A Lightweight YOLOv11n-Based Urban Surveillance Framework for Real-Time Auto Rickshaw Detection and Comparative Performance Evaluation Across YOLO Versions in Dhaka City

by

Bipul Chandro Roy
ID : CSE2301028001
Md Shawaon
ID : CSE2301028059
G.M. Shakil Ahmed
ID : CSE2301028100
Md. Abu Jafor Emon
ID : CSE2301028126
Jannatul Ferdaus
ID : CSE2301028175

Supervised by
Tarunnyamoye Kundu

Submitted in partial fulfilment of the requirements for the degree of
Bachelor of science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)
September 2026

A Lightweight YOLOv11n-Based Urban Surveillance Framework for Real-Time Auto Rickshaw Detection and Comparative Performance Evaluation Across YOLO Versions in Dhaka City

by

Bipul Chandro Roy
ID : CSE2301028001
Md Shawaon
ID : CSE2301028059
G.M. Shakil Ahmed
ID : CSE2301028100
Md. Abu Jafor Emon
ID : CSE2301028126
Jannatul Ferdaus
ID : CSE2301028175

Supervised by
Tarunnyamoye Kundu

Submitted in partial fulfilment of the requirements for the degree of
Bachelor of science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)
September 2026

APPROVAL

The thesis titled "**A Lightweight YOLOv11n-Based Urban Surveillance Framework for Real-Time Auto Rickshaw Detection and Comparative Performance Evaluation Across YOLO Versions in Dhaka City** " submitted by **Bipul Chandro Roy** (CSE2301028001), **Md Shawaon** (CSE2301028059), **G.M. Shakil Ahmed** (CSE2301028100), **Md. Abu Jafor Emon** (CSE2301028126), **Jannatul Ferdaus** (CSE2301028175), to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Tarunnyamoye Kundu — **Supervisor**

Lecturer,
Department of Computer Science and Engineering
Sonargaon University (SU)

(Examiner Name and Signature) — **Examiner 1**
Department of Computer Science and Engineering
Sonargaon University (SU)

(Examiner Name and Signature) — **Examiner 2**
Department of Computer Science and Engineering
Sonargaon University (SU)

(Examiner Name and Signature) — **Examiner 3**
Department of Computer Science and Engineering
Sonargaon University (SU)

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of Tarunnyamoye Kundu, Lecturer, Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this thesis has been or is being submitted elsewhere for the award of any degree.

Countersigned

(Tarunnyamoye Kundu)
Supervisor

Signature

Bipul Chandro Roy
ID : CSE2301028001

Md Shawaon
ID : CSE2301028059

G.M. Shakil Ahmed
ID : CSE2301028100

Md. Abu Jafor Emon
ID : CSE2301028126

Jannatul Ferdaus
ID : CSE2301028175

ABSTRACT

Dhaka City experiences persistent traffic congestion due to increasing vehicle density, limited road capacity, and the complexity of its mixed-traffic environment. Auto-rickshaws are a common component of urban transportation, making their automated detection valuable for real-time traffic monitoring and intelligent transportation systems. This study presents a lightweight YOLO-based urban surveillance framework for real-time auto-rickshaw detection in Dhaka City. An Auto-RickshawImageBD dataset containing 1,331 original images was used as the primary data source. Through data augmentation using the Albumentations library, the dataset was expanded to 3,671 images. The images were annotated into two classes, namely auto_rickshaw and non_auto_rickshaw, and prepared in the YOLO object-detection format. Three lightweight YOLO models, YOLO11n, YOLO12n, and YOLO26n, were trained and comparatively evaluated using Precision, Recall, mAP@0.5, mAP@0.5:0.95, and F1-score. The experimental results show that YOLO11n achieved the best overall detection performance. On the test dataset, YOLO11n obtained an mAP@0.5 of 83.93%, mAP@0.5:0.95 of 47.64%, Precision of 84.19%, Recall of 73.82%, and F1-score of 78.66%. YOLO26n achieved the second-best performance, while YOLO12n produced comparatively lower detection results under the evaluated configuration. Based on these findings, YOLO11n was selected as the primary model for the proposed lightweight surveillance framework. The framework is intended to process traffic-camera imagery and provide vehicle detection information that can support traffic-density estimation and congestion alerts. The study demonstrates that a lightweight YOLO model can provide an effective balance between detection accuracy and computational efficiency for auto-rickshaw monitoring in the challenging urban traffic environment of Dhaka City.

KEYWORDS

YOLOv11n; Dhaka City; Auto Rickshaw; Performance Evaluation; Urban Surveillance

ACKNOWLEDGMENT

First and foremost, we express our deepest gratitude to the Almighty Allah for granting us the strength, patience, and ability to successfully complete this thesis within the scheduled time.

We would like to express our sincere gratitude to our respected supervisor, **Tarunnyamoye Kundu**, Lecturer, Department of Computer Science and Engineering, Sonargaon University, for her valuable guidance, continuous support, constructive suggestions, and encouragement throughout the research. Her supervision and direction were essential to the successful completion of this work.

We are also deeply grateful to **Brig. Gen. (Retd) Prof. Habibur Rahman Kamal, ndc, psc**, Dean, Faculty of Science & Engineering, and **Professor Bulbul Ahamed**, Head, Department of Computer Science and Engineering, Sonargaon University, for their valuable support, encouragement, and cooperation during our academic and research activities.

We would like to extend our sincere thanks to all the respected teachers of the Department of Computer Science and Engineering, Sonargaon University, for sharing their knowledge, providing valuable academic guidance, and inspiring us throughout our educational journey.

Finally, we express our deepest love and gratitude to our parents and family members for their continuous support, encouragement, patience, prayers, and unconditional love. Their support gave us the motivation and confidence to complete this research successfully.

We are sincerely grateful to everyone who directly or indirectly contributed to the successful completion of this thesis.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
Alb	Albumentations
CNN	Convolutional Neural Network
CCTV	Closed-Circuit Television
CPU	Central Processing Unit
CSE	Computer Science and Engineering
DFL	Distribution Focal Loss
F1	F1-Score
FPN	Feature Pyramid Network
FPS	Frames Per Second
IoU	Intersection over Union
ITS	Intelligent Transportation System
mAP	mean Average Precision
NMS	Non-Maximum Suppression
PAN	Path Aggregation Network
PSA	Position-Sensitive Attention
R-ELAN	Residual Efficient Layer Aggregation Network
SPPF	Spatial Pyramid Pooling-Fast
TP	True Positive
FP	False Positive
FN	False Negative
YOLO	You Only Look Once
YOLO11n	YOLO11 Nano
YOLO12n	YOLO12 Nano
YOLO26n	YOLO26 Nano
CNG	Compressed Natural Gas
GPU	Graphics Processing Unit
ProgLoss	Progressive Loss Balancing
STAL	Small-Target-Aware Label Assignment
MuSGD	Muon–Stochastic Gradient Descent hybrid optimiser
BUET	Bangladesh University of Engineering and Technology
ARI	Accident Research Institute
C2PSA	Convolutional block with Parallel Spatial Attention

