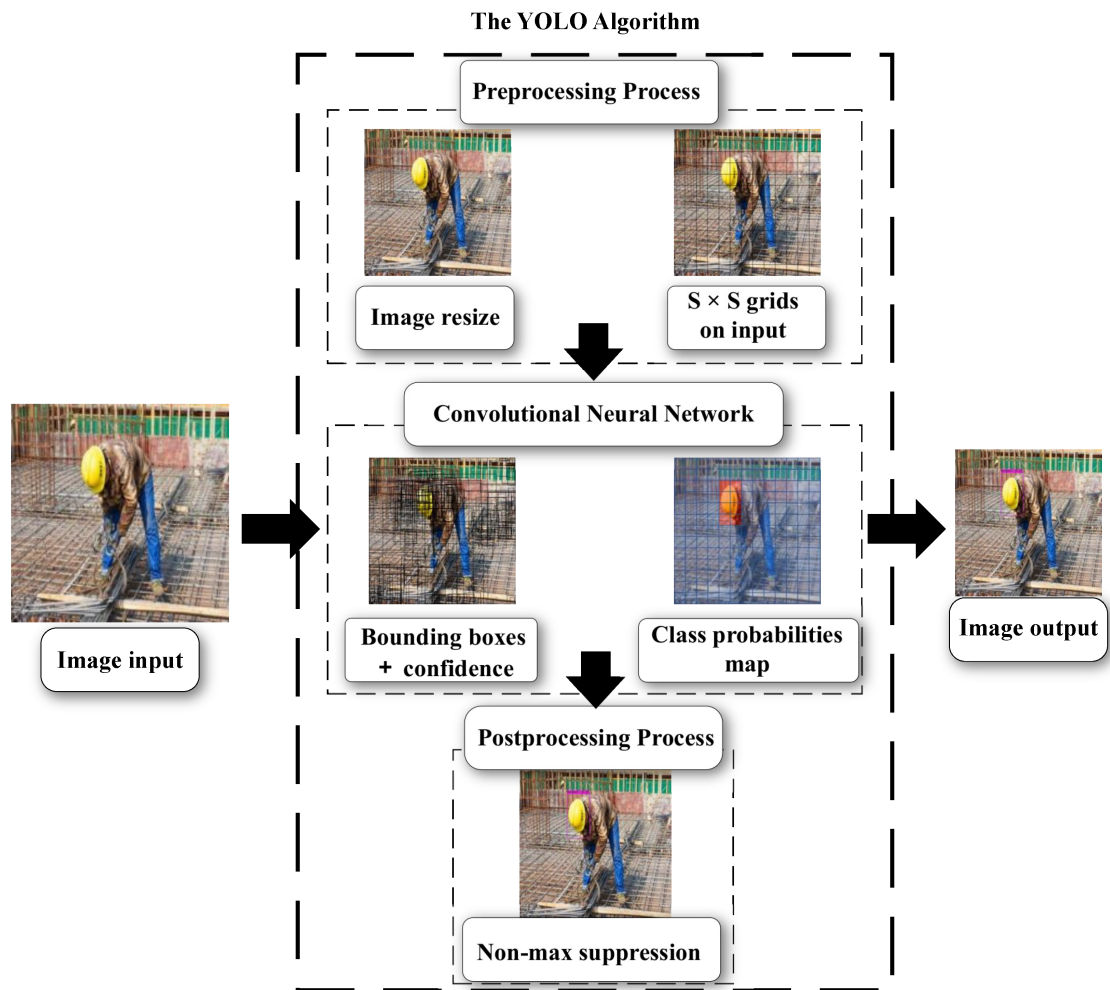


Figure 2.4: Main stages of the YOLO algorithm, from grid division to bounding-box prediction [29]

Architectural Foundations

The original YOLO architecture follows a backbone-neck-head design pattern. The backbone extracts hierarchical features from input images, the neck aggregates multi-scale feature maps, and the head performs the final detection predictions [48]. Early implementations employed simple convolutional networks, but evolution has produced increasingly sophisticated designs.

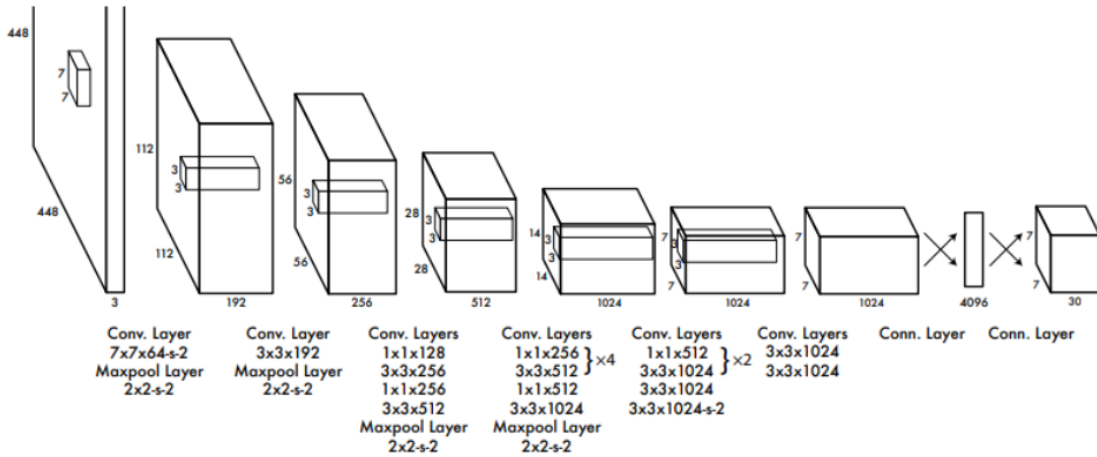


Figure 2.5: Core architecture of the YOLO family showing feature extraction and detection heads [48]

Backbone Evolution

The feature extraction backbone has undergone substantial refinement across YOLO generations. Initial versions used plain convolutional networks, while subsequent iterations introduced residual connections and cross-stage partial networks to improve gradient flow and feature reuse.

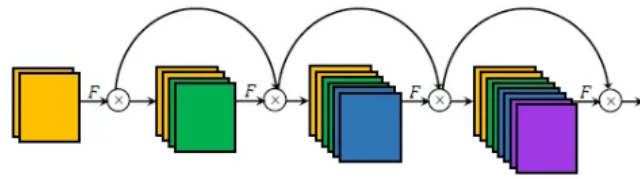
DarkNet Variants: YOLOv2 introduced DarkNet-19 with batch normalization, while YOLOv3 expanded to DarkNet-53 with residual blocks for deeper feature extraction [49], [50]. These backbones balanced accuracy with inference speed.

CSP Architectures: YOLOv4 introduced CSPDarkNet53, which splits feature maps to reduce computational redundancy while maintaining representational power [5]. This Cross-Stage Partial approach became foundational for subsequent variants.

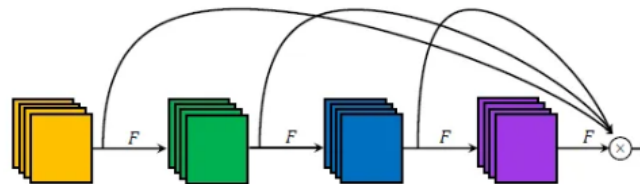
Efficient Designs: Later versions introduced specialized backbones like EfficientRep (YOLOv6), E-ELAN (YOLOv7), and GELAN (YOLOv9) that optimized for hardware efficiency through reparameterization and strategic aggregation [38], [65], [71].

Table 2.2: Structural summary of the Darknet-19 backbone used in earlier YOLO versions [49]

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			



(a) Dense Aggregation (DenseNet)



(b) One-Shot Aggregation (VoVNet)

Figure 2.6: Comparison between Dense Aggregation Network (DAN) and One-Shot Aggregation (OSA), emphasizing feature reuse and computational differences [37]

Table 2.3: Architectural breakdown of the Darknet-53 backbone used in YOLOv3 [50]

Times	Type	Filters	Size/Stride	Output
1	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3/2$	128×128
	Convolutional	32	1×1	128×128
	Convolutional Residual	64	3×3	128×128
2	Convolutional	128	$3 \times 3/2$	64×64
	Convolutional	64	1×1	64×64
	Convolutional	128	3×3	64×64
	Convolutional Residual	128	3×3	64×64
1	Convolutional	256	$3 \times 3/2$	32×32
	Convolutional	128	1×1	32×32
	Convolutional	256	3×3	32×32
	Convolutional Residual	256	3×3	32×32
8	Convolutional	512	$3 \times 3/2$	16×16
	Convolutional	256	1×1	16×16
	Convolutional	512	3×3	16×16
	Convolutional Residual	512	3×3	16×16
4	Convolutional	1024	$3 \times 3/2$	8×8
	Convolutional	512	1×1	8×8
	Convolutional	1024	3×3	8×8
	Convolutional Residual	1024	3×3	8×8
	Avgpool Connected Softmax	1000	Global	1000

The Dense Aggregation Network (DAN) connects each layer to every other layer in a feed-forward fashion, promoting feature reuse and improving gradient flow across the network. In contrast, the One-Shot Aggregation Network (OSAN) combines multi-scale features in a single aggregation step, reducing computational cost while capturing essential contextual information [37].

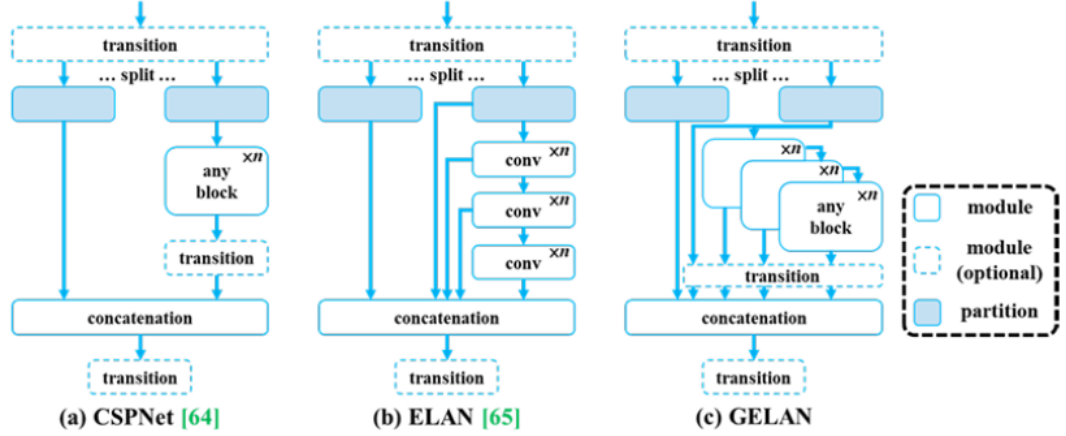


Figure 2.7: Structural comparison of CSPNet, ELAN, and GELAN, illustrating different strategies for gradient flow and feature learning [71]

ELAN (Efficient Layer Aggregation Networks) enhances feature representation by carefully aggregating layers to improve learning capacity without increasing computational cost. GELAN (Gated ELAN) extends ELAN by introducing gating mechanisms to selectively control feature flow, improving information efficiency. CSPNet (Cross Stage Partial Network) splits feature maps into two parts and merges them later, reducing computation while preserving gradient flow and enhancing feature learning [39].

Neck Architecture and Multi-Scale Processing

The neck architecture bridges backbone features with detection heads, enabling effective multi-scale object detection through feature pyramid networks.

FPN and PANet: YOLOv3 introduced Feature Pyramid Networks (FPN) for detecting objects at multiple scales [50]. YOLOv4 enhanced this with Path Aggregation Network (PANet) for improved feature flow between layers [5].

SPP Variants: Spatial Pyramid Pooling (SPP) blocks capture multi-scale contextual information. YOLOv4 used SPP, while YOLOv5 introduced SPPF (Spatial Pyramid Pooling Fast) with sequential rather than parallel pooling for improved efficiency [32].

Advanced Aggregation: YOLOv6’s Rep-PAN and YOLOv8’s C2F modules further refined feature aggregation through reparameterization and efficient cross-stage connections [38], [65].

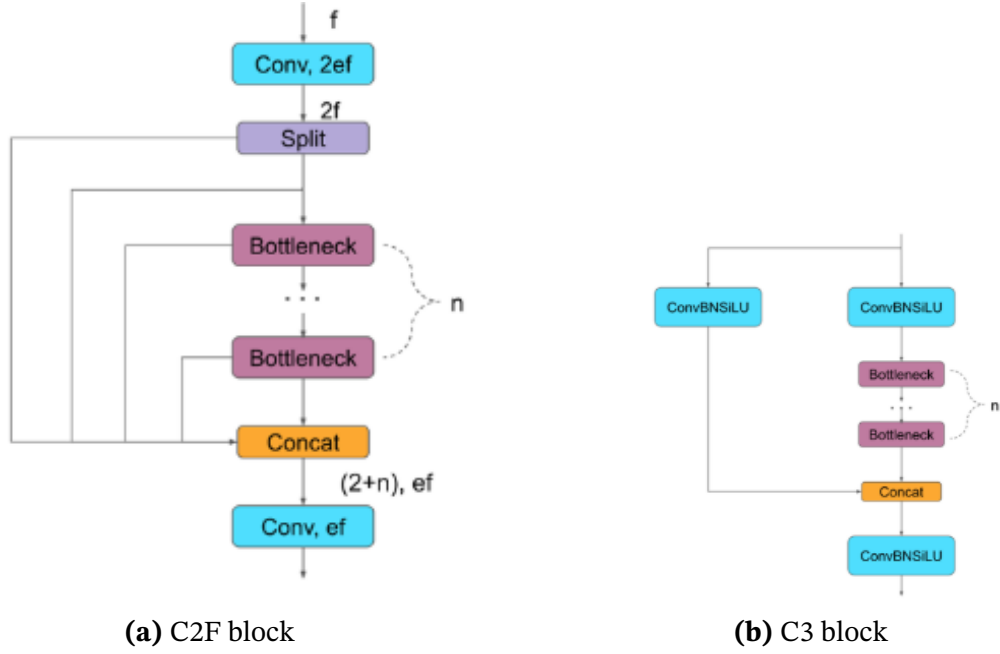


Figure 2.8: Architectural comparison of the C2f block and the earlier C3 block, focusing on feature fusion efficiency [65]

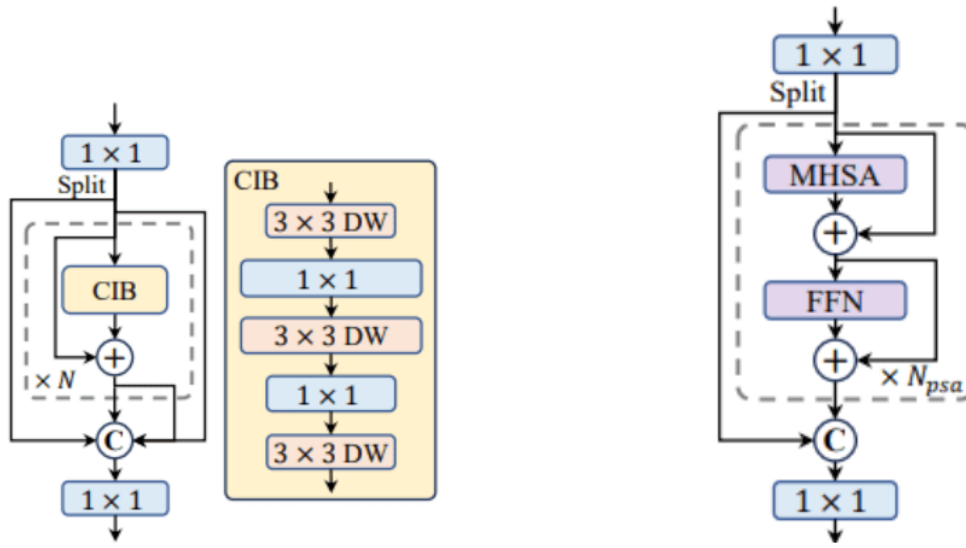
Detection Head Innovations

The detection head converts features into bounding boxes and class predictions, with significant evolution in prediction mechanisms.

Anchor-Based to Anchor-Free: Early YOLO versions used predefined anchor boxes with k-means clustering for dimension priors [49]. Modern variants like YOLOv6 and YOLOv10 transitioned to anchor-free designs that predict object centers directly, simplifying the detection pipeline [38], [70].

NMS-Free Detection: YOLOv10 introduced dual-head prediction with consistent matching metrics, eliminating the need for Non-Maximum Suppression during inference and reducing post-processing overhead [70].

Loss Function Evolution: Bounding box regression progressed from MSE to IoU-based losses (CIoU, DFL) that better capture overlap and aspect ratio alignment [65].



(a) Compact Inverted Block [70]

(b) Partial Self-Attention Block [70]

Figure 2.9: Key components of YOLOv10: (a) Compact Inverted Block and (b) Partial Self-Attention Block, used to boost accuracy while maintaining efficiency [70]

Training Methodologies and Optimization

Training strategies have evolved significantly, with "bags of freebies" improving accuracy without inference cost and "bags of specials" enhancing performance with minimal speed impact [5].

Data Augmentation: Advanced techniques like Mosaic, MixUp, and CutMix create more robust training data, while auto-anchoring adapts priors to specific datasets [5], [32].

Model Scaling: Compound scaling adjusts width, depth, and resolution simultaneously. Model reparameterization combines multiple training branches into a single, efficient structure for inference.[65].

Gradient Optimization: YOLOv9's Programmable Gradient Information preserves complete information flow during backpropagation. It addresses information degradation in deep networks [71].

Recent Architectural Innovations

Recent YOLO versions introduce novel blocks and modules that push performance boundaries while maintaining efficiency.

C3K2 Blocks: YOLOv11 replaces C2f blocks with C3k2 blocks in the backbone and

adds Cross Stage Partial with Spatial Attention (C2PSA) blocks to enhance feature processing with attention mechanisms [33].

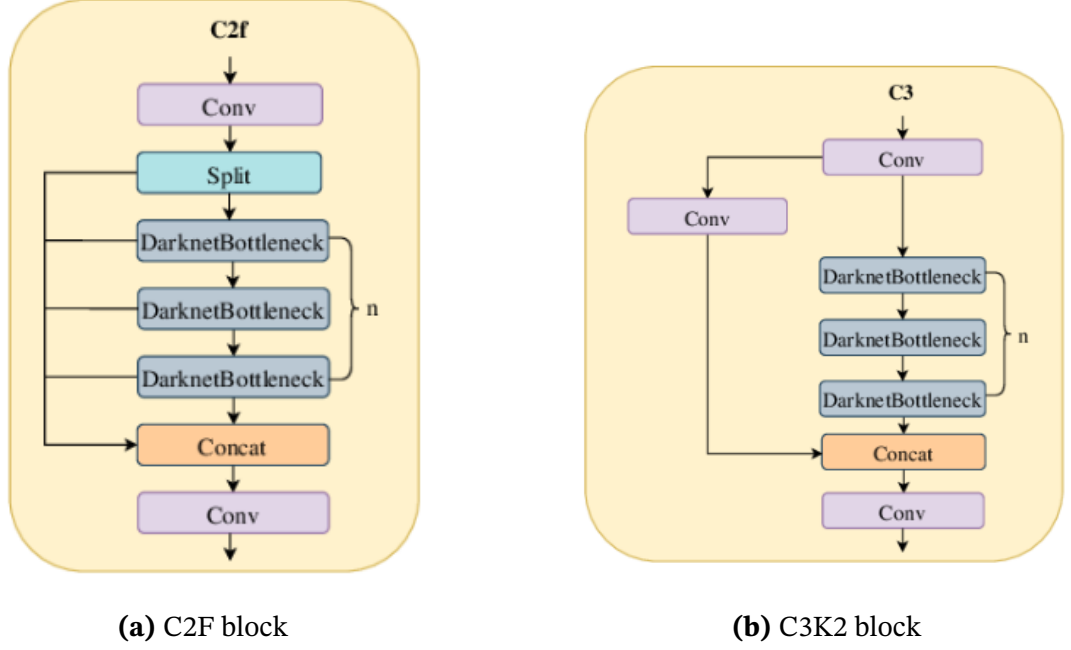


Figure 2.10: Comparison between the C2f block and the C3K2 block, demonstrating kernel expansion and multi-scale enhancement on YOLO architecture [72]

YOLO architectures demonstrate a consistent trajectory toward improved accuracy-speed tradeoffs. Modern variants achieve superior performance on small object detection through enhanced feature aggregation and loss formulations [3]. The evolution from computationally intensive early versions to optimized modern implementations has expanded YOLO’s applicability across embedded systems, mobile devices, and real-time video analysis.