TABLE OF CONTENTS

Title Page No

APPROVAL.....i
DECLARATION.....ii
ABSTRACT.....iii
ACKNOWLEDGEMENT.....iv
LIST OF ABBREVIATION.....v

CHAPTER 1 **1-4**
INTRODUCTION
1.0 Introduction 1
1.1 Background of the Research.....2
1.2 Objectives of the Research 3
1.3 Rationale of the Research 3
1.4 Required Tools and Packages.....4
1.5 Scope of the Research.....4

CHAPTER 2 **5-7**
LITERATURE REVIEW.....5

CHAPTER 3 **8-12**
PREPARING THE DATASET
3.0 Data Collection.....8
3.1 Dataset Sources and Regions.....8
3.2 Data Augmentations.....9
3.3 Annotation Protocol.....10
3.4 Data Segmentation.....10
3.5 Data Preparation.....11

CHAPTER 4 **13-16**
MODEL ARCHITECTURE
4.0 Model Architecture 13
4.1 Models Selected 13
4.1.1 Yolo v11.....13
4.1.2 Yolo v12.....14

4.1.3 Yolo v26.....15

CHAPTER 5 **16-21**
METHODOLOGY
5.0 Methodology 17
5.1 YOLOv11n-Based System Block Diagram.....17
5.2 Methodology Pipeline.....18
5.3 Libraries and Frameworks.....19
5.4 Experimental Setup.....19
5.5 Evaluation Metrics.....20

CHAPTER 6 **22-29**
MODEL TRAINING
6.0 YOLOv11n Model Training.....22
6.1 Environment and Package Setup.....22
6.2 YOLOv11n Model Initialization.....23
6.3Environment and Package Verification.....23
6.4 Model Loading.....24
6.5 Dataset Configuration.....25
6.6 Model Training Procedure 26
6.7 Model Evaluation.....27

CHAPTER 7 **30-33**
TESTING AND RESULT ANALYSIS
7.0 Testing and Result Analysis.....30

CHAPTER 8 **34**
CHALLENGES & LIMITATION
8.0 Challenges & Limitation.....34

CHAPTER 9 **35-36**
CONCLUSION AND FUTURE WORKS
9.0 Conclusion.....35
9.1. Future work.....35

REFERENCES.....37-38

LIST OF FIGURES

Figure No.	Title	Page
Fig. 3.5.1	YOLO-Formatted Annotation File Showing Class IDs and Normalized Bounding-Box Coordinates	11
Fig. 3.5.2	YOLO Dataset Configuration and Class Definition in data.yaml	12
Fig. 4.1.1	Architecture of YOLOv11 showing the C3k2 backbone, SPPF, C2PSA attention module, FPN–PAN feature-fusion neck, and anchor-free detection head	14
Fig. 4.1.2	Architecture of YOLOv12 illustrating the R-ELAN backbone, efficient convolution, Area Attention, feature-fusion neck, and detection head	15
Fig. 4.1.3	Architecture of YOLO26 showing the C3k2 backbone, residual SPPF, C2PSA attention, FPN–PAN neck, and end-to-end NMS-free detection head	16
Fig. 5.1	YOLOv11n-based object detection system	18
Fig. 5.2	Sequential steps of the proposed YOLOv11n-based detection methodology	18
Fig. 5.4	Progressive Experimental Training Configurations for YOLO11n	20
Fig. 6.3	Training Environment and YOLO11n Model Verification	24
Fig. 6.4	Directory Structure of the Proposed YOLO11n-Based Auto-Rickshaw Detection Project	25
Fig. 6.5	YOLO Dataset Configuration and Class Definition for YOLO26n	26
Fig. 7.0.1	Representative detection results of the trained YOLOv11n model on unseen test images	30
Fig. 7.0.2	Training and validation performance curves of the YOLOv11n model	31

LIST OF TABLE

Table No.	Title	Page
Table 6.7.1	Performance comparison of the models on the validation dataset	28
Table 6.7.2	Performance comparison of the models on the test dataset	28

CHAPTER 1
INTRODUCTION

1.0 Introduction

Dhaka City is experiencing severe traffic congestion due to rapid urbanization, increasing vehicle numbers, limited road capacity, and challenges in conventional traffic management. Auto-rickshaws are a common mode of transportation in Bangladesh and are frequently involved in mixed-traffic environments. Effective detection and monitoring of these vehicles can provide valuable information for understanding traffic conditions and supporting intelligent transportation systems.

Recent advances in deep learning and computer vision have created opportunities for automated vehicle detection and real-time traffic surveillance. The YOLO (You Only Look Once) family of object detection models is particularly suitable for such applications because it performs object localization and classification within a single detection framework, enabling fast inference [1]. Research conducted specifically in Bangladesh has also demonstrated the effectiveness of YOLO-based models for detecting local vehicle types and analyzing traffic environments [2],[5].

However, Bangladesh has a distinctive traffic environment containing a wide variety of vehicles, including auto-rickshaws, rickshaws, CNGs, buses, motorcycles, and other locally common vehicles. Therefore, models trained on general-purpose datasets may not always provide reliable performance in local conditions. Region-specific datasets and model evaluation are consequently important for developing practical traffic-monitoring systems [3], [4].

In this research, a lightweight YOLO-based framework is proposed for real-time auto-rickshaw detection in Dhaka City. The study compares YOLO11n, YOLO12n, and YOLO26n and evaluates their performance using Precision, Recall, mAP, and F1-score. The detected vehicle information is further intended to support traffic-density estimation and congestion alerts. The objective is to identify an effective balance between detection accuracy and computational efficiency for practical urban traffic surveillance.

1.1 Background of the Research

Dhaka City faces severe traffic congestion due to rapid urbanization, increasing vehicle numbers, limited road capacity, and inefficient traffic management. Auto-rickshaws are a major part of the city's transportation system, but their large number and frequent movement can contribute to congestion. This research proposes a lightweight YOLOv11n-based real-time auto-rickshaw detection and traffic congestion alert system to support smarter traffic monitoring.

The major impacts of traffic congestion include:

1. Traffic Jam and Work-Time Loss: Long traffic jams increase travel time and reduce workers' productivity.
2. Medical Emergencies: Ambulances and patients may be delayed in reaching hospitals, which can be dangerous during emergencies.
3. Flight Misses and Commercial Loss: Traffic delays can cause people to miss flights and can delay business activities, deliveries, and important meetings.
4. Damage to Fresh Food: Long transportation delays can cause fruits, vegetables, fish, and other perishable foods to spoil.
5. Electricity-Service Delays: Traffic congestion can delay maintenance teams and emergency services, potentially increasing the duration of electricity-related disruptions.
6. Accidents from Poor Driving Practices: Inadequate driver training, traffic-rule violations, and unsafe driving can increase accident risks. Vehicle detection can help monitor traffic conditions.
7. Students Arriving Late for Exams: Unexpected traffic jams can cause students to reach examination centers late, potentially affecting their academic performance.