However, further integration of attention mechanisms, efficient transformer modules, and specialized hardware optimization, continuing YOLO’s legacy of balancing detection accuracy with practical deployment efficiency [29].

2.3.3 MHAF-YOLO

Modern YOLO-based detectors often encounter difficulty when dealing with objects that vary widely in scale. Small objects require precise spatial detail, while larger ones demand stronger semantic context. This creates a challenge that standard feature pyramids cannot fully address. MHAF-YOLO is developed specifically to address this challenge by strengthening multi-scale representation while preserving the real-time nature of the YOLO family. The model introduces the Multi-Branch Auxiliary

Feature Pyramid Network (MAFPN), a neck architecture designed to merge shallow texture information with deeper semantic cues through heterogeneous kernels, auxiliary pathways, and multi-level receptive field expansion.

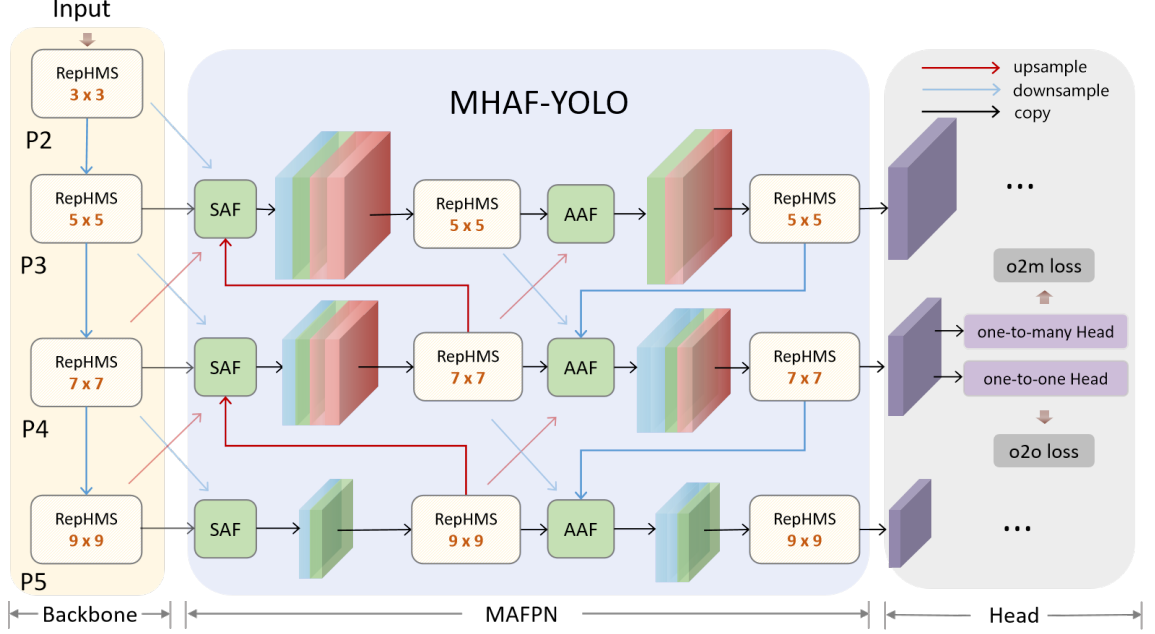


Figure 2.11: Overall architecture of MHAF-YOLO, showing how multi-branch feature aggregation improves detection [77]

Global Adaptive Heterogeneous Flexible Kernel Selection

The core idea behind this architecture is different stages of feature extraction benefit from different receptive field sizes. Smaller kernels capture local patterns critical for fine-grained structure, while larger kernels broaden contextual awareness [12]. Instead of choosing a fixed kernel size for an entire stage, MHAF-YOLO employs a heterogeneous and gradually varying kernel selection strategy across the backbone and the neck. This approach enables the model to adapt to diverse spatial patterns, similar to how attention mechanisms adaptively expand or contract their context in natural language processing. The backbone progressively increases kernel size to broaden semantic coverage, while the neck uses larger heterogeneous kernels to facilitate robust multi-scale fusion.

Multi-Branch Auxiliary Feature Pyramid Network

Traditional Feature Pyramid Networks primarily rely on a top-down flow of information. While effective, this structure can restrict the upward propagation of detailed spatial cues, particularly in scenarios involving small targets. The Multi-Branch Aux-

iliary FPN revisits this limitation by forming richer, bidirectional pathways that allow information to move across shallow and deep layers more freely [82].

By incorporating auxiliary branches that connect both neighboring and non-adjacent feature stages, MAFPN encourages more thorough interaction among features originating at different depths. This reduces the semantic gap between scales, maintains crucial texture information, and provides a stronger foundation for consistent detection across a wide range of object sizes [76].

Superficial Assisted Fusion Shallow features are essential for localizing small objects, yet they often lack stability and may degrade during deep pyramid fusion. The Superficial Assisted Fusion (SAF) module is designed to preserve and refine these early-stage features before they are merged with deeper representations [77]. SAF consolidates contextual cues from adjacent scales, normalizes their contribution, and prepares them for integration within the larger multi-scale structure of MAFPN. By ensuring that shallow information remains intact and meaningful, SAF enhances the network’s sensitivity to fine-grained patterns that would otherwise be lost.

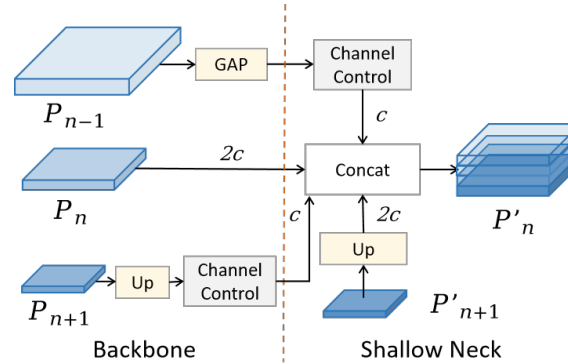


Figure 2.12: Structure of the Superficial Assisted Fusion (SAF) module, designed to preserve shallow spatial details [77]

Advanced Assisted Fusion

While SAF strengthens the shallow pathways, deeper layers require a more comprehensive aggregation strategy that accounts for broader semantic variations. The Advanced Assisted Fusion (AAF) module addresses this by drawing information simultaneously from multiple directions: shallower layers, deeper stages, and lateral neighbors. This dense, multi-path fusion structure enables AAF to enhance medium- and large-object representations while maintaining continuity with the shallow streams established by SAF [77].

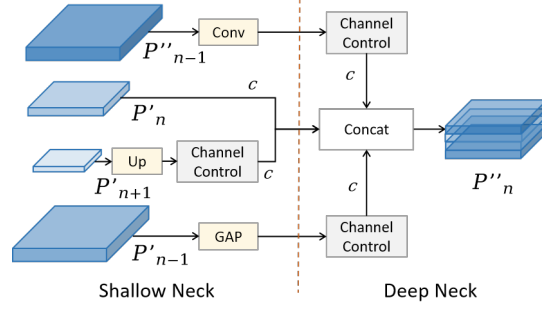


Figure 2.13: Structure of the Advanced Assisted Fusion (AAF) module, aimed at enhancing deeper semantic feature interactions [77]

Re-parameterized Heterogeneous Multi-Scale Module

To balance expressive power with computational efficiency, MHAF-YOLO incorporates the Re-parameterized Heterogeneous Multi-Scale (RepHMS) module. Instead of relying on a single pathway, RepHMS divides the input into several branches, each designed to extract features with progressively increasing depth [77]. Shallow information is propagated forward through cascade connections, ensuring that early structural patterns remain available throughout deeper stages. During inference, the multi-branch structure collapses into an equivalent single-path form through re-parameterization. The process preserves speed without sacrificing representational diversity [76].

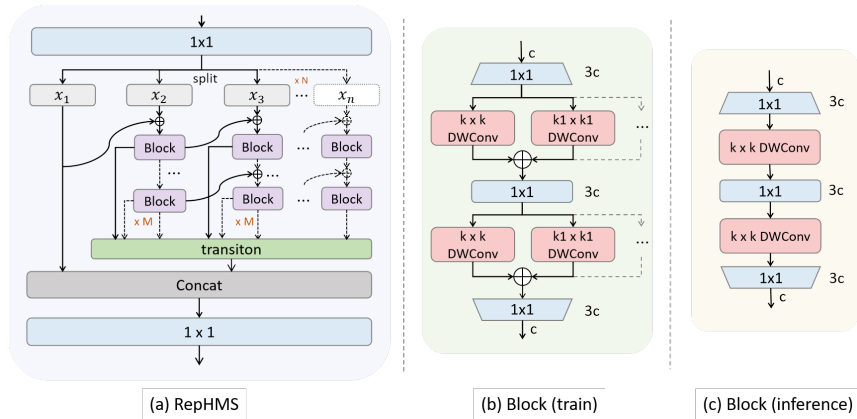


Figure 2.14: Architecture of RepHMS, illustrating hybrid multi-scale representation with re-parameterizable blocks [77]

Re-parameterized Heterogeneous Depthwise Convolution

To further amplify feature diversity, MHAF-YOLO employs heterogeneous depthwise convolutions during training [77]. Each parallel branch uses a different kernel size, enabling the model to simultaneously learn local detail and broader structural patterns [76]. These parallel paths are later fused into a single equivalent kernel via re-parameterization, resulting in a lightweight yet expressive convolutional block. This

mechanism expands the effective receptive field without increasing inference cost, aligning with the model’s goal of retaining real-time performance while strengthening multi-scale capability.

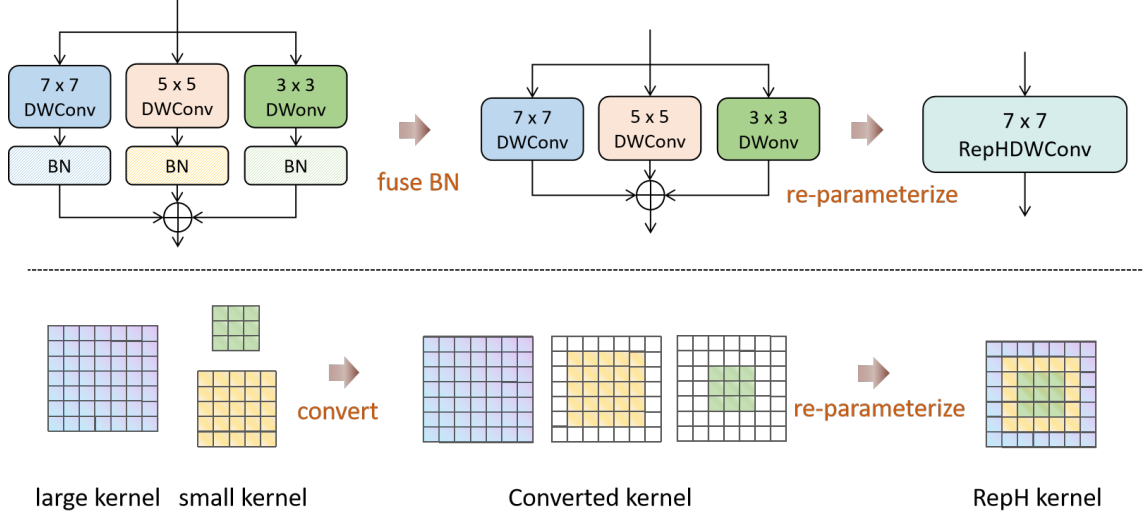


Figure 2.15: Example of re-parameterizing small kernels (3×3, 5×5) into a larger equivalent kernel (7×7) for inference optimization [77]

2.4 Transformer Based Model

2.4.1 Detection Transformer(DETR)

The Detection Transformer (DETR) revolutionized object detection by reframing the task as a direct set-prediction problem [6]. This work replaced traditional multi-stage pipelines involving region proposals, anchor generation, and non-maximum suppression with a transformer-based encoder–decoder architecture. In DETR, a convolutional backbone such as ResNet extracts image features that are flattened and augmented with positional encodings before entering the transformer encoder. On the other hand, a fixed set of learned object queries is processed by the decoder to output bounding boxes and class labels in parallel. The model optimized through a bipartite matching (Hungarian) loss ensuring a one-to-one correspondence between predictions and ground truths [35]. This paradigm eliminates the need for hand-crafted detection components, offering a fully end-to-end training framework that learns global relationships between objects and background.

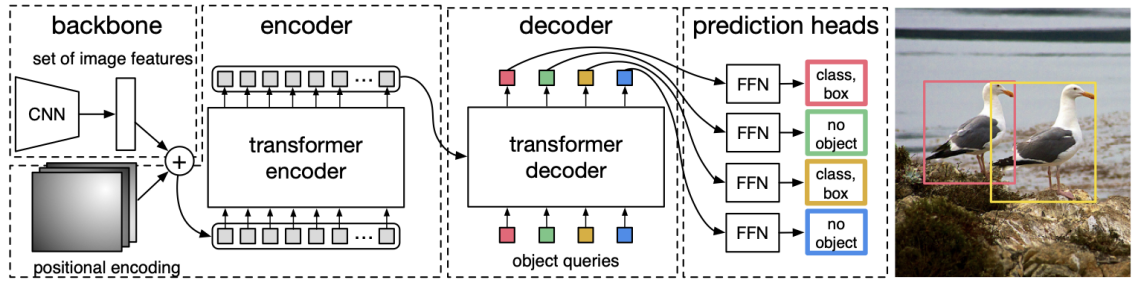


Figure 2.16: Object detection framework of DETR, showing transformer-based end-to-end prediction [6]

However, DETR suffers from several limitations. Firstly, the training convergence is extremely slow. It often requires hundreds of epochs due to the high computational complexity of global attention and the random initialization of object queries [17], [85]. Secondly, it performs poorly on small or densely clustered objects because global self-attention dilutes fine-grained spatial details [56]. Furthermore, the high computational cost limits usage in latency sensitive or embedded applications.

Real-Time Detection Transformer(RT-DETR)

To address the limitations of DETR, Real-Time Detection Transformer (RT-DETR) was proposed [84]. It retains end-to-end philosophy of DETR but introduces architectural refinements that significantly improve speed and efficiency.

RT-DETR employs an efficient hybrid encoder that decouples intra-scale interaction and cross-scale fusion, reducing sequence length and computation. Moreover, an IoU-aware query selection strategy has been employed which enhances query initialization and detection accuracy.

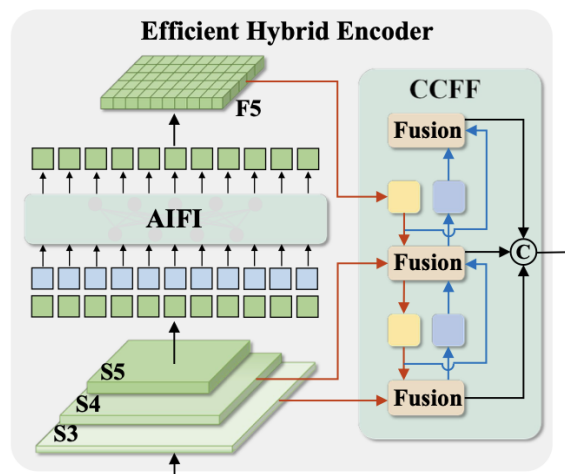


Figure 2.17: Efficient Hybrid Encoder combining CNN and transformer features for improved global-local representation [84]

Furthermore, RT-DETR enables flexible inference by allowing dynamic adjustment of decoder layers to balance speed and accuracy, achieving up to 53.0% AP on COCO val2017 at 114 FPS on a T4 GPU—surpassing YOLO-series detectors in real-time performance [84]. Despite these advancements, RT-DETR is not free from drawbacks. Although its architecture reduces latency, performance on small or distant objects remains suboptimal. The real-time throughput can cause an increase in false positives under complex backgrounds, and the transformer backbone remains too resource-intensive for many mobile or edge devices [56].

Overall, the evolution from DETR to RT-DETR highlights the ongoing pursuit of unifying accuracy and efficiency in transformer-based detectors. Yet, challenges in convergence speed, small-object recognition, fine-scale feature modeling, and hardware optimization remain active research frontiers. For tasks like traffic sign detection and autonomous driving, where speed and lightweight design are key, RT-DETR still needs improvement.

2.5 Publicly Available Traffic Sign Datasets

Traffic sign detection has been a widely researched area, with numerous publicly available datasets developed to address various geographical, environmental, and technical challenges. These datasets differ significantly in scale, diversity, resolution, and annotation quality.

2.5.1 Early Benchmark Datasets

Early traffic sign datasets primarily focused on small-scale, controlled environments, aiming to establish standard benchmarks for detection and classification models.

Belgium Traffic Sign Dataset (BTSD)

The Belgium Traffic Sign Dataset was published in 2011 [67] and contains 145,000 images with 62 types of signs, of which 13,000 are annotated. Image sizes range from 100×100 to 1628×1236 pixels. Unlike highway-focused datasets, it includes signs from smaller roads, increasing the difficulty of detection. The study introduced selective window extraction and sliding window classification methods, proposing a hybrid approach for improved results. It also implemented automatic feature selection and multiview 3D localization for enhanced performance, achieving a detection accuracy of 95.3% and a recognition accuracy of 97.0%.



Figure 2.18: Sample images from the Belgium Traffic Sign Dataset, illustrating category diversity [67]

Swedish Traffic Sign Dataset

Introduced in 2011 [36], the Swedish dataset includes 20,000 frames collected across 350 km of highways and city roads using a 1.3 MP dashboard-mounted camera. Every fifth frame was manually labeled, covering six traffic sign classes. While the dataset provided early insight into real-road scenarios, it suffered from limited environmental diversity and annotation depth. Fourier Descriptors were used to build a star-shaped spatial model for recognition without using a dedicated region detector.



Figure 2.19: Sample images of Swedish Traffic Sign Dataset illustrating situations of Highway [36]

German Traffic Sign Recognition Benchmark (GTSRB)

The German Traffic Sign Recognition Benchmark (GTSRB) was created from 10 hours of video captured in diverse lighting and weather conditions. Manual annotation produced 50,000 images across 43 classes [57]. Frames were cropped with a 10% margin for contextual consistency, and sequences with fewer than 30 images were excluded. Although widely used to train CNNs and SVMs, it suffers from class imbalance and occasional low-quality frames.



Figure 2.20: Sample Signs from German Traffic Sign Recognition Benchmark (GTSRB) [57]

German Traffic Sign Detection Benchmark (GTSDb)

Developed in Bochum, Germany, the GTSDb includes 900 images (600 training and 300 evaluation) featuring 1,206 signs from urban, rural, and highway settings [26]. Captured using a Prosilica GC 1380CH camera, it annotates four sign categories: prohibitory, danger, mandatory, and others. Despite its precise labeling, GTSDb's small scale and limited weather diversity restrict generalization. Traditional methods such as Viola–Jones and HOG-based detectors were originally evaluated using this dataset.



Figure 2.21: Sample images from German Traffic Sign Detection Benchmark (GTSDDB) simulating real road side situations [26]

2.5.2 Large-Scale and Diverse Datasets

To address the limitations of small, geographically constrained datasets, several large-scale collections have been developed, emphasizing diversity in conditions, camera setups, and annotation standards.

Mapillary Traffic Sign Dataset (MTSD)

The Mapillary Traffic Sign Dataset (MTSD) [14] introduced global diversity by collecting 100,000 images across multiple continents with 313 traffic sign classes. Images were gathered from various camera types to ensure heterogeneity in resolution and quality. Annotation involved three stages: image approval, sign localization, and classification using Wikimedia Commons templates. Despite its global coverage, class imbalance and annotation complexity persist as challenges. Transfer learning experiments demonstrated that MTSD enhanced model convergence compared to earlier datasets.



Figure 2.22: Sample scenes from the Mapillary Traffic Sign Dataset (MTSD), highlighting real-world complexity [14]

LISA Traffic Sign Dataset

The LISA dataset [46], published in 2012, contains 6,610 images and 7,855 annotations across 49 U.S. traffic sign classes, derived from annotated video sequences. While it introduced temporal consistency by linking frames to video sources, the dataset lacks environmental diversity, including limited weather and lighting conditions. Nevertheless, it provided an important benchmark for early vision-based autonomous driving research.



Figure 2.23: Sample images from LISA Traffic Sign Dataset [46]

TT100K Dataset

The TT100K dataset [86], constructed from Tencent Street View imagery, comprises 100,000 images from 300 Chinese cities, capturing urban variability. Traffic signs were annotated manually using bounding boxes, polygons, and ellipses to handle occlusion and distortion. Although extensive in size, TT100K contains limited weather diversity and class imbalance. Experimental results showed Faster R-CNN outperforming earlier CNN architectures on this dataset.

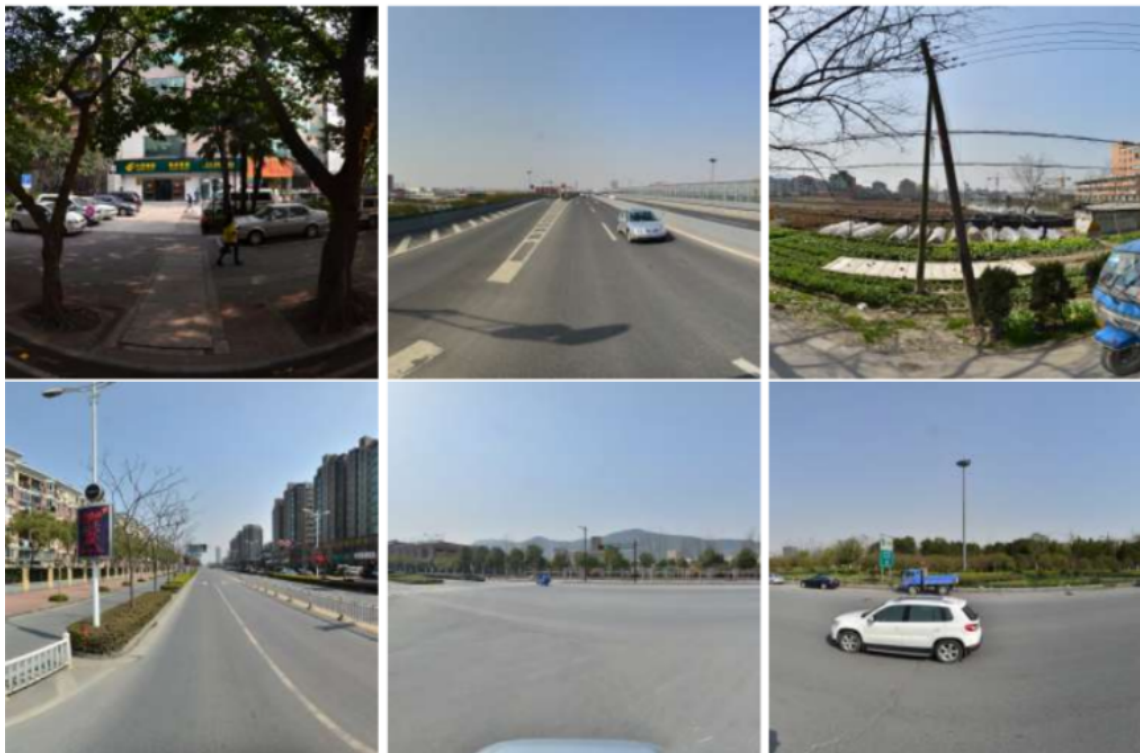


Figure 2.24: Sample images from the TT100K Dataset, showing dense traffic sign clutter [86]

CCTSD2021 Dataset

The Chinese Traffic Sign Database (CCTSD2021) [81] was curated from 423 driving videos, yielding 5,260 images annotated in XML format using LabelImg. It maintains balanced representation across sign sizes and weather conditions. Models such as YOLOv3 to YOLOv5, Faster R-CNN, RetinaNet, and SSD were benchmarked, revealing that two-stage detectors achieved higher accuracy, whereas one-stage models offered superior inference speed.



Figure 2.25: Example images from the CCTSDB2021 Dataset displaying Chinese road conditions [81]

2.5.3 Regionally Specialized Datasets

Recent research emphasizes localized datasets that capture the environmental, linguistic, and cultural characteristics of regional traffic systems. These datasets support developing models that generalize across non-Western traffic environments.

Malaysian Traffic Sign Dataset

Introduced in 2016 [43], this dataset comprises 2,096 images spanning 66 classes collected from various sources including Google Maps, mobile phones, and digital cameras. It was among the first to include rainy and nighttime scenes, though limited in number. Manual annotation via MATLAB R2015a produced separate ground truth files for recognition and detection tasks. Distinctive regional characteristics, such as color variance (52% yellow signs), highlight its uniqueness within Southeast Asian contexts.



Figure 2.26: Malaysian Traffic Sign Dataset examples illustrating local sign variations [43]

Indian Traffic Sign Dataset

The Indian dataset [44] contains 6,480 images with 7,056 traffic signs across 100 categories, collected from various urban and rural locations. Roughly 97% of signs are visible, captured under diverse lighting conditions. Using Mask R-CNN and its optimized variant, the study achieved improved detection precision, though challenges such as scale and orientation variation remain.



Figure 2.27: Sample images from Indian Traffic Sign Dataset [44]

Bangladeshi Traffic Sign Detection Benchmark (BTSD)

The BTSD dataset [3] includes 2,986 images with 3,063 annotated signs across 15 categories, collected from urban, rural, and highway regions. It captures real-world complexities such as blur, occlusion, and variable lighting. Training and test splits were separated by video sources to ensure independence. Evaluation results showed YOLOv7 achieving the highest mAP score among compared models.



Figure 2.28: Example scenes from Bangladeshi Traffic Sign Detection Benchmark (BTSD) [3]

BanTrafficNet Dataset

BanTrafficNet [69] consists of 6,400 images across 48 Bangladeshi traffic sign classes, totaling 24,000 samples after augmentation. Although it lacks challenging environmental conditions, artificial augmentations such as blur, grayscale, and brightness variation simulate real-world noise. A lightweight CNN with only 4.6 million parameters achieved an ROC-AUC of 99.97%, outperforming larger architectures such as ResNet50 and InceptionV3.



Figure 2.29: Sample images from BanTrafficNet showing lack of real driving circumstances [69]

2.5.4 Summary and Observations

Overall, the evolution of traffic sign datasets reflects a progression from small, region-specific collections to large-scale, multi-environment repositories. While early datasets like GTSRB and BTSD established foundational benchmarks, more recent datasets such as MTSD and CCTSDB2021 emphasize environmental diversity and global representation. Nevertheless, challenges such as class imbalance, limited weather variation, and dataset bias persist. It motivates the creation of regionally adaptive datasets like BanTrafficNet and BTSD for improving real-world detection robustness.

2.6 Existing Techniques for Traffic Sign Detection

In recent years, notable progress has been made in the development of the Traffic Sign Dataset and different frameworks have been proposed on top of that for more robust detection. However, several limitations persist in the publicly available traffic sign datasets. These include significant class imbalance, a scarcity of nighttime images, and the presence of samples that do not accurately reflect real-world conditions [63]. The GLARE dataset presents the first traffic sign dataset featuring realistic sun glare conditions with 2,157 annotated images across 41 classes, supporting both detection and recognition tasks [23]. Models trained on this dataset, particularly Faster R-CNN with SwinT backbone, achieved higher mAP scores compared to those trained on the LISA or combined datasets [23]. When examining newly proposed datasets on traffic signs alongside their corresponding models, it becomes evident that they often suffer from limited robustness and reduced accuracy, particularly in detecting small or underrepresented traffic signs.

The study conducted by Mogelmoose et.al. proposed a three-step framework for traffic sign detection, comprising segmentation, feature extraction and modeling, and detection [46]. Unlike traditional methods that rely solely on color or shape cues, their approach integrated multiple features to enhance detection accuracy. Despite that the proposed approach exhibited several limitations, including inconsistent evaluation metrics, insufficient contextual modeling, and limited robustness under challenging conditions such as varying illumination, occlusion, and adverse weather. Rehman et. al. employed Aggregated Channel Features (ACF) and Discriminative Patches (D-patches) to enhance traffic sign detection under occlusion [51]. However, the approach was computationally expensive and showed reduced effectiveness for signs undergoing significant deformation. Xu et. al. proposed MSER (Maximally Stable Extremal Regions) and Adaptive Color Threshold Segmentation with Red-Blue color normalization to enhance traffic sign detection through symmetry analysis [75]. Still, the method was computationally intensive, relied solely on the controlled GTSDB dataset, and showed reduced robustness under color inconsistencies, occlusion, and irregular sign shapes. During this time, we have seen many machine learning based frameworks being proposed for Traffic Sign Detection tasks. Chen et. al. employed a combination of Discriminative Adaboost and Support Vector Regression (SVR) for traffic sign detection, integrating visual saliency and spatial cues for improved precision [8]. However, the method struggled with lighting variations, motion blur, and perspective distortions, while dataset imbalance and redundancy limited its generalization in real-world scenarios. Temel et. al. utilized real-world images from

the BelgiumTS dataset and synthesized sequences from Unreal Engine 4 to benchmark traffic sign detection under challenging environmental conditions [63]. However, existing algorithms struggled with adverse conditions, and many prior datasets lacked comprehensive evaluation against weather, lighting, and occlusion variations. Changzhen et al. employed Faster R-CNN with pre-trained CNN models (VGG16, VGG_CNN_M_1024, ZF) on an augmented Chinese traffic sign dataset including motion blur and brightness variations [7]. Despite achieving high accuracy, the dataset was geographically limited to Beijing, lacked full diversity, and generalization to other real-world conditions remained uncertain.

Furthermore, Serna et. al. leveraged the European Traffic Sign Dataset (ETSD), combining multiple public datasets and newly captured images, and trained five CNN architectures for classification [54]. Nonetheless, the dataset was unbalanced, lacked extreme environmental conditions such as nighttime or severe weather, and some classes were underrepresented. Therefore, it has limited robustness and generalizability. The CTSD dataset [83] introduces 2,205 real-world images containing 3,755 traffic signs captured from online videos, driving recorders, and volunteer recordings, annotated with 153 subclasses and rich metadata including weather, lighting, occlusion, distance, color fading, and viewpoint. Baseline methods, including Haar+AdaBoost, HOG+SVM, and deep-learning approaches such as OverFeat, Faster R-CNN, YOLOv2, and SSD, were evaluated to benchmark performance under complex scenarios. However, the study did not include the advance neural network based models. Another large-scale study conducted by Tabernik et. al. [59] employed an improved Mask R-CNN with Online Hard Example Mining (OHEM), sample weighting, and adjusted region pass-through to detect traffic signs. Geometric and appearance-based data augmentations were applied to simulate real-world variations and address intra-category variability. Still, the dataset excludes small objects, lacks nighttime images, and contains mostly easily detectable signs. As a result performance on challenging scenarios had not been evaluated. Further, Temel et. al. [64] analyzed 896,000 frames from the BelgiumTS dataset, annotated with sign types, bounding boxes, and 12 challenging conditions, and evaluated deep-learning models including U-Net, ResNet, VGG, and GoogleNet to quantify the impact of environmental factors. While this study provides valuable insights into robustness under varying conditions and employs spectral analysis to assess effects, the automated replication of challenges is not fully accurate, and some scenarios such as snow are poorly represented, reflecting the inherent limitations of the underlying dataset. Liu et al. [42] proposed a Deconvolution Region-based Convolutional Neural Network (DR-CNN) that leverages a multi-scale fused feature map and a two-stage adaptive classification loss to enhance the detection of small traf-

fic signs in high-resolution images from the Tsinghua-Tencent 100K dataset. While the model demonstrates improved performance in controlled settings, its reliance on very high-resolution images (2048×2048) and lack of evaluation in real-time scenarios limit its practical applicability and generalization to more diverse, real-world conditions. European urban traffic sign study [55] extended the GTSDB and ETSD urban subsets to enable improved detection and classification of both symbol and text-based signs using a Mask R-CNN framework. Despite these enhancements, the study is limited by insufficient representation of text-based signs, constrained intra-class variability, geographic specificity, and the disjoint treatment of detection and classification tasks, which may restrict the generalizability and robustness of the proposed approach. Ayachi et. al. [4] collected images from Chinese roads with 73 classes and used MobileNet for real-time detection. But mobileNet has difficulty detecting small signs in cluttered scenes. Mijic et. al.[45] used video frames from Osijek, Croatia, annotated with OpenLabeling, and trained YOLOv3. However, it had limited geographic scope, class imbalance, and small number of sign classes. Therefore, it is not suitable for real world application. Kilic et. al. [34] used images from moving vehicles and used Faster R-CNN Inception V2 for detection. However, limitations were missing sign types, insufficient training samples, and long training time due to large image sizes. Hybrid ANN recognition study [31] combined GTSRB and TSRD datasets and used nine individual ANNs with RGB, HOG, and LBP features, although uneven class distributions in the source datasets could bias the model despite augmentation efforts. The study [18] proposes an improved Faster R-CNN-based traffic sign detection algorithm aimed at enhancing detection performance for small traffic signs, images captured under low-light conditions, and those affected by adverse weather. The model was trained and evaluated on the TT100K dataset, which contains 100,000 images spanning 221 traffic sign classes, with annotations manually created using LabelImg and stored in XML format. While the improved Faster R-CNN demonstrates notable gains in precision and robustness for small and low-visibility signs, it still exhibits limitations in challenging conditions such as rain and low sunlight, and it struggles to accurately differentiate between visually similar traffic signs, highlighting areas for further refinement in real-world deployments.

In a performance evaluation of the dataset, Alamgir et al. demonstrated that YOLOv5x achieved superior results compared to other YOLO variants available at the time [1]. Nevertheless, their findings also revealed that none of the models surpassed a mean Average Precision (mAP) score of 0.3 [1]. In the last few years, many variants of YOLO algorithms have been applied to detect traffic signs.

In [78], Yang et al. improved YOLOv3 by incorporating the inception architecture in

the backbone. Although YOLOv3 works well with larger targets due to its deeper structure, it fails to perform well for smaller targets. Therefore, Yang et al. proposed an inception architecture and color enhancements to YOLOv3 to enrich features to incorporate the shallow network, which showed significant improvement on the TT100K dataset over the base YOLOv3 model. Although the proposed architecture has improved the results of YOLOv3, later models outperform it as well. Also, Dewi et al. consolidated the SPP Layer within YOLOv4 tiny and showed around 99% accuracy on the Taiwan Dataset [10]. Despite outperforming YOLOv4, the results were limited by the dataset, which lacked sufficient complexity to truly challenge the model.

In [25], Han et al. replaced the original convolution block of YOLOv5s with Space to Depth Convolution (SPD Convolution) and changed the head with a decoupled head. They also integrated CAM (Context Augmentation Module) into the architecture. The idea of a decoupled head was introduced first in YOLOX, where one head is responsible for the classification task and the other is responsible for the regression task [20]. Han et al. showed significant improvements over YOLOv5 on small object detection. The Space-to-Depth (SPD) module increases the number of channels, which can lead to extra computation without adding much useful information. At the same time, it rearranges spatial features, which can cause the model to lose important details about where things are in the image. Also, Chu et al. used a similar decoupled head but added self-attention in the backbone of YOLOv5, which allows the model to detect small objects by focusing on global context information [9]. It increased the mAP score of YOLOv5 by 6.8% on the TT100K dataset. However, self-attention may overlook regional context by giving more priority to global context.

In [41], Liu et al. proposed ETSR-YOLO, which includes Coordinate Attention in the backbone, embedding both coordinate information and generating coordinate attention. The purpose of this attention block is to suppress noise interference and improve learning. Moreover, they added a ConvNeXt block next, which uses depthwise separable convolution. ETSR-YOLO showed 6.6% improvement over the base YOLOv5s model on the TT100K dataset and 1.2% improvement on the CCTSDB2021 dataset [41]. Unfortunately, the model failed to detect small objects properly because it does not allow multiscale targets to reach a balanced state. Also, pixel features of small targets are very weak and can interfere with other objects.

We have seen improvements in the later YOLO models for Traffic Sign Detection. In [13], Du et al. added an SPD module to integrate spatial information into the depth channel in the YOLOv8 model. They further added Select Kernel Attention (SK Attention) in the neck architecture and the WIoUv3 loss function to optimize. They

showed 3.2% improvement in the CCTSDB dataset and 5.1% improvement over the base YOLOv8 model. Yang et al. proposed the MHAF-YOLO model, which replaced FPN with the Multi-branch Auxiliary Feature Pyramid Network, and also integrated a Reparameterized Heterogeneous Multiscale Module and Global Heterogeneous Flexible Kernel Selection to enhance feature fusion [77]. Their model showed 48.9% AP on COCO with just 7.1M parameters, which is a reduction of 24.4% compared to YOLOv11. However, most of these improvements were demonstrated on datasets that differ significantly from Bangladeshi conditions. Traffic signs in Bangladesh present unique challenges in terms of design, placement, and visibility, which are not well represented in widely used datasets like TT100K or CCTSDB.

Substantial advancements have been made in traffic sign detection through the development of diverse datasets and increasingly sophisticated models. Still, significant challenges remain which require more robust approach. Existing datasets often lack comprehensive environmental diversity, such as varying weather, illumination, and geographic conditions, which limits the generalization of trained models to real-world scenarios. Similarly, despite architectural innovations, from region-based frameworks like Faster R-CNN to attention-augmented and decoupled-head variants of YOLO, most models continue to struggle with small, occluded, and underrepresented traffic signs. Moreover, many studies rely on datasets that do not reflect the complex visual and structural variations found in developing regions where differences in sign design, color fading, and placement introduce additional detection difficulties. Therefore, there is a clear need for region-specific, diverse, and contextually rich datasets, along with lightweight yet robust detection models capable of maintaining accuracy across diverse environmental and geographic conditions.

Chapter 3

Proposed Methodology

3.1 Overview of Proposed Architecture

One stage object detectors like YOLO (You Only Look Once) comprised of 3 main components: backbone, neck, and head. In Mhaf-YOLO [77], a new variant of neck is introduced due to the architectural weakness of Path Aggregation Feature Pyramid Network which struggles in spatial variations. In our proposed architecture, input images pass through a Squeeze and Excitation Block (SE) which amplifies the important channels and suppress the unimportant ones. Therefore, placing an SE block before the backbone helps to get better channel discriminability, reduced redundancy, and improved SNR (Signal to Noise Ratio). This allows the backbone layers to operate more efficiently and focus their representational power on informative, task-relevant features.

After passing through the SE block, the feature representations are forwarded into the backbone, which consists of four hierarchical stages: P2, P3, P4, and P5. Since the conventional PAFPN is limited in its ability to transfer rich semantic information from deeper layers to the spatially detailed shallow layers, the Multi-Branch Auxiliary FPN (MAFPN) is integrated into the neck to address this issue.

In the initial bottom-up pathway, the SAF module gathers multi-scale features from the backbone and performs an early stage of assisted fusion within the shallow portion of the neck. In parallel, the AAF module establishes a secondary top-down pathway with dense interconnections, enabling it to collect gradient information from deeper layers and guide the detection head toward producing more diverse outputs across three resolutions.