Therefore, an intelligent real-time traffic detection system can help identify increasing vehicle density and provide early congestion alerts. Comparing YOLO versions can also determine which model provides the best balance between accuracy, speed, and computational efficiency for real-time auto-rickshaw detection in Dhaka City.

1.2 Objectives of the Research

- To develop a lightweight YOLOv11n-based system for real-time auto-rickshaw detection in Dhaka City.
- To detect and monitor traffic congestion using surveillance-camera footage.
- To generate real-time traffic congestion alerts when vehicle density becomes high.
- To compare the performance of different YOLO versions for auto-rickshaw detection.
- To evaluate models based on accuracy, precision, recall, mAP, FPS, and inference time.
- To identify a model that provides a good balance between detection accuracy and real-time processing speed.
- To investigate how real-time congestion alerts can help reduce travel-time loss, emergency delays, commercial losses, food wastage, and students' delays.
- To develop a framework suitable for practical urban traffic surveillance in Dhaka City.

1.3 Rationale of the Research

Dhaka City experiences severe traffic congestion due to rapid urbanization, increasing numbers of vehicles, limited road capacity, and inefficient traffic management. Traffic congestion causes significant problems such as loss of working time, delayed medical emergencies, missed flights, commercial losses, food wastage, and students arriving late at examinations. Auto-rickshaws are a common mode of transportation in Dhaka, making their detection important for understanding urban traffic conditions.

Traditional traffic monitoring often depends on manual observation, which can be time-consuming and less effective for real-time monitoring. Therefore, an automated computer-vision-based system can provide a faster and more efficient solution. YOLO-based object detection models are suitable for real-time vehicle detection because of their speed and accuracy. In particular, YOLOv11n is a lightweight model that can potentially provide efficient detection with lower computational requirements. This research is therefore motivated by the need to develop a real-time auto-rickshaw detection and traffic congestion alert system and to compare different YOLO versions to identify an effective model for urban traffic surveillance in Dhaka City.

1.4 Required tools & Packages

The following software tools, libraries, and Python packages were used to develop, train, evaluate, and manage the proposed YOLOv11n-based object detection system.

1. Visual Studio Code - Code editing and project development.
2. Python - Programming and model implementation.
3. Ultralytics - YOLO model training and evaluation.
4. PyTorch - Deep learning and tensor computation.
5. Label Studio - Image annotation and labeling.
6. Albumentations - Image data augmentation.
7. NumPy - Numerical and array operations.
8. OpenCV (cv2) - Image and video processing.
9. Tqdm - Progress-bar visualization.
10. Os - File and directory management.
11. Virtual Environment (venv) - Isolated Python environment.
12. GitHub - Version control and code management.
13. IPython Kernel (ipykernel) - Jupyter Notebook execution.

1.5 Scope of the Research

This research focuses on developing a lightweight YOLOv11n-based real-time auto-rickshaw detection system for urban traffic surveillance in Dhaka City. The study will use surveillance or traffic-camera video to detect and monitor auto-rickshaws and estimate traffic conditions. It will also investigate the use of vehicle density and detection results to generate traffic congestion alerts. Different YOLO versions will be compared based on detection accuracy, precision, recall, mAP, FPS, and inference time. The research will mainly focus on auto-rickshaw detection and traffic congestion monitoring and will not cover complete traffic-control systems, accident prediction, or automatic traffic signal control. The proposed framework is intended to provide a foundation for efficient and low-cost real-time traffic monitoring that can support better traffic management in Dhaka City.

CHAPTER 2
LITERATURE REVIEW

2.0 Literature Review

1. Deep learning-based object detection has become an important research area for intelligent transportation systems, traffic surveillance, and automated vehicle monitoring. The development of YOLO introduced a single-stage detection approach in which object localization and classification are performed within a single neural network, enabling real-time detection [1]. This characteristic has made YOLO-based architectures particularly attractive for traffic-monitoring applications where both accuracy and processing speed are important.
2. In the context of Bangladesh, Alamgir et al. [2] conducted a comparative study of YOLOv3, YOLOv5s, and YOLOv5x for vehicle detection using 7,390 traffic images representing 21 vehicle categories. Their results showed that YOLOv5x achieved the best overall performance among the evaluated models. The study demonstrated the potential of YOLO architectures for vehicle detection in Bangladeshi traffic environments while also highlighting the importance of selecting an appropriate model for real-time applications.
3. The availability of Bangladesh-specific datasets has also been investigated. Tabassum et al. [3] introduced the Poribohon-BD dataset containing 9,058 annotated images of 15 native Bangladeshi vehicle categories. The dataset was developed to support vehicle classification and applications such as traffic surveillance, congestion prevention, and intelligent transportation systems. The study emphasized the importance of collecting vehicle images under different viewpoints, lighting conditions, and weather environments for developing robust computer-vision systems.
4. Saha et al. [4] further addressed the lack of large-scale region-specific datasets by introducing the Bangladeshi Native Vehicle Dataset (BNVD). The dataset contains 17,326 images and more than 81,000 annotated vehicle instances across 17 native vehicle classes. Several YOLO models were evaluated using this dataset, demonstrating that locally representative training data can improve vehicle-detection performance in Bangladeshi road environments.
5. Fahim [5] investigated the application of YOLOv9 for native vehicle detection in Dhaka. The study fine-tuned YOLOv9 using a Bangladesh-based dataset and achieved an mAP@0.5 of 93.4%. The research further proposed the use of CCTV-based vehicle detection for understanding traffic congestion, mobility patterns, and traffic surveillance. This study is

particularly relevant to the present research because it demonstrates how deep-learning-based vehicle detection can contribute to intelligent transportation systems in Dhaka.

6. More recent research has focused on comparing different YOLO architectures under Bangladeshi traffic conditions. Hossain et al. [6] evaluated six YOLO variants using a dataset containing 29 vehicle classes, including locally common vehicles such as battery rickshaws and CNGs. Their results showed a clear trade-off between detection accuracy and inference speed. Although YOLO11x achieved the highest detection performance, lightweight models such as YOLO11n provided substantially faster inference. This finding supports the importance of considering computational efficiency when developing real-time traffic-monitoring systems.
7. Research specifically focused on three-wheelers has also been conducted. Nabil, Kundu, and Karmoker [7] developed a real-time three-wheeler detection system using YOLO models and roadside CCTV footage. Their study reported that YOLOv8m achieved 88.2% accuracy, compared with 83.6% for YOLOv8n. The successful deployment of the trained model on real-time CCTV footage demonstrates the feasibility of using YOLO-based systems for three-wheeler monitoring in practical traffic environments.
8. Sukanto et al. [8] specifically investigated automatic auto-rickshaw detection in Bangladesh. Their research addressed the difficulty of monitoring auto-rickshaws because of their visual similarity to other road vehicles and the limitations of manual video analysis. Using 1,730 annotated images captured under different traffic conditions, the researchers developed a YOLOv8-based detection system and achieved an mAP@0.5 of 83.447%. This work is particularly close to the present research because both studies focus on automatic auto-rickshaw detection in Bangladesh.
9. Ali et al. [9] investigated real-time object detection for Bangladeshi road environments using a dataset containing multiple local vehicle categories. Their work further demonstrated the challenges associated with variations in lighting, vehicle appearance, and road environments. Such variations are important considerations when designing a detection system intended for real-world deployment.
10. Udoy et al. [10] also investigated real-time traffic vehicle detection in Bangladesh using YOLO-based approaches. Their work demonstrates the applicability of deep-learning-based detection to local traffic environments and provides further evidence that automated vehicle detection can reduce dependence on manual traffic observation.
11. Beyond vehicle detection itself, computer vision has increasingly been applied to broader traffic-surveillance tasks. Majumder et al. [11] investigated real-time wrong-way driving detection using deep learning in a Bangladeshi traffic environment. Such research

demonstrates that object-detection systems can provide information beyond simple vehicle classification and can support automated traffic monitoring and road-safety applications.

12. The development of newer YOLO architectures has continued to focus on improving the balance between accuracy and computational efficiency. YOLOv12 introduces an attention-centric architecture designed to improve feature representation while maintaining real-time performance [12]. YOLO26 further introduces end-to-end, NMS-free inference and a lighter detection head, with the objective of improving efficiency for real-time deployment [13]. These developments demonstrate the continuing importance of evaluating different YOLO versions according to the requirements of a specific application.

Overall, previous research demonstrates that YOLO-based models are effective for vehicle detection and traffic surveillance, particularly when trained on datasets representing local road environments. However, most existing studies have focused either on general vehicle detection, individual YOLO architectures, or specific traffic-monitoring tasks. Comparatively limited research has combined lightweight auto-rickshaw detection, comparative evaluation of recent YOLO versions within a single framework designed for Dhaka City. Therefore, the present research focuses on developing a lightweight YOLO11n-based framework and comparing YOLO11n, YOLO12n, and YOLO26n to determine a suitable balance between detection accuracy and real-time computational efficiency.

CHAPTER 3
DATA COLLECTION

3.0 Data Collection

The first step in developing a deep learning-based traffic surveillance system is to obtain a suitable dataset for training and evaluating the proposed models. In this research, the Auto-RickshawImageBD open-source dataset was used for auto-rickshaw detection. The original dataset contains 1,331 high-resolution images collected from urban and suburban areas of Bangladesh. To improve the diversity and robustness of the training data, data augmentation was performed using the Albumentations library. After augmentation, the total number of images increased from 1,331 to 3,671 images.

3.1. Dataset Sources and Regions

The original Auto-RickshawImageBD dataset contains images collected from seven cities and regions of Bangladesh:

1. Dhaka
2. Narayanganj
3. Gazipur
4. Savar
5. Cumilla
6. Narsingdi
7. Khulna

The images were captured using smartphones and portable cameras at different times of the day. Both daytime and nighttime images were included to represent different lighting conditions and real-world urban traffic environments.

The dataset is distributed under the Creative Commons Attribution 4.0 International (CC BY 4.0) license.

Dataset Source:*Auto-RickshawImageBD: An Image Dataset for Auto-Rickshaw Detection, Mendeley Data, Version 1, DOI: 10.17632/bg6wvvhsjh.1.*

3.2. Data Augmentation

Since the original dataset contains a limited number of images, Albumentations was used to increase the diversity of the training data. Different augmentation techniques were applied to simulate real-world variations in traffic scenes, lighting, scale, and image orientation. These augmentations help the model learn more robust visual features and reduce the possibility of overfitting.

```
transform = A.Compose(
  [
    A.HorizontalFlip(
      p=0.5
    ),
    A.RandomBrightnessContrast(
      brightness_limit=0.3,
      contrast_limit=0.3,
      p=0.7
    ),
    A.Rotate(
      limit=5,
      p=0.3
    ),
    A.RandomGamma(
      gamma_limit=(80, 120),
      p=0.3
    ),
    A.GaussianBlur(
      blur_limit=(3, 5),
      p=0.2
    ),
    A.HueSaturationValue(
      hue_shift_limit=10,
      sat_shift_limit=20,
      val_shift_limit=15,
      p=0.3
    ),
    A.RandomShadow(
      p=0.2
    ),
  ],
  bbox_params=A.BboxParams(
    format="yolo",
```

```
    label_fields=["class_labels"],
    min_visibility=0.2
  )
)
```

The augmented dataset was subsequently used for training and evaluating the YOLO-based models. Special emphasis was given to YOLOv11n because of its lightweight architecture and suitability for real-time auto-rickshaw detection and traffic monitoring applications.

3.3 Annotation Protocol

The original images were annotated using Label Studio following the YOLO object-detection format. The dataset contains two classes: auto_rickshaw (Class 1) and non_auto_rickshaw (Class 2). The bounding-box annotations were manually verified to ensure reliable and accurate training data.

3.4 Data Segmentation

To increase the size and variability of the dataset, Albumentations were applied to the original 1,331 images. Appropriate image augmentation techniques were used to introduce variations in the training samples while preserving the original object annotations. Through this augmentation process, the dataset was expanded to 3,671 images.

The augmented dataset provides greater variation in vehicle appearance, image orientation, scale, and environmental conditions. This helps improve the generalization capability of the YOLO models and reduces the risk of overfitting. The final dataset of 3,671 images was used for the development and evaluation of the proposed real-time auto-rickshaw detection framework, with particular focus on YOLOv11n and comparative evaluation across different YOLO versions.

3.5 Data Preparation

To train and evaluate the proposed object detection models, the Auto-RickshawImageBD dataset was prepared and structured in the YOLO object detection format. The original dataset contained 1,331 high-resolution images, which were expanded to 3,671 images through data augmentation using the Albumentations library. Each image was paired with a corresponding annotation file containing the bounding-box information for the detected vehicles.

Each annotation file follows the standard YOLO format, where every object is represented in a single line using five values:

[class_id] [x_center] [y_center] [width] [height]

The bounding-box coordinates and dimensions were normalized between 0 and 1 according to the width and height of the corresponding image. The dataset contains two classes: auto_rickshaw (Class 1) and non-auto_rickshaw (Class 2).