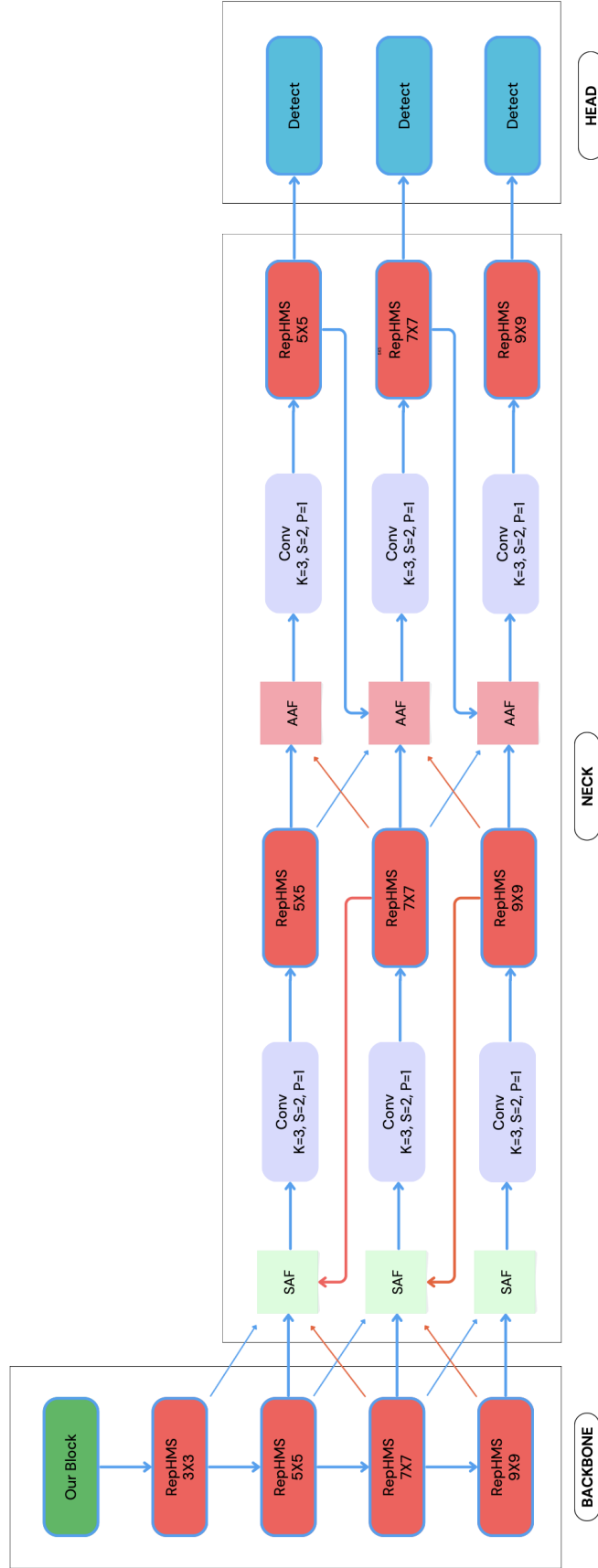


Figure 3.1: Overview of the Proposed Architecture

Both SAF and AAF employ the RepHMS module for feature extraction. Through its integrated GHFKS mechanism, RepHMS adaptively adjusts convolutional kernel sizes, allowing the receptive field to flexibly expand or contract depending on the complexity of the features. Functionally, the SAF block operates between the backbone and the shallower neck layers, while the AAF block continues the refinement process from the shallow neck to its deeper stages.

Finally, the detection heads utilize the fused feature maps at multiple scales to predict object bounding boxes and their corresponding categories. After that, loss is computed from these predictions and used for backpropagation.

3.2 Multi-Branch Auxiliary FPN

The Multi-Branch Auxiliary Feature Pyramid Network (MAFPN) enhances traditional FPNs by introducing bidirectional and multi-path interactions that jointly preserve fine-grained spatial cues and high-level semantics [76]. MAFPN operates through two key modules.

Superficial Assisted Fusion (SAF): SAF enriches shallow spatial features using complementary deep semantics, improving localization of small objects [76].

Let $P_{n-1}, P_n, P_{n+1} \in \mathbb{R}^{H \times W \times C}$ denote feature maps at adjacent resolutions. The SAF output is

$$P'_n = \text{concat}(\delta(\text{GAP}(P_{n-1})), P_n, \delta(\text{Conv}(U(P_{n+1}))), U(P'_{n+1})), \quad (3.1)$$

where $U(\cdot)$ is upsampling, GAP is global average pooling, and δ is the SiLU activation. Channel alignment is regulated through 1×1 convolutions for stable fusion.

Advanced Assisted Fusion (AAF): AAF strengthens cross-scale communication by combining features from shallow high-resolution, shallow low-resolution, sibling, and preceding layers. Its output is defined as

$$P''_n = \text{concat}(\delta(\text{Down}(P'_{n-1})), \delta(\text{GAP}(P''_{n-1})), P'_n, \delta(\text{Conv}(U(P'_{n+1}))))), \quad (3.2)$$

yielding a more comprehensive multi-scale representation for robust detection [76].

3.3 Re-parameterized Heterogeneous Multi-Scale Module

The Re-parameterized Heterogeneous Multi-Scale (RepHMS) module is designed to enhance feature diversity while maintaining computational efficiency, particularly in dense detection scenarios where objects vary widely in scale and appearance. Traditional multi-branch designs often limit representational capacity due to homogeneous branch structures or expensive parallel convolutions [76]. RepHMS addresses these limitations through a heterogeneous, cascade-aware architecture that progressively enriches the representation space [77].

The module begins with a 1×1 convolution that compresses and reorganizes the input features, ensuring that subsequent operations remain lightweight. The transformed feature map is then partitioned into N parallel branches. The first branch serves as an identity-preserving pathway, allowing the network to retain the original low-level information crucial for stable gradient propagation. The remaining $(N - 1)$ branches each contain a sequence of M heterogeneous blocks designed to extract progressively deeper and more discriminative features [76], [77].

Within each branch, the blocks are connected in a cascaded manner, enabling the output of one block to act as the input for the next. This progressive design allows early-stage features to be reused and refined in later stages instead of being overridden, which aligns with the philosophy of ELAN-style feature preservation. Importantly, the output of every block in the cascade is retained rather than discarded. This ensures that the final feature set captures a rich spectrum of information, ranging from shallow details to highly abstract patterns [76], [77].

After the cascaded processing, the outputs from all blocks across all branches are concatenated along the channel dimension. This aggregated representation integrates heterogeneous features extracted at different depths and receptive fields, significantly boosting multi-scale expressiveness. A concluding 1×1 convolution then fuses and projects the concatenated tensor back into a compact form suitable for downstream detection heads.

The parameters N and M govern the diversity and depth of feature extraction, making the module highly flexible. Larger values encourage richer hierarchical learning, while smaller values preserve efficiency for real-time applications. Through this re-parameterized and heterogeneous design, the RepHMS module achieves a balanced trade-off between accuracy and computational cost, offering substantial improvements

in multi-scale representation without introducing unnecessary overhead [76], [77].

3.4 Re-parameterized Heterogeneous Depthwise Convolution

The Re-parameterized Heterogeneous Depthwise Convolution (RepHDWConv) is designed to simultaneously increase the effective receptive field and improve representational capacity without introducing additional inference cost. Traditional depthwise convolutions with a single kernel size often limit the network’s ability to capture features at multiple spatial scales [76], [77]. RepHDWConv addresses this by employing multiple depthwise convolution kernels of varying sizes during training, allowing the model to learn rich, multi-scale feature representations.

During the training phase, each depthwise convolution kernel is independently followed by a Batch Normalization (BN) layer with learnable parameters $\mu, \sigma, \gamma, \beta$. The BN layers stabilize training by normalizing the feature distribution and introducing scale and shift transformations [76], [77].

To enable efficient inference, the kernels and BN parameters are fused into an equivalent single kernel and bias as follows:

$$W_{\text{fused}} = \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}} \cdot \text{RepHDW}, \quad B_{\text{fused}} = \frac{\gamma(-\mu)}{\sqrt{\sigma^2 + \epsilon}} + \beta, \quad (3.3)$$

where ϵ is a small constant for numerical stability. This fusion ensures that the influence of BN is directly integrated into the convolutional weights, eliminating the need for separate normalization during inference [76], [77].

After training, smaller kernels are padded to match the spatial size of the largest kernel. This allows all heterogeneous kernels to be re-parameterized into a single equivalent depthwise convolution, thereby enabling efficient inference without sacrificing the multi-scale benefits learned during training.

The resulting output can be expressed as:

$$O = I \otimes K_{2n-1} + B_{2n-1} + \sum_{i=1}^m (I \otimes K_{2n-(2i+1)} + B_{2n-(2i+1)}), \quad (3.4)$$

where $n \geq 3$, m is the largest integer satisfying $2n - (2m + 1) \geq 3$, I denotes the input feature map, and $\otimes$ represents depthwise convolution. This formulation ensures that

the output incorporates information from all kernel sizes, effectively capturing both local fine-grained features and broader contextual patterns [76], [77].

The primary advantage of RepHDWConv lies in its heterogeneous design: by leveraging multiple kernel sizes during training, the network gains the capacity to model complex spatial relationships without increasing the runtime complexity during inference. This approach combines the representational richness of multi-scale convolutions with the efficiency of a single depthwise kernel at deployment, making it particularly suitable for real-time applications and lightweight architectures.

3.5 Squeeze-and-Excitation (SE) Module

Convolutional Neural Networks (CNNs) have achieved outstanding success in computer vision tasks due to their ability to capture local spatial features through convolutional filters. However, conventional convolutions treat all feature channels equally, ignoring the fact that different channels carry different levels of semantic importance. To address this limitation, Hu et. al. [27] introduced the Squeeze-and-Excitation (SE) block, a lightweight yet powerful attention mechanism that explicitly models channel interdependencies and adaptively recalibrates channel-wise feature responses.

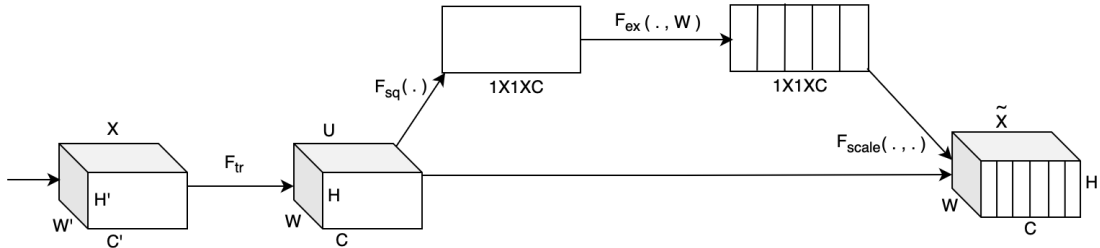


Figure 3.2: Illustration of the Squeeze-and-Excitation (SE) process used for channel-wise feature recalibration

The SE block is mathematically defined as follows. Consider an intermediate feature tensor

$$X \in \mathbb{R}^{B \times C \times H \times W}, \quad (3.5)$$

where B denotes the batch size, C the number of channels, and $H \times W$ the spatial resolution. The SE operation consists of three sequential steps: squeeze, excitation, and recalibration.

In the squeeze operation, a global average pooling (GAP) is applied to aggregate spatial information of each channel into a scalar descriptor:

$$z_c = \frac{1}{H \cdot W} \sum_{i=1}^H \sum_{j=1}^W X_c(i, j), \quad z \in \mathbb{R}^C. \quad (3.6)$$

In the excitation step, the vector z passes through two fully connected layers that form a bottleneck structure:

$$s = \sigma(W_2 \delta(W_1 z)), \quad (3.7)$$

where $W_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ reduces the channel dimensionality, $W_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ restores it, $\delta(\cdot)$ is a nonlinear activation function (e.g., ReLU or SiLU), and $\sigma(\cdot)$ is the sigmoid activation ensuring $s \in (0, 1)^C$.

In the recalibration step, the learned channel weights modulate the original feature maps by channel-wise multiplication:

$$\tilde{X}_c = s_c \cdot X_c, \quad \forall c \in \{1, \dots, C\}, \quad (3.8)$$

producing a recalibrated feature tensor $\tilde{X} \in \mathbb{R}^{B \times C \times H \times W}$.

This entire procedure can be summarized as:

$$z_c = \frac{1}{H \cdot W} \sum_{i=1}^H \sum_{j=1}^W X_c(i, j), \quad (3.9)$$

$$s = \sigma(W_2 \delta(W_1 z)), \quad (3.10)$$

$$\tilde{X}_c = s_c \cdot X_c. \quad (3.11)$$

The computational cost introduced by the SE block is $\mathcal{O}\left(\frac{2C^2}{r}\right)$, which is negligible for typical reduction ratios such as $r = 8$ or $r = 16$, especially when compared to convolutional operations. Despite its simplicity, the SE block significantly enhances feature representation by modeling global channel dependencies.

3.6 SE-Enhanced Bottleneck Layer

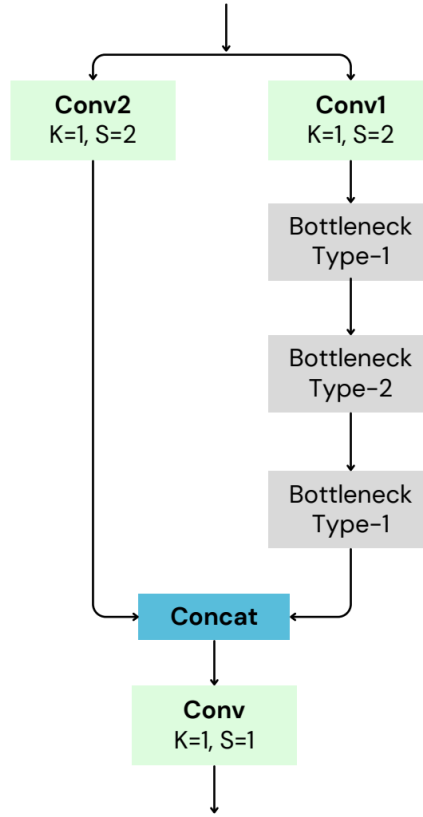


Figure 3.3: High-level overview of the proposed block architecture

The standard initial convolutional stage of MHAFF-YOLO was a simple convolutional block with 3x3 kernel [77]. In order to improve low-level feature encoding and introduce adaptive channel recalibration at the entry stage, it is replaced with the SE-enhanced Bottleneck. Instead of a single fixed kernel size bottleneck, we have incorporated three bottlenecks of two types in a sequence.

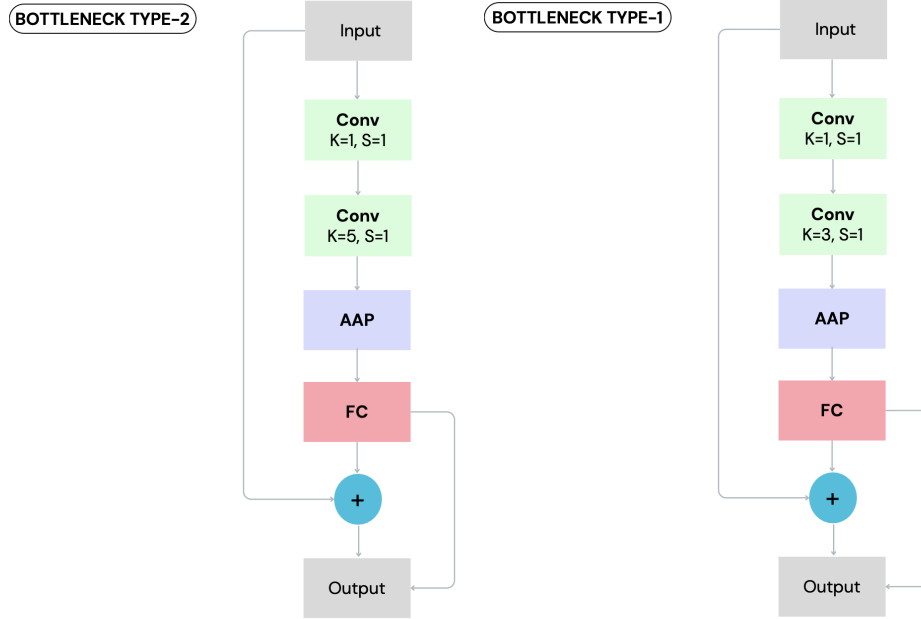


Figure 3.4: Comparison of different bottleneck types and their feature transformation patterns

1. **Bottleneck Type1:** This bottleneck incorporates a 3x3 convolution followed by a depthwise convolution. After the channel manipulation from depthwise convolution, 3x3 Convolution captures local features with short-range spatial dependencies. Finally, Adaptive Average Pooling(AAP) ensures scale and input size variance. Finally Fully Connected Layer compact descriptors for final boundary.
2. **Bottleneck Type2:** Bottleneck type 2 replaces 3x3 convolution with 5x5 convolution. It allows capture larger pattern, helps to model middle to large size features for the signs.

If we look at the whole picture, there is a sequence of operations. It starts with of a 1x1 convolution followed by a 3x3 convolution, Adaptive Average Pooling, and a fully connected layer augmented with the input from the 1x1 convolution. Then a second block starts with a 1x1 convolution followed by a 5x5 convolution, Adaptive Average Pooling, and a fully connected layer augmented with the input from the second 1x1 convolution, and finally a third block with a 1x1 convolution followed by a 3x3 convolution, Adaptive Average Pooling, and a fully connected layer augmented with the input from the third 1x1 convolution. This structure enables a balanced multi-scale feature extraction process where local and global contextual information are captured and refined across stages. The first block emphasizes fine-grained spatial details while

retaining low-level channel-projected features through the skip connection, the second block expands the receptive field to incorporate broader context while still preserving detail from its 1×1 projection, and the third block returns focus to local refinement built on top of the earlier global representation. This layered design ensures that local patterns, global context, and residual connections work together, producing richer and more discriminative feature representations for downstream tasks.

This early placement allows early recalibration of feature channels before deeper layers perform hierarchical abstraction. The structure combines variable kernel bottlenecks with channel attention, ensuring that low-level features such as edges, contours, and texture gradients are emphasized during the first pass of feature extraction. The block employs a dual-path structure with stride-two convolution, formulated as

$$f_1(x) = \text{Conv}_{1 \times 1, s=2}(x), \quad f_2(x) = \text{Conv}_{1 \times 1, s=2}(x),$$

where $f_1(x)$ is further processed through a series of n bottlenecks with alternating kernel sizes $k \in \{3, 5\}$,

$$b_i(z) = \text{Conv}_{1 \times 1}(z) \rightarrow \text{Conv}_{k \times k}(z),$$

and the sequence is expressed as

$$y_1 = b_n \circ b_{n-1} \circ \cdots \circ b_1(f_1(x)).$$

where $[\circ]$ denotes channel concatenation.

The second path $y_2 = f_2(x)$ serves as a compressed bypass, allowing gradient stability and direct information flow. The two paths are concatenated channel-wise and projected through a 1×1 convolution,

$$y = \text{Conv}_{1 \times 1}([y_1, y_2]),$$

The subsequent layer receives feature maps that are both compact and discriminative, which strengthens the stability of optimization and improves the reliability of small-object encoding.