The prepared dataset was divided into training, validation, and testing subsets in a 70:15:15 ratio, respectively, to ensure systematic model development and unbiased performance evaluation. A data.yaml configuration file was used to define the dataset paths and class information required by the YOLO models. This structured dataset was subsequently used for training and comparing different YOLO versions, with particular emphasis on the lightweight YOLOv11n model.

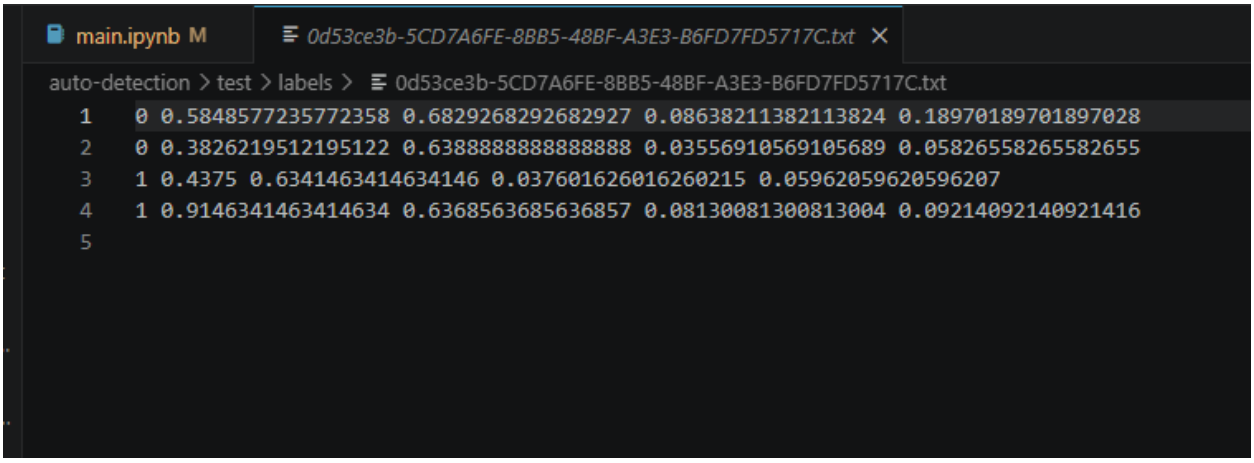


Figure 3.5.1: YOLO-Formatted Annotation File Showing Class IDs and Normalized Bounding-Box Coordinates

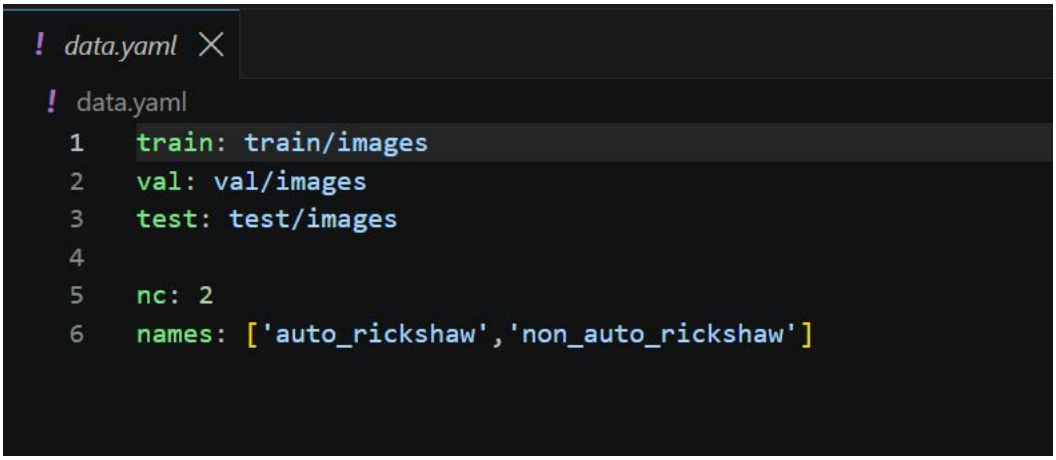


Figure 3.5.2: YOLO Dataset Configuration and Class Definition in data.yaml

CHAPTER 4
MODEL ARCHITECTURE

4.0 Model Architecture

In this research, three lightweight YOLO models, YOLOv11n, YOLOv12n, and YOLOv26n, were selected for comparative evaluation. YOLO follows a single-stage object detection architecture that primarily consists of a Backbone, Neck, and Detection Head. The Backbone extracts important visual features from input images, the Neck combines features at different scales, and the Detection Head predicts object classes, bounding boxes, and confidence scores.
All three models were trained using the same augmented dataset and evaluated under the same experimental conditions. Their performance was compared using Precision, Recall, mAP, FPS, and inference time. Based on the experimental results, YOLOv11n was selected as the final model for the proposed real-time auto-rickshaw detection framework.

4.1 Models Selected

Among the three considered models, YOLOv11n was selected as the primary model for this research. The selection was based on its lightweight architecture, computational efficiency, and suitability for real-time object detection. YOLOv11n was used for detecting auto_rickshaws and non_auto_rickshaw from traffic images. The model was trained using the augmented dataset and evaluated using standard metrics such as Precision, Recall, mAP, FPS, and inference time. The selected YOLOv11n model was then used as the main detection model for the proposed real-time auto-rickshaw detection and traffic monitoring framework in Dhaka City.

4.1.1 Yolo v11

YOLOv11 is a real-time, single-stage object detection model that follows the conventional backbone–neck–head architecture. The backbone uses C3k2 blocks for hierarchical feature extraction, followed by Spatial Pyramid Pooling-Fast (SPPF) for multi-scale contextual feature aggregation and C2PSA (C2 with Position-Sensitive Attention) for enhanced spatial feature representation. The neck employs Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) structures to combine features at different spatial scales. Finally, an anchor-free, decoupled detection head predicts object classes and bounding boxes at multiple scales. This architecture

provides a balance between detection accuracy, computational efficiency, and real-time inference, making YOLOv11 suitable for urban vehicle detection applications.