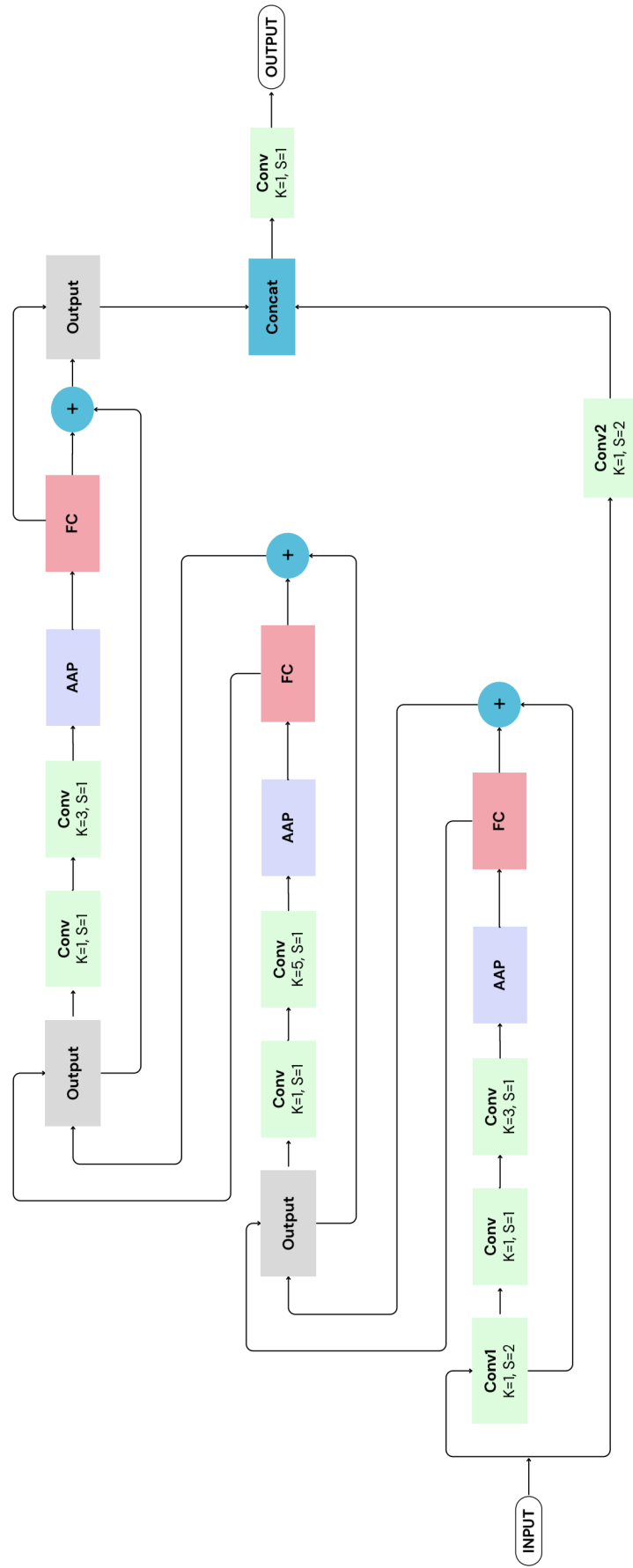


Figure 3.5: Detailed view of the proposed block, showing internal operations and feature flow

Chapter 4

Results and Discussion

This chapter presents a comprehensive evaluation of the proposed traffic sign detection methodology. We begin by describing the datasets used for training and testing, including the Bangladeshi Traffic Sign Dataset, the German Traffic Sign Dataset (GTSDB), and CCTSDB2021. Next, the experimental setup including training protocols and implementation details is discussed. It is followed by a description of the evaluation metrics to assess detection performance. Subsequently, the results are analyzed in detail. We have compared the proposed approach with state-of-the-art models across different datasets to demonstrate improvements in accuracy, robustness, and generalization. Finally, we perform a thorough error analysis to identify common failure modes and provide insights for potential future enhancements. This chapter aims to illustrate not only the quantitative performance gains of our method but also its practical effectiveness in real-world traffic sign detection scenarios.

4.1 Dataset

4.1.1 Bangladeshi Traffic Sign Dataset

This dataset introduced by Ashik et al [3]. The dataset was specifically curated to address the unique challenges of traffic sign recognition in Bangladesh. Its unique geo-social and environmental factors often complicates detection. It comprises 2,986 images with a total of 3,063 annotated traffic signs distributed across 15 distinct classes. The images were extracted from videos recorded in diverse conditions—including urban streets, rural roads, and highways. The footage captured a wide range of challenges such as occlusion, blurriness, low lighting, and varying object sizes. This diversity reflects the complexity of real-world driving scenarios in Bangladesh.

The dataset is structured such that the training set consists of 2,324 images, while the test set contains 662 images. Within the training set, a subset of 2,096 images is designated for actual model training, while the remaining images are reserved for validation. This helps in model tuning and early stopping to prevent overfitting.

Table 4.1: Number of images available for each traffic sign category in Bangladeshi dataset.

Class Name	Images
Pedestrian Crossing	593
No Overtaking	329
Speed Limit 40 km/h	309
School	262
Road Hump	195
Side Road Left	195
Side Road Right	193
Sharp Bend To The Left	174
Crossroads	142
Speed Limit 80 km/h	138
U Turn	116
Sharp Bend to the Right	115
Speed Limit 60 km/h	115
No Use of Horn	109
Narrow Bridge	78
Total	3063

4.1.2 GTSDDB

The German Traffic Sign Detection Benchmark (GTSDDB) consists of 900 road images captured under real-world driving conditions, containing a total of approximately 1,200 annotated traffic signs. Out of these, 788 images are used for training, 218 images for validation, and the test set contains 190 traffic sign instances. Traffic signs in the dataset are categorized into three main classes: prohibitory signs, danger signs, and mandatory signs, covering a variety of shapes, sizes, and appearances.

Table 4.2: Distribution of objects across different traffic sign categories in the German dataset.

Category	Number of Objects
Prohibitory	396
Danger	156
Mandatory	114
Other	186
Total Images	900

Originally, GTSDb dataset has 900 images, 296 new images added in the later version. We have used the later version to train our model and performance evaluation.

4.1.3 CCTSDB2021

The CCTSDB2021 dataset comprises a total of 17,856 images, which are divided into a training set and a positive-sample test set. All traffic signs within these images are annotated and categorized into three semantic groups based on their meanings: mandatory signs, prohibitory signs, and warning signs.

The training set contains 16,356 images, sequentially numbered from 00000 to 18991. These images are primarily intended for model training and encompass a wide variety of traffic sign instances across different environments and conditions. We have randomly chosen certain train samples and put it into valid set. We have separated 3271 images randomly to add into valid set and these images are not visible during training [9], [79], [80], [81].

4.2 Experiment Setup

All experiments were conducted on an NVIDIA GeForce RTX 3060 GPU equipped with 12 GB of VRAM in a local environment. The YOLO-based detection models were trained for up to 700 epochs with a batch size of 8, using an input image resolution of 640×640 pixels. To prevent overfitting and ensure optimal generalization, early stopping was employed with a patience value of 70 epochs, monitoring the validation loss and mAP metrics to terminate training once convergence was reached. This strategy not only reduced unnecessary computation but also helped stabilize model performance across different datasets.

The implementation was based on the Ultralytics YOLO framework, which provides a robust and modular foundation for developing object detection architectures. The proposed Squeeze-and-Excitation (SE) enhanced bottleneck module was integrated into the network using PyTorch. The integration leveraged the `mhaf-yolo`¹ repository as a reference base for structural modifications. Training was carried out using the default hyperparameter configurations of the Ultralytics framework, including the SGD optimizer with momentum, cosine learning rate scheduling, and weight decay regularization. It ensures both reproducibility and comparability with existing benchmarks.

These experiments were conducted using Google Colab’s free-tier cloud environment, which provides an NVIDIA T4 GPU with 16 GB of VRAM. Due to the increased memory capacity, a larger batch size of 16 was employed to expedite training while maintaining stable gradient updates. The Colab environment facilitated rapid prototyping and reproducible experimentation with minimal configuration overhead, further validating the scalability of the proposed model under different computational resources.

Evaluation was performed on the designated test sets of each dataset. Model performance was assessed using standard detection metrics, including mean Average Precision (mAP) at Intersection-over-Union (IoU) thresholds of 0.5 (mAP@0.5) and the averaged range of 0.5:0.95 (mAP@[0.5:0.95]), along with precision, recall, and inference latency. All inference-time measurements were recorded on the same respective GPU environments to ensure fairness and consistency across experiments.

4.3 Evaluation Metrics

In this study, several widely-used metrics are employed to evaluate the performance of the proposed model. These metrics provide complementary insights into different aspects of model performance, such as detection accuracy, completeness, and balance between precision and recall.

1. **Precision:** Precision measures the proportion of correctly predicted positive samples among all samples predicted as positive by the model. High precision indicates that the model produces few false positives, reflecting the reliability of its positive predictions. Precision is formally defined as:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (4.1)$$

¹<https://github.com/yang-0201/MHAF-YOLO>

2. **Recall:** Recall, also known as sensitivity or true positive rate, evaluates the ability of the model to correctly identify all actual positive samples. A high recall score implies that the model misses very few positive instances. It is calculated as:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (4.2)$$

3. **Average Precision (AP):** Average Precision summarizes the precision–recall trade-off for a single class by computing the area under the precision–recall curve. The AP score accounts for both the correctness and completeness of predictions and ranges from 0 to 1. It provides a more robust measure than precision or recall alone, especially for imbalanced datasets:

$$\text{AP} = \int_0^1 \text{Precision}(\text{Recall}) d(\text{Recall}) \quad (4.3)$$

4. **Mean Average Precision (mAP):** Mean Average Precision extends AP to multi-class detection by averaging the AP values over all classes. It is widely used in object detection benchmarks to evaluate overall model performance across diverse categories. mAP is defined as:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^N \text{AP}_i \quad (4.4)$$

where N is the total number of classes, and AP_i denotes the Average Precision for the i -th class.

5. **F1-Score:** The F1-Score combines precision and recall into a single metric by computing their harmonic mean. It provides a balanced measure of a model’s accuracy, especially when precision and recall are in conflict. The F1-Score is given by:

$$\text{F1-Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (4.5)$$

4.4 Performance Comparison with State of the Art Models

For the performance comparison in the experimental phase, four representative models were selected to ensure diversity and relevance in evaluating detection effectiveness. These include three state-of-the-art YOLO variants: YOLOv8, YOLOv10, and

YOLOv11, along with the base architecture of our proposed model MHAF-YOLO, and the final Squeeze-Aware-YOLO. This combination allows us to compare general-purpose YOLO detectors with our specialized models designed specifically for traffic sign detection.

Table 4.3: List of evaluated models along with their total parameter counts

Name	Parameters
YOLOv8	2 M
YOLOv10	2.2 M
YOLOv11	2.6 M
MHAF-YOLO	2.2 M
Squeeze-Aware-YOLO (Ours)	2.2 M

All models train for 700 epochs with SGD (momentum 0.9, initial LR 0.01, decayed by 0.1 at epochs 50/75), batch size 8. A patience value of 70 assigned to stop the model early. We report mean Average Precision (mAP@IoU=0.5) both overall and per class

Bangladeshi Traffic Sign Dataset

Bangladeshi Traffic Sign Dataset has 15 classes encompassing around 3096 images. We can see the performance in the dataset in table 4.4 .

Table 4.4: Performance comparison of all models on the Bangladeshi Traffic Sign Dataset

Models	mAP
YOLO-v8	0.663
YOLO-v10	0.679
YOLO-v11	0.668
MHAF-YOLO	0.685
Squeeze-Aware-YOLO (Ours)	0.747

The comparative results across different models (Tables 4.5) provide several critical insights. At the overall level, Squeeze-Aware-YOLO achieves the highest mAP of 0.747, surpassing YOLOv8 (0.663), YOLOv10 (0.679), YOLOv11 (0.668), and MHAF-YOLO (0.685). This indicates that the proposed modifications contribute significantly to improving detection accuracy in challenging traffic sign recognition tasks. A more granular, per-class breakdown (Table 4.5) reveals where each model excels or struggles.

Squeeze-Aware-YOLO attains the top score in classes 6, 10, 12, 13, and 15, indicating its particular strength in those object categories.

A deeper per-class breakdown highlights the nuanced performance patterns of the models 4.5. YOLOv8 performs exceptionally well in highly distinctive categories such as “No Overtaking” (0.964) and “U-Turn” (0.938), suggesting that it is well-tuned for prohibitory signs with clear visual boundaries. Similarly, YOLOv10 and YOLOv11 show competitive results in categories like “Speed Limit 80 km/h” and “Sharp Bend to the Left,” where structural features are moderately consistent. MHAF-YOLO, despite showing stability in some classes (e.g., “Side Road Left” with 0.980), underperforms in others, indicating limitations in handling intra-class variations.

Table 4.5: Class-wise performance metrics for various YOLO models in Bangladeshi Dataset

Class / Model	YOLOv8	YOLOv10	YOLOv11	MHAF-YOLO	Squeeze-Aware-YOLO (Ours)
Pedestrian Crossing	0.760	0.735	0.425	0.675	0.730
Speed Limit 40 km/h	0.845	0.775	0.722	0.759	0.850
Speed Limit 80 km/h	0.563	0.729	0.725	0.709	0.705
Speed Limit 60 km/h	0.622	0.671	0.737	0.894	0.950
No Overtaking	0.964	0.915	0.947	0.906	0.958
Sharp Bend Right	0.794	0.821	0.793	0.687	0.894
Sharp Bend Left	0.865	0.915	0.858	0.185	0.320
Side Road Right	0.180	0.302	0.379	0.736	0.627
Narrow Bridge	0.568	0.740	0.544	0.558	0.638
No Horn	0.331	0.536	0.428	0.840	0.952
Road Hump	0.887	0.644	0.902	0.513	0.687
School	0.432	0.385	0.404	0.911	0.883
Side Road Left	0.421	0.399	0.509	0.758	0.831
U-Turn	0.938	0.933	0.916	0.529	0.514
Crossroads	0.770	0.645	0.728	0.622	0.660

In contrast, Squeeze-Aware-YOLO demonstrates superior adaptability, particularly in complex or visually ambiguous categories. It secures the best scores in “Sharp Bend

to the Right” (0.894), “No Use of Horn” (0.952), “School” (0.883), “Side Road Left” (0.831), and “Side Road Right” (0.627). These categories typically involve finer structural cues and background clutter which challenge traditional YOLO variants. The model’s ability to excel in such cases highlights its strength in feature recalibration and attention driven learning which enables more discriminative representations.

German Traffic Sign Dataset

The evaluation on the German Traffic Sign Detection Benchmark (GTSDDB) demonstrates competitive performance across different models, with overall mean Average Precision (mAP) results presented in Table 4.7. Among the baselines, YOLOv11 achieves a strong score of 0.836, while YOLOv10 (0.826) and MHAF-YOLO (0.816) remain close in performance. YOLOv8 performs slightly lower at 0.809, yet still shows strong consistency across classes. The proposed Squeeze-Aware-YOLO achieves the highest mAP of 0.84, surpassing all counterparts and highlighting the effectiveness of its design in improving detection accuracy for complex traffic sign recognition tasks.

Table 4.6: mAP scores of different models on the German Traffic Sign Dataset (GTSDDB)

Models	mAP
YOLO-v8	0.809
YOLO-v10	0.826
YOLO-v11	0.836
MHAF-YOLO	0.816
Squeeze-Aware-YOLO (Ours)	0.84

A class-wise analysis reveals further insights into model behavior. YOLOv8 delivers outstanding performance on the “Danger” class (0.949), reflecting its strength in detecting highly distinctive triangular signs. YOLOv10, however, achieves the highest score in “Prohibitory” (0.919), showing its ability to handle circular regulatory signs that often vary in scale and positioning. YOLOv11 performs competitively across multiple categories, securing the best results for the “Others” class (0.825), which includes less frequent and irregular signs. MHAF-YOLO shows balanced but slightly lower performance, with consistent results in “Mandatory” (0.730) and “Danger” (0.866), though it does not dominate in any category.

Table 4.7: mAP performance for individual traffic sign classes in the GTSDDB dataset

Class	Prohibitory	Mandatory	Danger	Others
YOLOv8	0.799	0.711	0.949	0.636
YOLOv10	0.919	0.726	0.906	0.755
YOLOv11	0.907	0.733	0.878	0.825
Mhaf-YOLO	0.9	0.73	0.866	0.767
Squeeze-Aware-YOLO (Ours)	0.91	0.76	0.865	0.825

In contrast, Squeeze-Aware-YOLO attains the leading score in the “Mandatory” category (0.760) and ties YOLOv11 in “Others” (0.825). These results suggest that the proposed model demonstrates particular robustness in categories with higher intra-class variation and more ambiguous visual patterns, where traditional YOLO architectures face limitations. While it does not always secure the top value in every class, its stable performance across categories contributes to the highest overall mAP, reflecting an effective trade-off between specialization and generalization.

CCTSDDB2021

The results on the CCTSDDB2021 dataset highlight the competitiveness of modern object detection models under challenging conditions. As shown in the overall evaluation, YOLOv10 achieves the highest mean Average Precision (mAP) of 0.792, marginally outperforming Squeeze-Aware-YOLO at 0.783, while MHAF-YOLO lags slightly behind at 0.769. Although the proposed Squeeze-Aware-YOLO does not surpass YOLOv10 in the global metric, it demonstrates improved robustness in specific categories, reflecting its adaptability in complex visual environments.

Table 4.8: Overall mAP scores of evaluated models on the CCTSDDB dataset

Models	mAP
YOLO-v10	0.792
MHAF-YOLO	0.769
Squeeze-Aware-YOLO (Ours)	0.783

A closer examination of class-wise performance reveals important distinctions. YOLOv10 excels in “Mandatory” (0.728) and “Warning” (0.855) categories, where signs often include strong geometric cues and distinct color boundaries, suggesting its effective

feature extraction in well-structured contexts. MHAF-YOLO shows balanced performance across all categories but does not lead in any, indicating moderate generalization ability without specialization. In contrast, Squeeze-Aware-YOLO secures the highest performance in the “Prohibitory” class (0.807), where signs are visually diverse and frequently subject to occlusion and scale variations. This suggests that its feature recalibration mechanism enhances recognition of circular regulatory signs, which are typically more prone to background clutter.

Table 4.9: Class-specific mAP values for the CCTSDB2021 traffic sign dataset

Class	Prohibitory	Mandatory	Warning
YOLOv10	0.794	0.728	0.855
Mhaf-YOLO	0.782	0.687	0.838
Squeeze-Aware-YOLO (Ours)	0.807	0.694	0.849

The marginal differences in mAP between Squeeze-Aware-YOLO and YOLOv10 suggest a trade-off between overall stability and class-specific strength. While YOLOv10 maintains an edge in aggregate accuracy, Squeeze-Aware-YOLO provides superior detection in more visually complex or ambiguous categories.

Therefore, the result indicates its potential advantage in real-world deployments where prohibitory signs are crucial for safety-critical decision-making.

4.5 Performance on Bangladeshi Traffic Sign Dataset

The results from the Bangladeshi Traffic Sign Dataset (Table 4.5) demonstrate that Squeeze-Aware-YOLO achieves the highest overall mean Average Precision (mAP = 0.747), outperforming MHAF-YOLO (0.685) and the YOLO variants (0.663–0.679) by a clear margin, indicating its effectiveness in learning discriminative, context-aware features. A closer look at the class-wise performance reveals that our model dominates in seven out of fifteen categories, particularly excelling in classes such as Speed Limit 60 km/h (0.950) and No Use of Horn (0.952), where background clutter and subtle discriminative cues often challenge conventional detectors.

Following the f-1 curve which signifies model performance across confidence thresholds, revealing optimal balance between precision and recall for each traffic sign class.

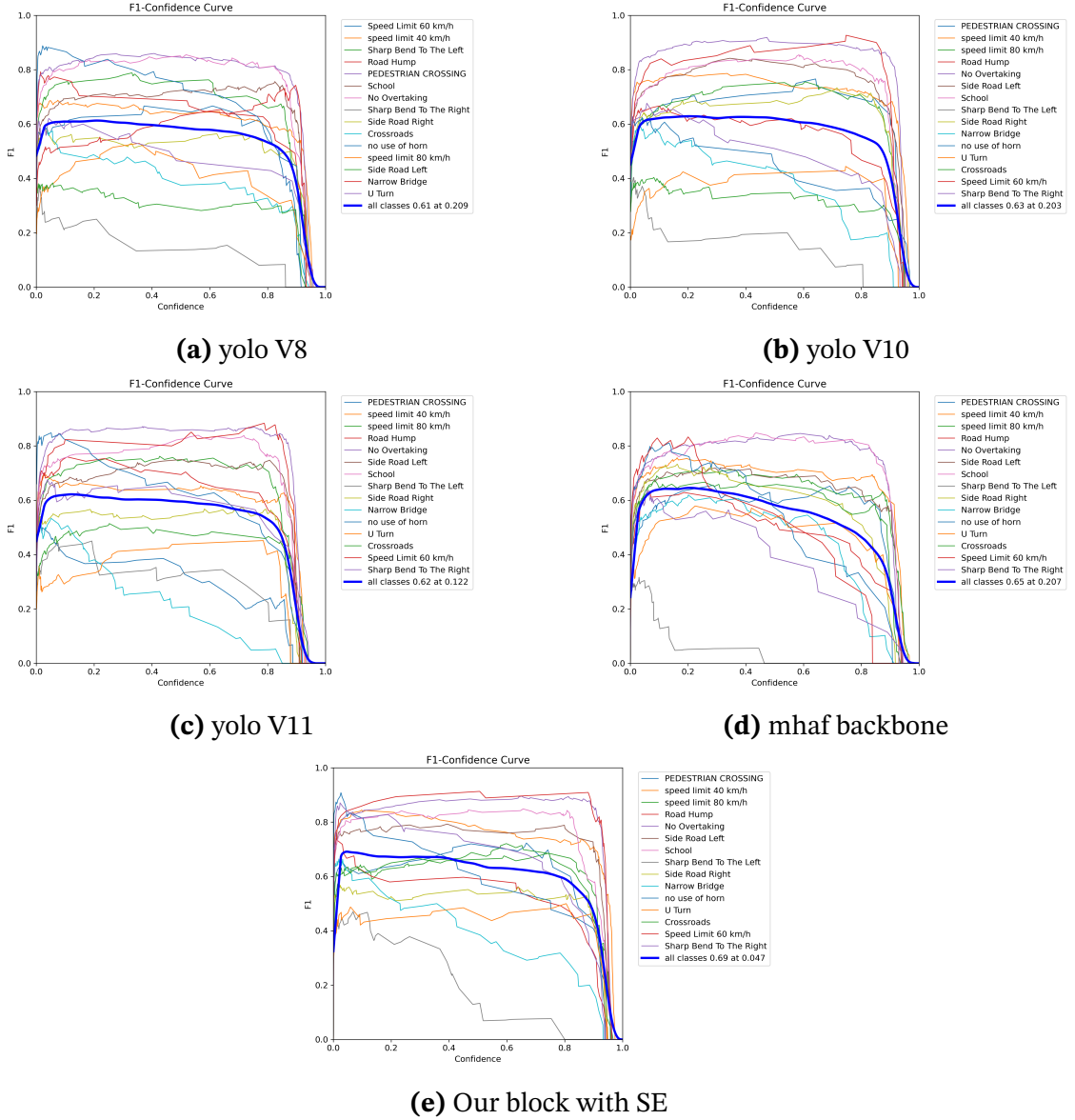


Figure 4.1: F1-score curve comparison across different models in Bangladeshi Dataset

However, the model struggles in specific cases, most notably Sharp Bend to the Left (0.320 vs. 0.929 for MHAF) and U-Turn (0.514 vs. 0.980 for MHAF), highlighting potential issues with orientation sensitivity, class imbalance, or overemphasis on certain feature patterns. While YOLOv8, YOLOv10, and YOLOv11 demonstrate competitiveness in isolated classes, their overall results remain weaker compared to the proposed approach. These discrepancies underline the strengths of Squeeze-Aware-YOLO in handling complex and noisy categories, but also point to failure modes in orientation-sensitive signs and underrepresented classes. Such findings suggest the need for targeted strategies like orientation-preserving data augmentation, class-balanced training, or auxiliary orientation heads to complement the improvements already achieved.

Following is a PR Curve which illustrates the trade-off between precision and recall across thresholds, highlighting model reliability, class-specific detection performance, and threshold selection, especially useful for imbalanced or safety-critical traffic sign datasets.

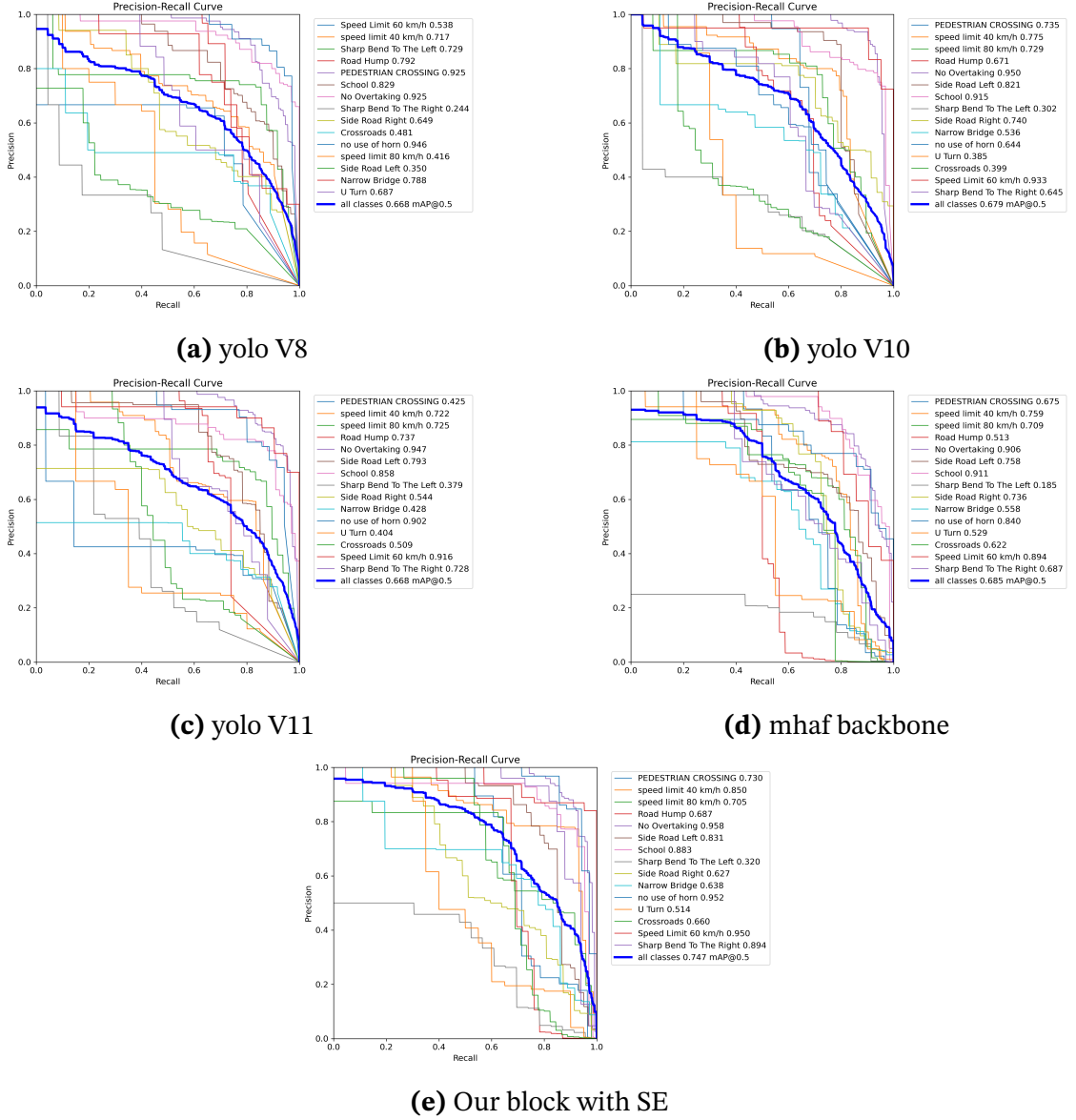


Figure 4.2: Precision–Recall (PR) curve comparison across models on Bangladeshi Dataset

Overall, the analysis confirms that Squeeze-Aware-YOLO provides robust and meaningful improvements across diverse Bangladeshi traffic sign categories, while still leaving space for further refinements to ensure balanced performance across all safety-critical classes. Performance comparison in two different images is shown in the 4.3 and 4.4, where in one image, a traffic sign is far away, and another is occluded with

leaves of a tree.



(a) yolo V8



(b) yolo V10



(c) yolo V11



(d) mhaf backbone



(e) Our Model

Figure 4.3: Sample images illustrating the superior performance of the model on the Bangladeshi Traffic Sign Dataset.

In the 4.3 image, none of the base YOLO models (YOLOv8, YOLOv10, YOLOv11) were able to detect the traffic sign. Among the advanced models, Squeeze-Aware-YOLO achieved a higher confidence score than the base MHAF-YOLO, while MHAF-YOLO itself produced the lowest confidence among the models that made detections. Our proposed model, however, demonstrated the highest confidence, accurately identifying the sign.

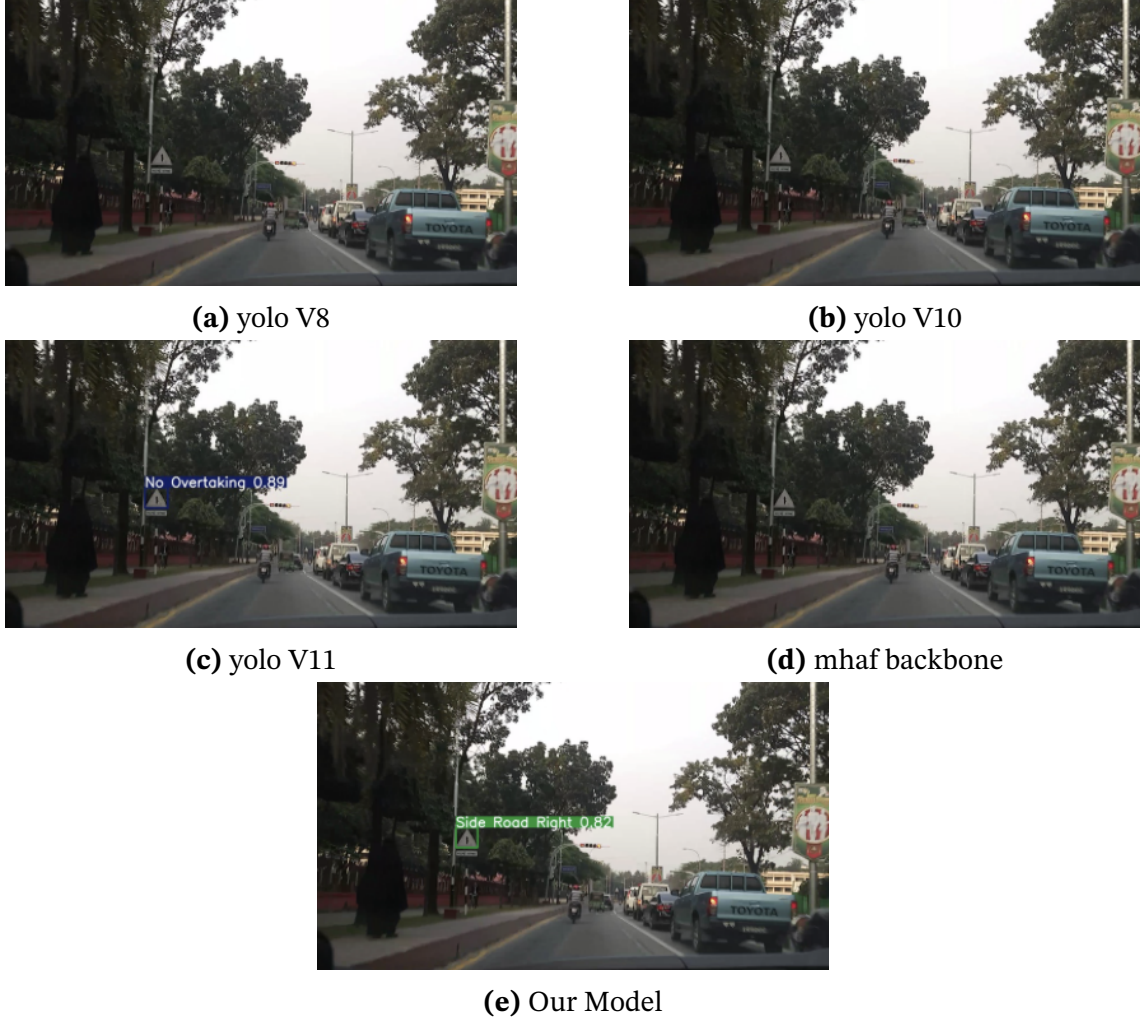


Figure 4.4: Sample instances illustrating the superior performance of the model on the Bangladeshi Traffic Sign Dataset

In another instance on the 4.4 image, most models failed to detect the sign correctly, whereas our model accurately identified it. Notably, YOLOv11 made a detection but misclassified the sign: the correct class is 'Side Road Right,' but YOLOv11 incorrectly labeled it as 'No Overtaking.'

4.6 Performance on German Traffic Sign Dataset

The German Traffic Sign Dataset evaluation reveals nuanced insights into model behavior across different traffic sign categories. While Squeeze-Aware-YOLO achieves the highest overall mAP (0.84), the per-class breakdown shows that no single model dominates all categories which reflects the inherent complexity of this dataset. F-1 Curve for German dataset indicates the confidence threshold.

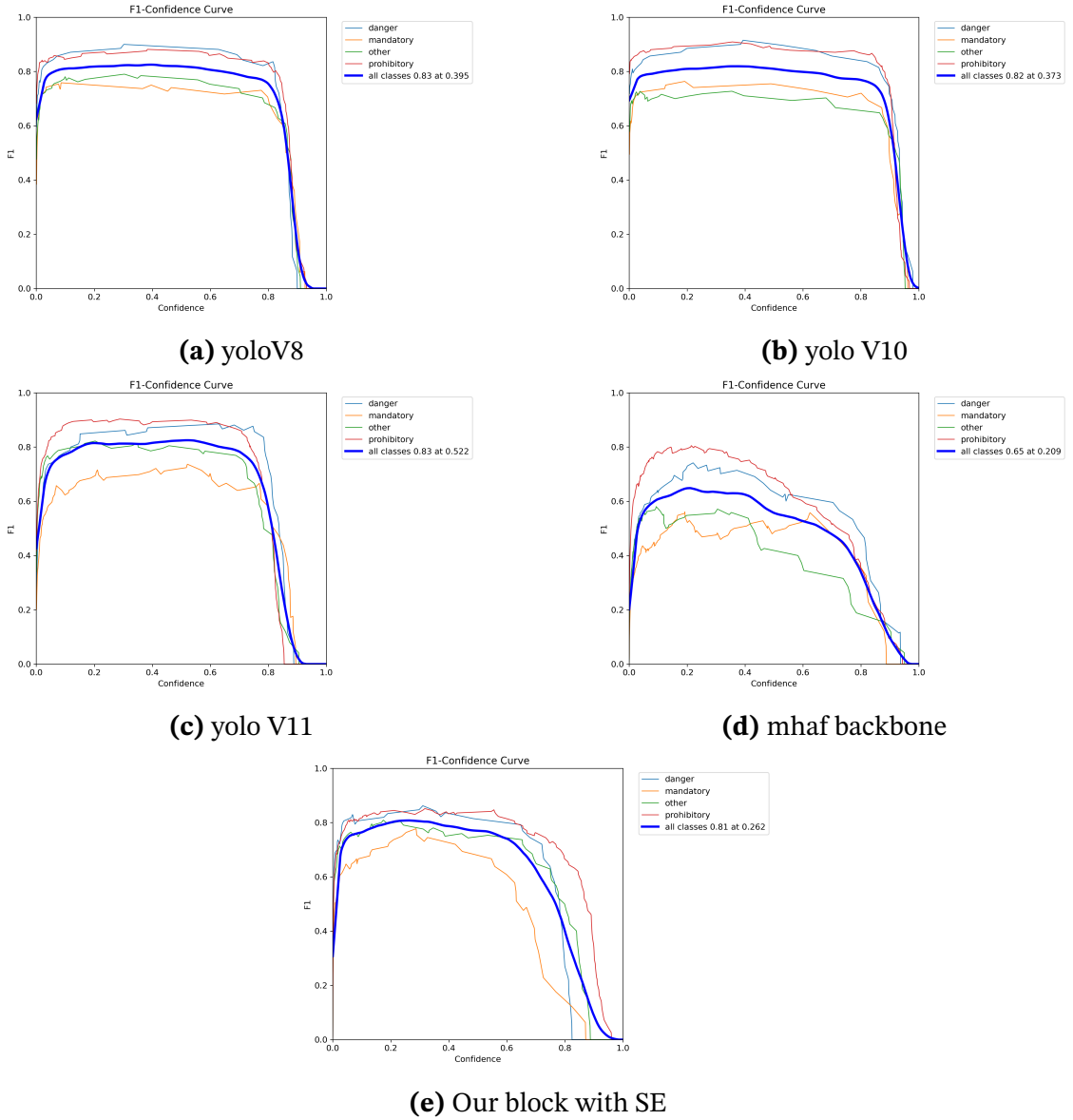
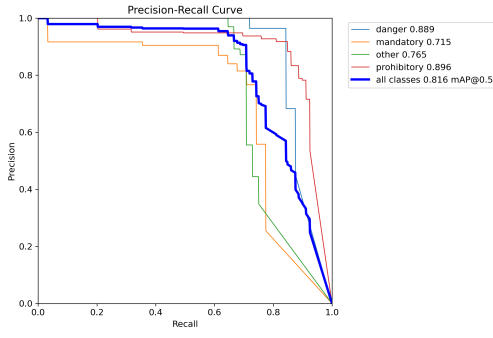
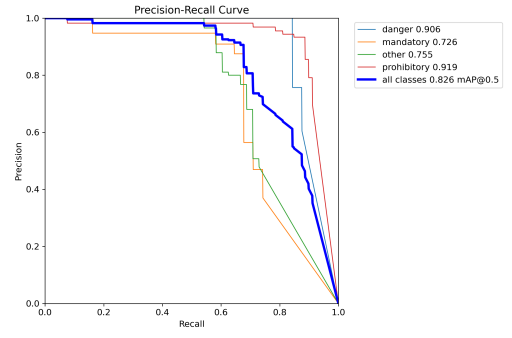


Figure 4.5: F1-score curves of models evaluated on German Dataset

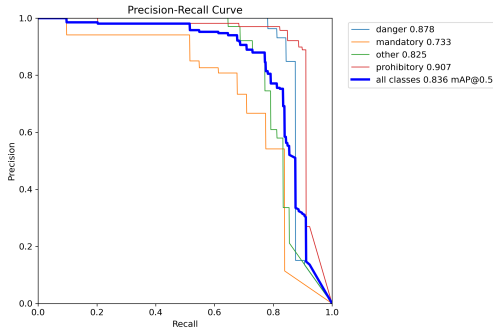
Danger signs, often large and visually prominent, are detected extremely well by YOLOv8 (0.949), whereas subtler signs in the Mandatory and Others categories benefit from Squeeze-Aware-YOLO’s feature refinement and contextual awareness. This indicates that the architecture of Squeeze-Aware-YOLO effectively balances learning generalizable features for less salient or occluded signs while maintaining strong detection for common, well-defined categories. The variation in performance across classes highlights challenges such as intra-class similarity, occlusion, and varying sign sizes, suggesting that even state-of-the-art detectors can be sensitive to visual and contextual nuances. The P-R curve shows performance on imbalanced classes.



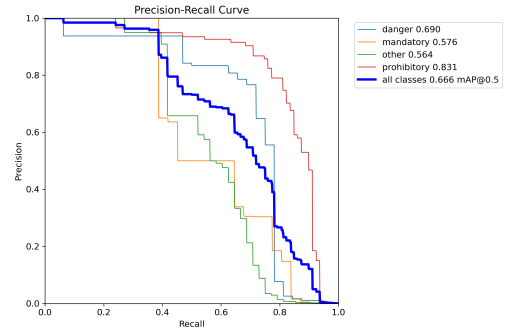
(a) yolo V8



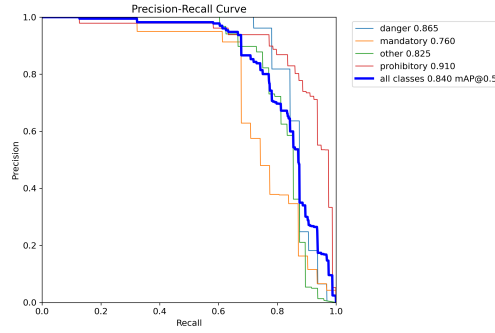
(b) yolo V10



(c) yolo V11



(d) mhaf backbone



(e) Our block with SE

Figure 4.6: PR curve of the models evaluated on German Dataset

Overall, Squeeze-Aware-YOLO demonstrates robust generalization, offering more consistent detection across all classes, a critical attribute for real-world deployment in heterogeneous traffic environments, while also pointing to potential areas for improvement in detecting highly distinctive or rare classes.

The performance comparison on a slightly rotated image from the German Traffic Sign Dataset is illustrated in 4.7 and 4.8 images.