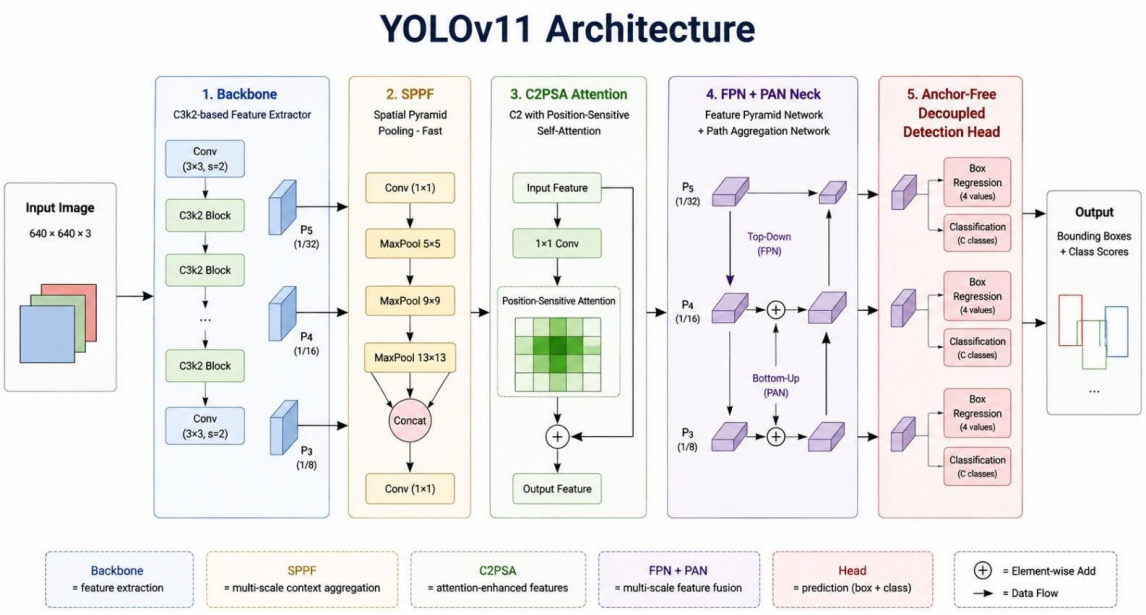


Figure 4.1.1: Architecture of YOLOv11 showing the C3k2 backbone, SPPF, C2PSA attention module, FPN–PAN feature-fusion neck, and anchor-free detection head.

4.1.2 Yolo v12

YOLOv12 is an attention-centric real-time object detection model designed to improve feature representation while maintaining low inference latency. Its architecture introduces Area Attention (A^2) to capture long-range spatial relationships more efficiently than conventional self-attention. The model also incorporates Residual Efficient Layer Aggregation Network (R-ELAN) for improved feature extraction and information flow, together with efficient convolutional operations. The feature maps are subsequently processed through the feature-fusion neck and detection head to generate object classes and bounding-box predictions. The integration of efficient attention mechanisms enables YOLOv12 to better model complex visual contexts while retaining real-time detection capability.

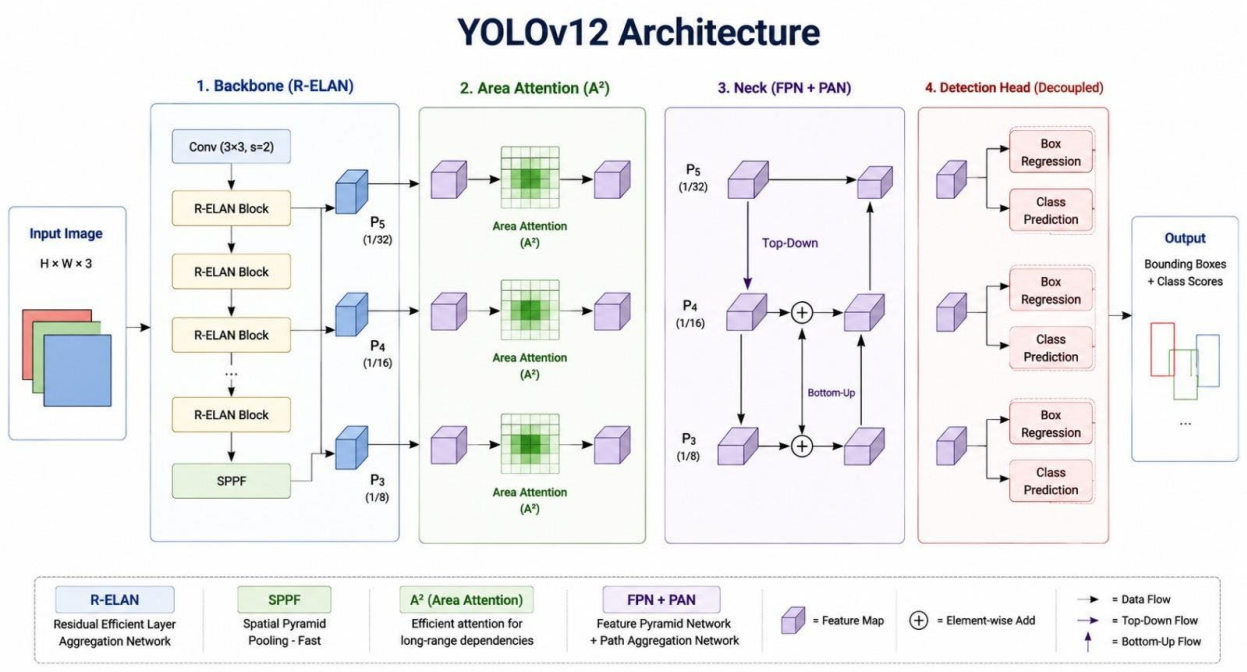


Figure 4.1.2: Architecture of YOLOv12 illustrating the R-ELAN backbone, efficient convolution, Area Attention, feature-fusion neck, and detection head.

4.1.3 Yolo v26

YOLO26 is a recent real-time object detection architecture designed with a deployment-oriented approach. Its backbone uses C3k2 blocks followed by SPPF with a residual shortcut and C2PSA attention for efficient multi-scale feature extraction. The neck follows an FPN–PAN feature-fusion structure. A major architectural improvement is its native end-to-end, NMS-free detection capability, where the default one-to-one detection head directly produces final predictions without requiring Non-Maximum Suppression (NMS). YOLO26 also removes Distribution Focal Loss (DFL) from its box-regression pipeline, reducing detection-head complexity. An alternative one-to-many head is available when conventional NMS-based inference is preferred.