(a) yolo V8



(b) yolo V10



(c) yolo V11



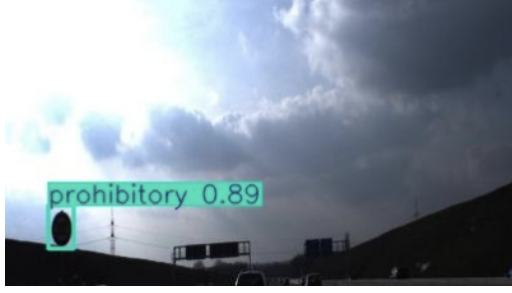
(d) mhaf backbone



(e) Our Model

Figure 4.7: Sample images demonstrating the improved detection performance and higher confidence scores achieved by the proposed model on the GTSDDB dataset

In the 4.7 image, no detections were observed in the base models, while Squeeze-Aware-YOLO achieved a higher confidence score than the other models that made detections. The base models YOLOv8 and YOLOv10 failed to detect the sign, and MHAF-YOLO produced the lowest confidence among the models that detected it. In contrast, our proposed model achieved the highest confidence.



(a) yolo V8



(b) yolo V10



(c) yolo V11



(d) mhaf backbone



(e) Our Model

Figure 4.8: Sample images demonstrating higher confidence scores achieved by the proposed model on the GTSDDB dataset

In the 4.8 image, all models successfully detected the sign; however, only Squeeze-Aware-YOLO exhibited the highest confidence.

4.7 Performance on CCTSDB2021 Dataset

The F1-score curves for the three evaluated models provide deeper insights into their threshold-dependent performance. While YOLOv10 consistently maintains the highest peak F1 across most thresholds, both MHAFF-YOLO and the proposed Squeeze-Aware-YOLO demonstrate relatively stable curves, indicating balanced precision-recall trade-offs over a wide range of operating points. The superior stability of Squeeze-Aware-YOLO, particularly at lower thresholds, highlights its robustness in scenarios where recall is prioritized, such as detecting partially occluded or small-scale traffic signs. This suggests that even though YOLOv10 achieves a higher global mAP,

the proposed model ensures more reliable detection consistency under varying confidence levels.

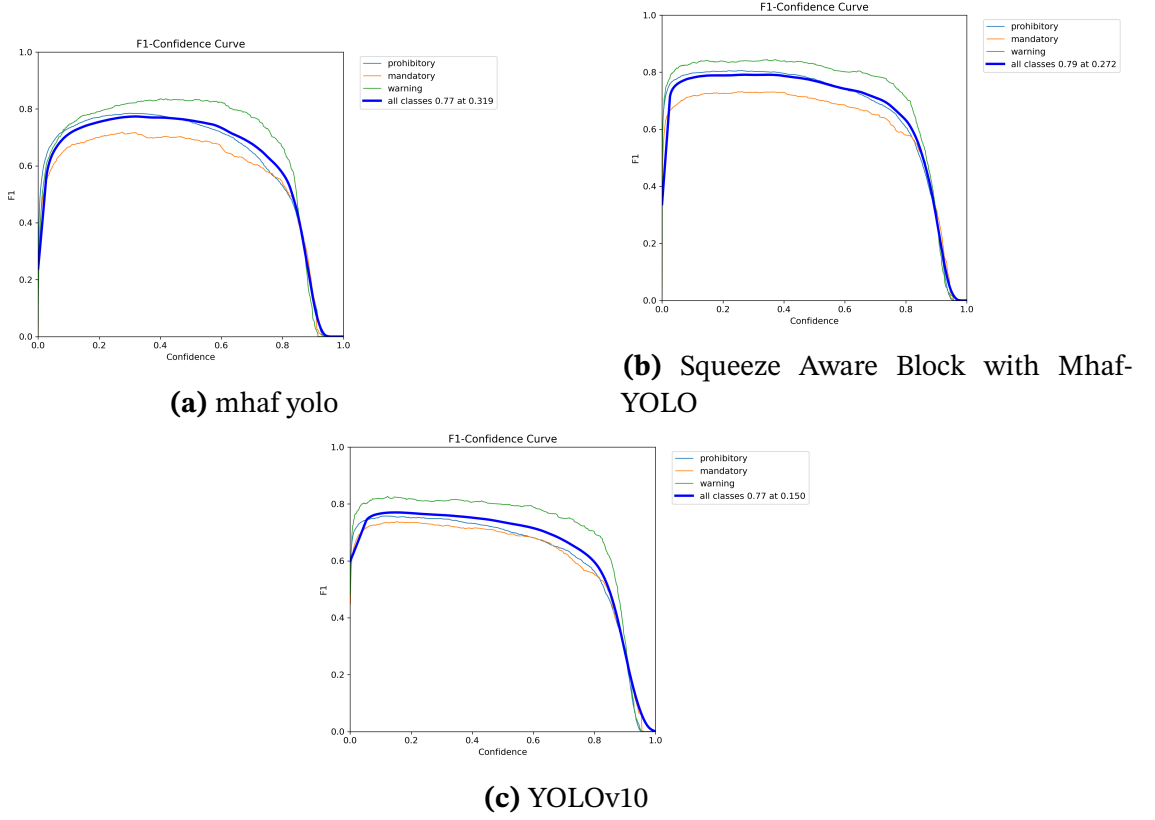


Figure 4.9: F1 curve comparison of the models in CCTSDB

The Precision–Recall (PR) curves further emphasize these distinctions by illustrating the comparative trade-offs between precision and recall. YOLOv10 achieves the best area under the PR curve, reaffirming its dominance in aggregate performance. However, Squeeze-Aware-YOLO maintains higher precision in the prohibitory category across most recall levels, aligning with the class-wise mAP analysis. This behavior suggests that its recalibration mechanism is particularly effective at filtering background noise while preserving sensitivity to critical regulatory signs. MHAF-YOLO, although trailing slightly, delivers more uniform PR curves across categories, reflecting its generalization capacity without clear specialization. Together, these curves underscore the nuanced strengths of each model beyond summary metrics, highlighting that practical deployment may depend on whether stability, precision or recall is prioritized.

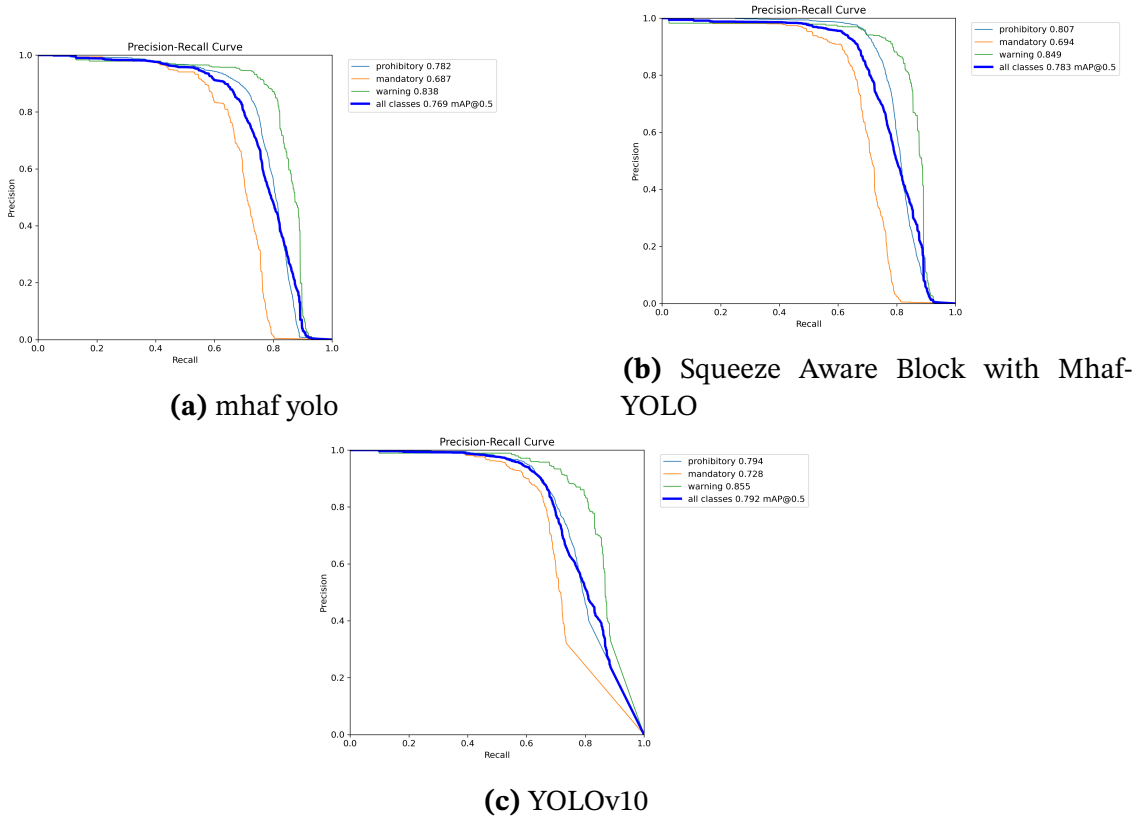


Figure 4.10: PR curve comparison of the models in CCTSDB

In CCTSDB dataset, most of the images have a consistent size, shape, and signs cover almost same amount of pixels. While using base models like YOLOv10, it does not have variable kernels like Mhaf-yolo. It performs better capturing the signs more accurately. However, we have improved the mhaf-yolo model with our block significantly. Our proposed model shows a 2% improvement.

4.8 Error Analysis

On CCTSDB (4-way, normalized), the model is strongest on warning signs, while mandatory suffers the most misses, with many samples collapsing into background. There is also a smaller but consistent prohibitory mandatory swap—expected given their shared circular shapes and low-resolution glyphs. Likely drivers are small objects, motion blur, and class imbalance. However, higher effective input resolution, class-balanced sampling or focal loss, and hard-negative mining should help.

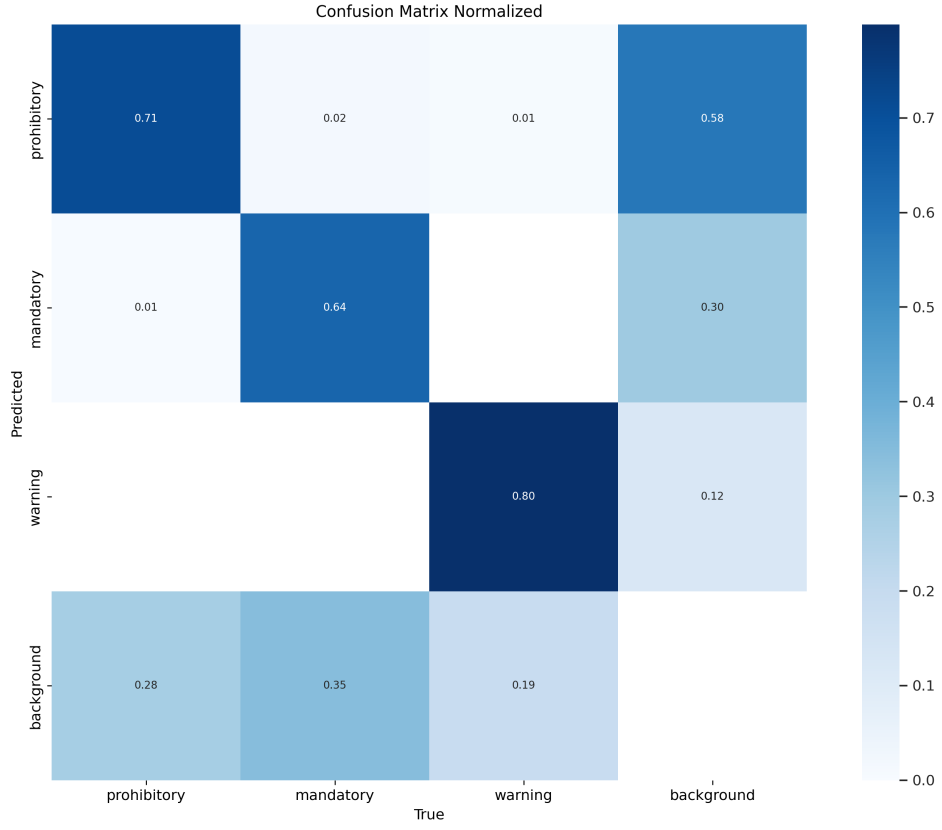


Figure 4.11: Confusion matrix of the Squeeze-Aware-YOLO model on the CCTSDB dataset, illustrating its class-wise detection performance

We can not strongly claim the performance improvement as significant while comparing Squeeze-Aware-YOLO with MHAF-YOLO. Instances like 4.12 and 4.13 show that both models show exact same confidence score and correct detection.

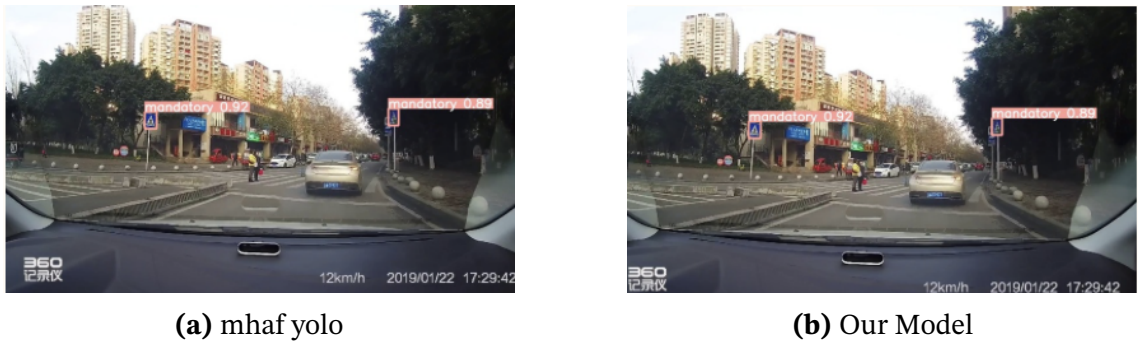


Figure 4.12: Examples of CCTSDB cases where no improvement is observed

Therefore, it undermines the effectiveness of Squeeze-Aware block.



(a) mhaf yolo



(b) Our Model

Figure 4.13: Additional CCTSDB samples showing no significant model improvement

On the Bangladeshi fine-grained matrix, errors cluster among look-alike subclasses: speed-limit digits (40/60/80) confuse each other, left/right variants (Side Road, Sharp Bend) flip under orientation changes, and pairs like U-Turn / No U-Turn or Pedestrian Crossing / School occasionally interchange. Busy urban scenes create both false positives and false negatives. As observed in Table 4.5, although Squeeze-Aware-YOLO achieves the highest overall mean average precision, its performance across individual classes reveals several notable deficiencies. The model demonstrates severe weaknesses in Sharp Bend to the Left, where its score of 0.320 is dramatically lower than the 0.915 obtained by YOLOv10, and in U-turn, where its 0.514 is far below the 0.938 achieved by YOLOv8. Such large gaps indicate that the proposed modifications are not uniformly effective and struggle to generalize to certain critical traffic sign categories. Furthermore, Squeeze-Aware-YOLO is outperformed by MHAF-YOLO in Side Road Right, 0.736 vs. 0.627, School, 0.911 vs. 0.883, and U-turn, 0.529 vs. 0.514, suggesting that Squeeze-Aware Block actually decreased the model capacity to detect certain categories. These shortcomings highlight that while Squeeze-Aware-YOLO introduces improvements at an aggregate level, its inconsistent per-class performance undermines its reliability for comprehensive traffic sign detection tasks.

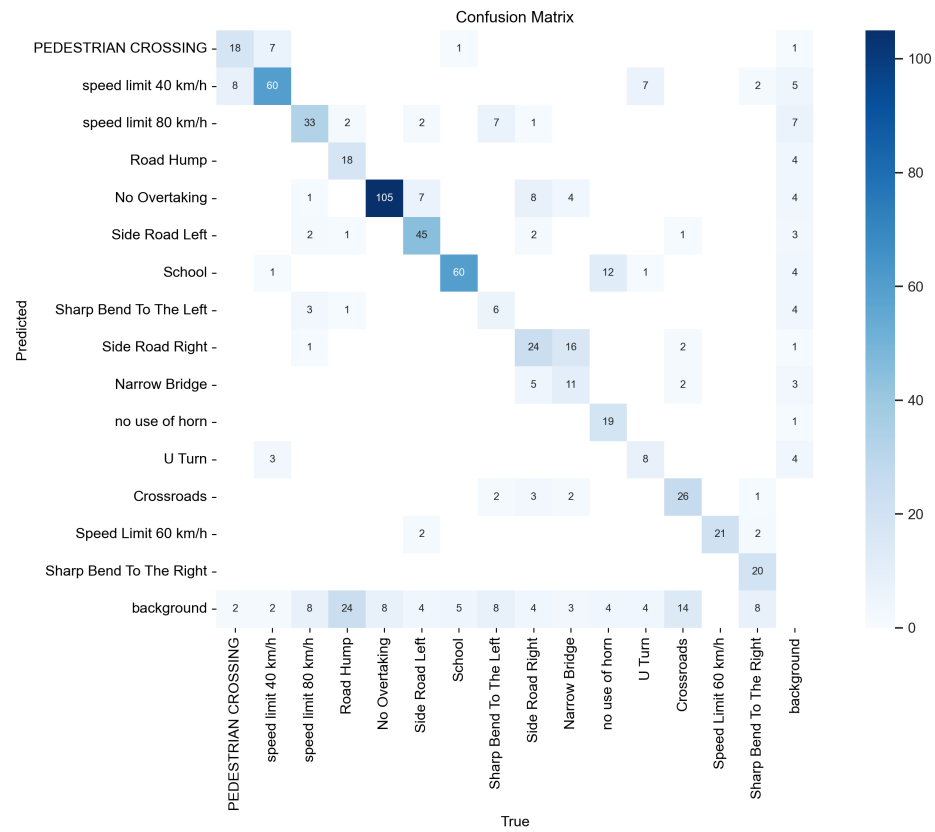


Figure 4.14: Confusion matrix of the Squeeze-Aware-YOLO model on the Bangladeshi dataset, highlighting class-wise misdetections and overall recognition performance

At the macro-family level on the German set (normalized), prohibitory and danger retain high true-positive rates, whereas mandatory and “other” leak into background most often, echoing the CCTSDB trend. A modest mandatory prohibitory confusion persists across families.

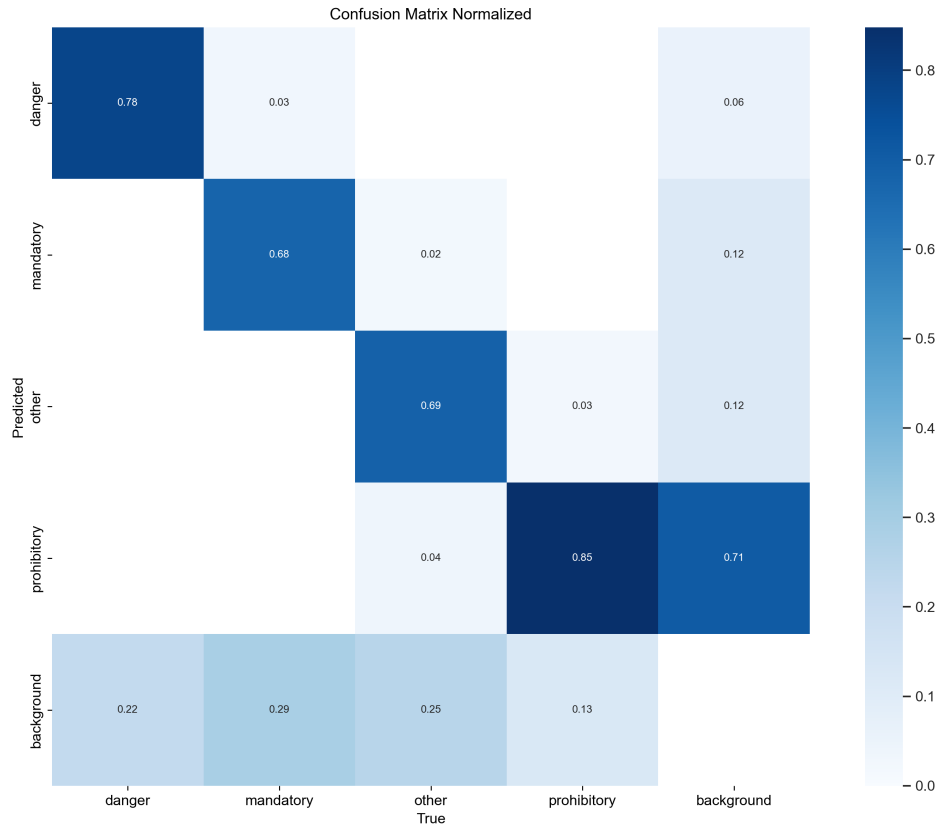


Figure 4.15: Confusion matrix of the Squeeze-Aware-YOLO model on the German dataset, highlighting class-wise misdetections and overall recognition performance

However, Squeeze-Aware-YOLO demonstrates competitive performance across most categories in Table 4.7, its primary limitation is observed in the "Danger" class, where it achieves a mAP of 0.865, notably lower than YOLOv8 (0.949), YOLOv10 (0.906), and YOLOv11 (0.878). This underperformance is particularly significant since danger signs represent critical traffic scenarios where accurate detection is essential for safety-sensitive applications. In contrast, the model performs strongly in the "Mandatory" class, attaining the highest mAP of 0.76, and shows robust results in the "Prohibitory" (0.91) and "Others" (0.825) classes. Therefore, while Squeeze-Aware-YOLO excels in mandatory and general categories, its relative weakness in detecting danger signs highlights an important area for improvement, as misclassification in this class could undermine the reliability of the system in real-world deployment.

Chapter 5

Conclusion

5.1 Summary

A smart traffic sign detection system plays a crucial role in ensuring safe driving, preventing road accidents, and supporting the broader development of autonomous vehicles. For this purpose, a robust dataset that simulates a variety of real-world conditions is essential. German, Bangladesh, Chinese datasets are used for this purpose.

Initially, our proposed enhanced Squeeze Aware Yolo is compared with other state-of-the-art YOLO models in Bangladeshi, Chinese, and German Datasets. The Multi-branch Auxiliary Fusion architecture demonstrated effective feature extraction, utilizing Superficial Assisted Fusion to connect the backbone to a shallow neck, and applying feature fusion to bridge the shallow and deeper necks. However, the initial calibration introduced in our model showed greater performance. We have evaluated both with our Squeeze Aware Block, and without the block. Squeeze Aware block shows significant improvement over normal model.

Quantitative analysis further revealed that our proposed block successfully extracts relevant features at an early stage of processing. This approach outperforms the use of a simple convolutional block in the original MHAF-YOLO. However, the model still struggles with detecting small and distant targets. In these cases, signs occupy only a few pixels, and multiple objects may overlap within a single pixel, making detection particularly challenging. These difficulties are exacerbated when distant signs are blurry, stained, or otherwise obscured.

Nonetheless, it was observed that the base YOLO models exhibited even poorer performance under these conditions. However, only in Chinese dataset (CCTSDb) YOLOv10

outperformed our model. Chinese dataset has a consisted size of traffic signs which Bangladeshi and German does not have. Also, Bangladesh has more challenging images than other, our proposed architecture shows significant improvment over other arccitectures. Morover, our proposed architecture significantly improves upon the YOLOv10 model’s capabilities without introducing a large number of additional parameters. Consequently, the system remains lightweight enough for real-time applications and is suitable for deployment on embedded computing devices, making it highly applicable for autonomous driving systems.

5.2 Limitations

Although the proposed approach demonstrates encouraging outcomes, several limitations were encountered throughout this study that must be acknowledged to provide a balanced perspective on the research findings.

First, the scope of model training was constrained by hardware limitations. All experiments were performed using an NVIDIA RTX 3060 GPU equipped with 12 GB of memory. The training of each epoch required more than two minutes to complete, which substantially restricted the number of epochs that could be executed. This limitation impacted hyperparameter tuning and large-scale experimentation that could have improved convergence stability and feature extraction quality. Furthermore, the restricted computational resources limited the ability to perform extensive ablation studies and multi-scale evaluation. Both of which are essential for fully understanding the behavior and robustness of the model under varying configurations.

Second, despite demonstrating notable improvements over the original MHAf-YOLO architecture, the proposed model did not outperform the standard YOLOv10 baseline on CCTSDB2021 dataset. While the integration of multi-branch feature aggregation enhanced the network’s capacity for capturing spatial dependencies and fine-grained details, it also introduced additional computational overhead that may have hindered real-time efficiency. Moreover, the model exhibited sensitivity when trained on datasets with uniformly scaled images. Under such conditions, the convergence rate slowed, and the model could not outperform base YOLO models. However, when trained on datasets containing images of varied resolutions and aspect ratios, the model maintained more stable gradient updates and yielded improved detection accuracy. Thus, it indicates that our proposed architecture may be more adaptive to heterogeneous visual data.

Finally, dataset diversity presented an inherent limitation that affected the evaluation

and generalization of the model. The Bangladeshi traffic sign dataset employed in this research contains more than ten distinct classes. The dataset encompasses a wide range of environmental variations, lighting conditions, and sign shapes. In contrast, other benchmark datasets such as the German Traffic Sign Detection Benchmark (GTSDB) and CCTSDB2021 comprise fewer than five classes and are collected under relatively consistent imaging conditions. This discrepancy in the number of classes and scene variability complicates direct performance comparison across datasets. The presence of intra-class similarity and class imbalance in the Bangladeshi dataset further contributes to fluctuations in the precision and recall, underscoring the complexity of real-world traffic sign detection in diverse geographical contexts.

5.3 Future Work

To further enhance the robustness and practicality of traffic sign detection systems, future work will focus on several key directions.

First, we aim to develop a more diverse and comprehensive dataset using advanced techniques. Tools such as Generative Adversarial Networks (GANs) and Vision - Language Models will be employed to create variations of existing images. It can help by subtly altering backgrounds or by removing target objects to introduce negative samples.

In addition, we plan to refine the detection model further. This will involve integrating energy efficient architectures, specialized network blocks, and small-object enhancement modules into the proposed YOLO framework. The objective is to enable the model to effectively handle newly generated datasets while maintaining real-time performance and computational efficiency, ensuring broader applicability in autonomous vehicle systems.

References

- [1] R. M. Alamgir et al., “Performance analysis of yolo-based architectures for vehicle detection from traffic images in bangladesh,” in *2022 25th International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2022, pp. 982–987.
- [2] N. AlDahoul, H. A. Karim, A. De Castro, and M. J. T. Tan, “Localization and classification of space objects using efficientdet detector for space situational awareness,” *Scientific reports*, vol. 12, no. 1, p. 21 896, 2022.
- [3] A. N. Ashik, M. S. H. Shanto, R. H. Khan, M. H. Kabir, and S. Ahmed, “Recognizing bangladeshi traffic signs in the wild,” in *2022 25th International Conference on Computer and Information Technology (ICCIT)*, IEEE, 2022, pp. 1004–1009.
- [4] R. Ayachi, M. Afif, Y. Said, and M. Atri, “Traffic signs detection for real-world application of an advanced driving assisting system using deep learning,” *Neural Processing Letters*, vol. 51, no. 1, pp. 837–851, 2020.
- [5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [6] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-end object detection with transformers,” in *European conference on computer vision*, Springer, 2020, pp. 213–229.
- [7] X. Changzhen, W. Cong, M. Weixin, and S. Yanmei, “A traffic sign detection algorithm based on deep convolutional neural network,” in *2016 IEEE International Conference on Signal and Image Processing (ICSIP)*, IEEE, 2016, pp. 676–679.
- [8] T. Chen and S. Lu, “Accurate and efficient traffic sign detection using discriminative adaboost and support vector regression,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4006–4015, 2015.

- [9] J. Chu, C. Zhang, M. Yan, H. Zhang, and T. Ge, "Trd-yolo: A real-time, high-performance small traffic sign detection algorithm," *Sensors*, vol. 23, no. 8, p. 3871, 2023.
- [10] C. Dewi, R.-C. Chen, X. Jiang, and H. Yu, "Deep convolutional neural network for enhancing traffic sign recognition developed on yolo v4," *Multimedia Tools and Applications*, vol. 81, no. 26, pp. 37 821–37 845, 2022.
- [11] F. A. Dina, M. H. Manik, and S. Mahmud, "Urban road accident scenarios in dhaka metropolitan area," *Journal of Transportation Systems*, pp. 8–23, 2024.
- [12] C. Ding, Y. Li, Y. Xia, L. Zhang, and Y. Zhang, "Automatic kernel size determination for deep neural networks based hyperspectral image classification," *Remote Sensing*, vol. 10, no. 3, p. 415, 2018.
- [13] S. Du et al., "Tsd-yolo: Small traffic sign detection based on improved yolo v8," *IET Image Processing*, vol. 18, no. 11, pp. 2884–2898, 2024.
- [14] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, G. Neuhold, and Y. Kuang, "The mapillary traffic sign dataset for detection and classification on a global scale," in *European Conference on Computer Vision*, Springer, 2020, pp. 68–84.
- [15] A. Farzipour, O. N. Manzari, and S. B. Shokouhi, "Traffic sign recognition using local vision transformer," in *2023 13th International Conference on Computer and Knowledge Engineering (ICCCKE)*, IEEE, 2023, pp. 191–196.
- [16] D. Feng, A. Harakeh, S. L. Waslander, and K. Dietmayer, "A review and comparative study on probabilistic object detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 9961–9980, 2021.
- [17] P. Gao, M. Zheng, X. Wang, J. Dai, and H. Li, "Fast convergence of detr with spatially modulated co-attention," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 3621–3630.
- [18] X. Gao, L. Chen, K. Wang, X. Xiong, H. Wang, and Y. Li, "Improved traffic sign detection algorithm based on faster r-cnn," *Applied Sciences*, vol. 12, no. 18, p. 8948, 2022.
- [19] B. Gašparović, J. Lerga, G. Mauša, and M. Ivašić-Kos, "Deep learning approach for objects detection in underwater pipeline images," *Applied artificial intelligence*, vol. 36, no. 1, p. 2 146 853, 2022.
- [20] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

- [21] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [22] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.
- [23] N. Gray et al., “Glare: A dataset for traffic sign detection in sun glare,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 12 323–12 330, 2023.
- [24] G. Han, W. Zhou, N. Sun, J. Liu, and X. Li, “Feature fusion and adversary occlusion networks for object detection,” *IEEE Access*, vol. 7, pp. 124 854–124 865, 2019.
- [25] T. Han, L. Sun, and Q. Dong, “An improved yolo model for traffic signs small target image detection,” *Applied Sciences*, vol. 13, no. 15, p. 8754, 2023.
- [26] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, “Detection of traffic signs in real-world images: The german traffic sign detection benchmark,” in *The 2013 international joint conference on neural networks (IJCNN)*, Ieee, 2013, pp. 1–8.
- [27] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.
- [28] J. M. Kaleybar, H. Khaloo, and A. Naghipour, “Efficient vision transformer for accurate traffic sign detection,” in *2023 13th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, 2023, pp. 036–041.
- [29] S. Kang, Z. Hu, L. Liu, K. Zhang, and Z. Cao, “Object detection yolo algorithms and their industrial applications: Overview and comparative analysis,” *Electronics*, vol. 14, no. 6, p. 1104, 2025.
- [30] S. Karim, Y. Zhang, S. Yin, I. Bibi, and A. A. Brohi, “A brief review and challenges of object detection in optical remote sensing imagery,” *Multiagent and Grid Systems*, vol. 16, no. 3, pp. 227–243, 2020.
- [31] A. Kerim and M. Ö. Efe, “Recognition of traffic signs with artificial neural networks: A novel dataset and algorithm,” in *2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, IEEE, 2021, pp. 171–176.
- [32] R. Khanam and M. Hussain, “What is yolov5: A deep look into the internal features of the popular object detector,” *arXiv preprint arXiv:2407.20892*, 2024.

- [33] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024.
- [34] I. Kilic and G. Aydin, "Traffic sign detection and recognition using tensorflow's object detection api with a new benchmark dataset," in *2020 international conference on electrical engineering (ICEE)*, IEEE, 2020, pp. 1–5.
- [35] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.
- [36] F. Larsson and M. Felsberg, "Using fourier descriptors and spatial models for traffic sign recognition," in *Scandinavian conference on image analysis*, Springer, 2011, pp. 238–249.
- [37] Y. Lee, J.-w. Hwang, S. Lee, Y. Bae, and J. Park, "An energy and gpu-computation efficient backbone network for real-time object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 0–0.
- [38] C. Li et al., "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.
- [39] Z. Li, D. Jia, Z. He, and N. Wu, "Msg-yolo: A multi-scale dynamically enhanced network for the real-time detection of small impurities in large-volume parenterals.," *Electronics (2079-9292)*, vol. 14, no. 6, 2025.
- [40] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.
- [41] H. Liu, K. Zhou, Y. Zhang, and Y. Zhang, "Etsr-yolo: An improved multi-scale traffic sign detection algorithm based on yolov5," *PLoS one*, vol. 18, no. 12, e0295807, 2023.
- [42] Z. Liu, D. Li, S. S. Ge, and F. Tian, "Small traffic sign detection from large image," *Applied Intelligence*, vol. 50, no. 1, pp. 1–13, 2020.
- [43] A. Madani and R. Yusof, "Malaysian traffic sign dataset for traffic sign detection and recognition systems," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 8, no. 11, pp. 137–143, 2016.
- [44] R. K. Megalingam, K. Thanigundala, S. R. Musani, H. Nidamanuru, and L. Gadde, "Indian traffic sign detection and recognition using deep learning," *International journal of transportation science and technology*, vol. 12, no. 3, pp. 683–699, 2023.

- [45] D. Mijić, M. Brisinello, M. Vranješ, and R. Grbić, "Traffic sign detection using yolov3," in *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, IEEE, 2020, pp. 1–6.
- [46] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE transactions on intelligent transportation systems*, vol. 13, no. 4, pp. 1484–1497, 2012.
- [47] Z. Ouairdhi, M. Zbakh, and S. Mahmoudi, "Bridging 2d and 3d object detection: Advances in occlusion handling through depth estimation," *Computer Modeling in Engineering & Sciences*, vol. 143, no. 3, p. 2509, 2025.
- [48] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [49] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.
- [50] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
- [51] Y. Rehman, I. Riaz, X. Fan, and H. Shin, "D-patches: Effective traffic sign detection with occlusion handling," *IET Computer Vision*, vol. 11, no. 5, pp. 368–377, 2017.
- [52] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.
- [53] F. Saeed, M. Ahmed, M. Gul, J. Kim, A. Paul, and M. Kavitha, *A robust approach for industrial small-object detection using an improved faster regional convolutional neural network. sci rep 11: 23390*, 2021.
- [54] C. G. Serna and Y. Ruichek, "Classification of traffic signs: The european dataset," *IEEE Access*, vol. 6, pp. 78 136–78 148, 2018.
- [55] C. G. Serna and Y. Ruichek, "Traffic signs detection and classification for european urban environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 10, pp. 4388–4399, 2019.
- [56] T. Shehzadi, K. A. Hashmi, M. Liwicki, D. Stricker, and M. Z. Afzal, "Object detection with transformers: A review," *Sensors*, vol. 25, no. 19, p. 6025, 2025.

- [57] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, “The german traffic sign recognition benchmark: A multi-class classification competition,” in *The 2011 international joint conference on neural networks*, IEEE, 2011, pp. 1453–1460.
- [58] F. Sultana, A. Sufian, and P. Dutta, “A review of object detection models based on convolutional neural network,” *Intelligent computing: image processing based applications*, pp. 1–16, 2020.
- [59] D. Tabernik and D. Skočaj, “Deep learning for large-scale traffic-sign detection and recognition,” *IEEE transactions on intelligent transportation systems*, vol. 21, no. 4, pp. 1427–1440, 2019.
- [60] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.
- [61] M. Tan and Q. Le, “Efficientnetv2: Smaller models and faster training,” in *International conference on machine learning*, PMLR, 2021, pp. 10 096–10 106.
- [62] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.
- [63] D. Temel, T. Alshaw, M.-H. Chen, and G. AlRegib, “Challenging environments for traffic sign detection: Reliability assessment under inclement conditions,” *arXiv preprint arXiv:1902.06857*, 2019.
- [64] D. Temel, M.-H. Chen, and G. AlRegib, “Traffic sign detection under challenging conditions: A deeper look into performance variations and spectral characteristics,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3663–3673, 2019.
- [65] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, “A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas,” *Machine learning and knowledge extraction*, vol. 5, no. 4, pp. 1680–1716, 2023.
- [66] Y. Tian, J. Gelernter, X. Wang, J. Li, and Y. Yu, “Traffic sign detection using a multi-scale recurrent attention network,” *IEEE transactions on intelligent transportation systems*, vol. 20, no. 12, pp. 4466–4475, 2019.
- [67] R. Timofte, K. Zimmermann, and L. Van Gool, “Multi-view traffic sign detection, recognition, and 3d localisation,” *Machine vision and applications*, vol. 25, pp. 633–647, 2014.

- [68] M. Trigka and E. Dritsas, "A comprehensive survey of machine learning techniques and models for object detection," *Sensors*, vol. 25, no. 1, p. 214, 2025.
- [69] M. M. R. Tusher et al., "Bantraffnet: Bangladeshi traffic sign recognition using a lightweight deep learning approach," 2024.
- [70] A. Wang et al., "Yolov10: Real-time end-to-end object detection. arxiv 2024," *arXiv preprint arXiv:2405.14458*, 2024.
- [71] C.-Y. Wang, I.-H. Yeh, and H.-Y. Mark Liao, "Yolov9: Learning what you want to learn using programmable gradient information," in *European conference on computer vision*, Springer, 2024, pp. 1–21.
- [72] Y. Wang, K. Zhang, L. Wang, and L. Wu, "An improved yolov8 algorithm for rail surface defect detection," *IEEE Access*, 2024.
- [73] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 129–137.
- [74] L. Wu, H. Li, J. He, and X. Chen, "Traffic sign detection method based on faster r-cnn," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1176, 2019, p. 032 045.
- [75] X. Xu, J. Jin, S. Zhang, L. Zhang, S. Pu, and Z. Chen, "Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry," *Future generation computer systems*, vol. 94, pp. 381–391, 2019.
- [76] Z. Yang et al., "Multi-branch auxiliary fusion yolo with re-parameterization heterogeneous convolutional for accurate object detection," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, Springer, 2024, pp. 492–505.
- [77] Z. Yang et al., "Mhaf-yolo: Multi-branch heterogeneous auxiliary fusion yolo for accurate object detection," *arXiv preprint arXiv:2502.04656*, 2025.
- [78] Z. Yang, "Intelligent recognition of traffic signs based on improved yolo v3 algorithm," *Mobile Information Systems*, vol. 2022, no. 1, p. 7 877 032, 2022.
- [79] J. Zhang, Y. Lv, J. Tao, F. Huang, and J. Zhang, "A robust real-time anchor-free traffic sign detector with one-level feature," *IEEE transactions on emerging topics in computational intelligence*, vol. 8, no. 2, pp. 1437–1451, 2024.

- [80] J. Zhang, W. Wang, C. Lu, J. Wang, and A. K. Sangaiah, "Lightweight deep network for traffic sign classification," *Annals of Telecommunications*, vol. 75, no. 7, pp. 369–379, 2020.
- [81] J. Zhang, X. Zou, L.-D. Kuang, J. Wang, R. S. Sherratt, and X. Yu, "Cctsd 2021: A more comprehensive traffic sign detection benchmark," *Human-centric Computing and Information Sciences*, vol. 12, 2022.
- [82] W. Zhang, C. Fu, H. Xie, M. Zhu, M. Tie, and J. Chen, "Global context aware rcnn for object detection," *Neural Computing and Applications*, vol. 33, no. 18, pp. 11 627–11 639, 2021.
- [83] Y. Zhang, Z. Wang, Y. Qi, J. Liu, and J. Yang, "Ctsd: A dataset for traffic sign recognition in complex real-world images," in *2018 IEEE Visual Communications and Image Processing (VCIP)*, IEEE, 2018, pp. 1–4.
- [84] Y. Zhao et al., "Detrs beat yolos on real-time object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 16 965–16 974.
- [85] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.
- [86] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2110–2118.