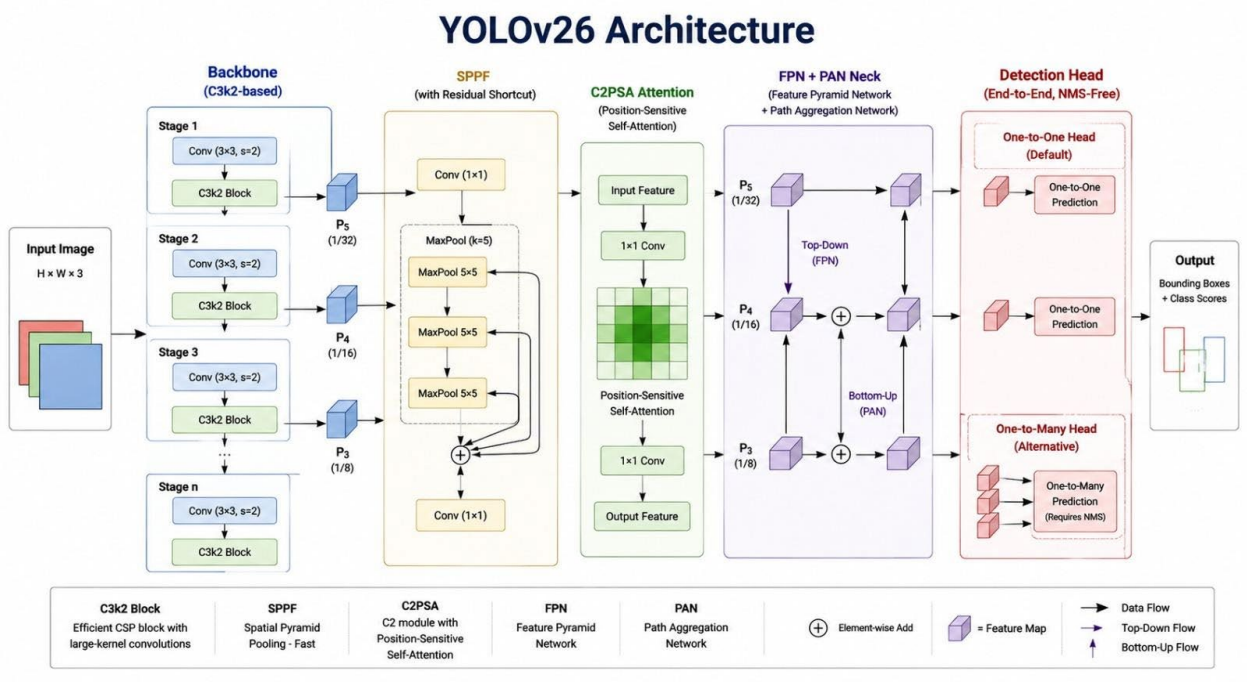


Figure 4.1.3: Architecture of YOLO26 showing the C3k2 backbone, residual SPPF, C2PSA attention, FPN–PAN neck, and end-to-end NMS-free detection head.

CHAPTER 5
METHODOLOGY

5.0 Methodology

The proposed YOLOv11n-based object detection methodology consists of several sequential stages. First, images are collected from the target environment and annotated with bounding boxes and class labels. The dataset is then cleaned, resized, augmented, and divided into training, validation, and testing sets. Next, the pretrained YOLOv11n model is initialized and fine-tuned using the prepared dataset. During training, the model learns object localization and classification through iterative forward and backward propagation. The trained model is evaluated using Precision, Recall, F1-score, mAP, and inference speed (FPS). Finally, the best-performing model is used for testing on unseen images, and the detection results are analyzed. The overall YOLOv11n architecture consists of a C3k2-based backbone, SPPF, C2PSA attention module, FPN–PAN neck, and anchor-free decoupled detection head

5.1 YOLOv11n-Based System Block Diagram

The proposed system takes input images or video frames from the target environment and processes them through the YOLOv11n object detection model. The system performs feature extraction, multi-scale feature fusion, and object classification and localization to identify the target objects. The final output consists of detected objects with bounding boxes, class labels, and confidence scores.

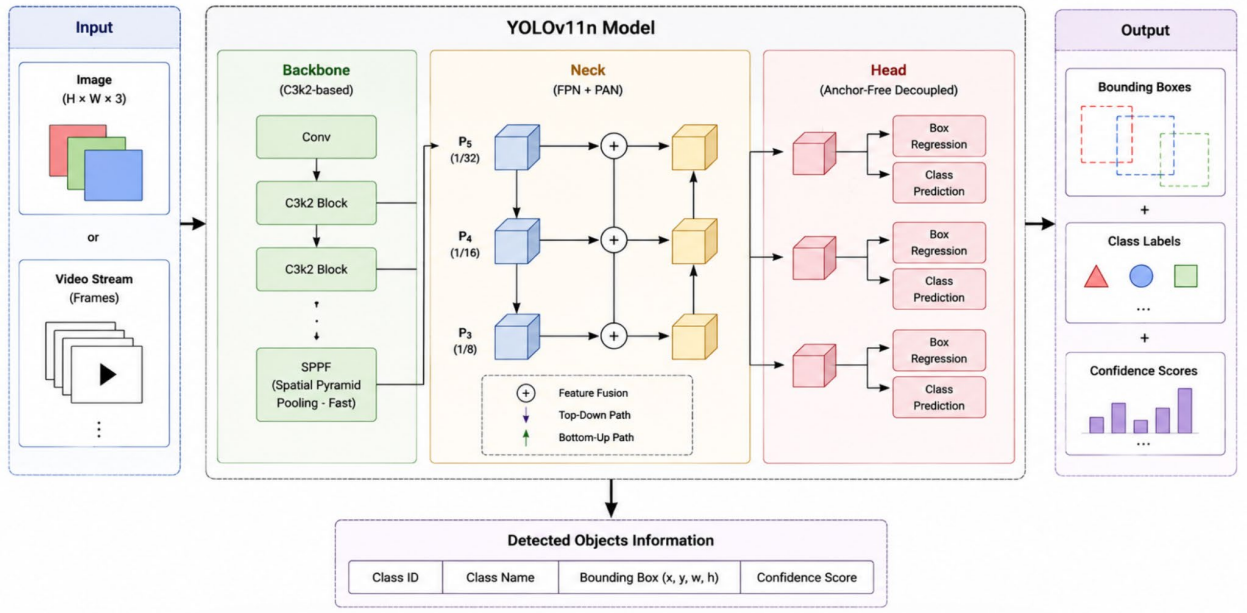


Fig. 5.1: YOLOv11n-based object detection system

5.2 Methodology Pipeline

The proposed methodology consists of sequential stages including data collection, data annotation, preprocessing, dataset splitting, model initialization, model training, validation, testing, and performance evaluation. This pipeline provides a systematic workflow from raw image collection to final object detection results.

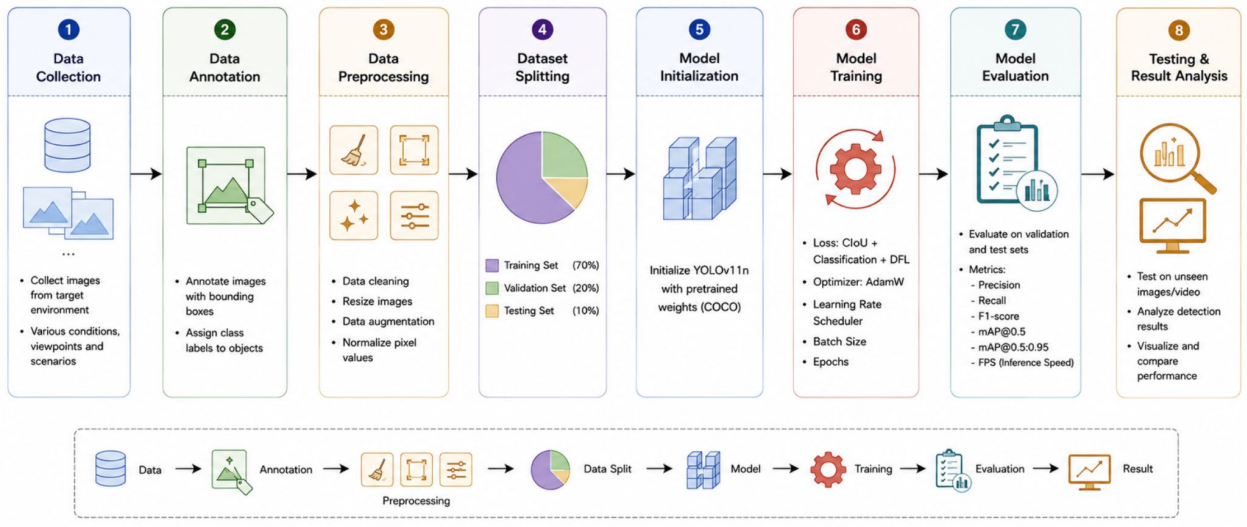


Fig. 5.2: Sequential steps of the proposed YOLOv11n-based detection methodology

5.3 Libraries and Frameworks

The following libraries and frameworks are used for implementing the proposed system:

- Ultralytics YOLO, YOLOv11n model training and inference
- PyTorch — Deep learning framework and model computation
- OpenCV — Image and video processing
- NumPy — Numerical computation and array processing
- Matplotlib — Visualization of training and evaluation results
- Pandas — Dataset and performance analysis

5.4 Experimental Setup

To investigate the effect of training data size, number of training epochs, and batch size on model performance, a series of experiments was conducted progressively. The training configuration was modified at each stage based on observations from the preceding experiment.

In the first experiment, the model was trained using a relatively small dataset consisting of 200 images. The model was trained for 30 epochs with a batch size of 4. This initial experiment was performed to establish a baseline and verify the overall training procedure.

In the second experiment, the training dataset was increased to 1,000 images to provide the model with a more diverse set of training samples. The number of training epochs was also increased to 50, while the batch size was changed to 8. This experiment was intended to examine whether increasing the amount of training data and using a larger batch size could improve the model's learning capability.

For the third experiment, the dataset was further expanded to 1,300 images, and the training duration was substantially increased to 200 epochs. The batch size was maintained at 8. The purpose of this experiment was to allow the model more training iterations and determine whether the additional data and extended training period would lead to improved performance.

Finally, in the fourth and final experiment, the model was trained on the 1,300-image dataset for 300 epochs with a batch size of 4. The batch size was reduced from 8 to 4 while the number of epochs was increased from 200 to 300. This configuration was selected as the final training setup based on the observations obtained from the earlier experiments.

Experiment	Number of Images	Epochs	Batch Size
Experiment 1	200	30	4
Experiment 2	1,000	50	8
Experiment 3	1,300	200	8
Experiment 4 (Final)	1,300	300	4

Figure 5.4: Progressive Experimental Training Configurations for YOLO11n

Overall, the experiments followed a progressive training strategy, beginning with a small dataset and short training duration and gradually increasing the dataset size and number of epochs. The final experiment used the largest dataset and longest training duration, while a smaller batch size was adopted to investigate its effect on the model's learning behavior.

5.5 Evaluation Metrics

The performance of the trained YOLOv11n model was evaluated using standard object detection metrics, including Precision, Recall, mAP@0.5, and mAP@0.5:0.95. Precision measures the proportion of correctly predicted objects among all detected objects, while Recall measures the proportion of actual objects successfully detected by the model. mAP@0.5 represents mean Average Precision at an IoU threshold of 0.50, whereas mAP@0.5:0.95 represents the mean Average Precision calculated over IoU thresholds ranging from 0.50 to 0.95 with a step of 0.05. In addition, the F1-score can be used to provide a combined measure of Precision and Recall.

The evaluation metrics are defined as:

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

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

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

where TP (True Positive) represents correctly detected objects, FP (False Positive) represents incorrectly detected objects, and FN (False Negative) represents objects that were present but not detected.

TP (True Positive): Correctly detects an actual auto-rickshaw or non-auto-rickshaw object.

TN (True Negative): Correctly identifies an image/object as not belonging to the target class.

FP (False Positive): Incorrectly detects an object as a target class when it is not.

FN (False Negative): Fails to detect an actual target object.

Precision: Measures how many of the detected objects are actually correct.

Recall: Measures how many of the actual objects are successfully detected.

mAP@0.5: Mean Average Precision at an IoU threshold of 0.50.
mAP@0.5:0.95: Mean Average Precision averaged across IoU thresholds from 0.50 to 0.95, with a step of 0.05.

For the proposed YOLOv11n model, the primary performance measures are Precision, Recall, mAP@0.5, and mAP@0.5:0.95, while inference speed can additionally be reported to assess the model's suitability for real-time detection.

CHAPTER 6
MODEL TRAINING

6.0 YOLOv11n Model Training

The YOLOv11n model was trained using the Ultralytics framework for object detection and classification of auto-rickshaw and non-auto-rickshaw images. The training procedure consisted of several stages, including environment setup, package installation and verification, model initialization, dataset configuration, and model training. The overall procedure was designed to ensure that the required software packages, model architecture, and dataset configuration were correctly prepared before training .

6.1 Environment and Package Setup

The training environment was first prepared by installing and importing the required Python libraries. The Ultralytics package was used as the primary framework for implementing YOLOv11n. Additional libraries required for dataset handling, image processing, and result visualization were also loaded as necessary.

After installing the required packages, the installed Ultralytics version was checked to verify that the correct framework version was available in the training environment. This step was important to ensure compatibility between the YOLOv11n model, training commands, and the configured dataset.

The basic environment verification procedure can be summarized as follows:

1. Install the required Python packages.
2. Import the Ultralytics library.
3. Verify the installed Ultralytics version.
4. Confirm that the YOLOv11n model can be loaded successfully.
5. Prepare and verify the dataset configuration before starting training.

6.2 YOLOv11n Model Initialization

After verifying the training environment, the YOLOv11n model was imported using the Ultralytics framework. The YOLOv11n (nano) model was selected because of its relatively lightweight architecture and computational efficiency, making it suitable for the intended detection task.

The pretrained YOLOv11n model was loaded before training. Loading the pretrained model provided an initial set of learned parameters, which could then be fine-tuned using the prepared dataset. This approach allowed the model to start from previously learned visual representations rather than learning all features from the beginning.

6.3 Environment and Package Verification

The first step was to verify that the required software packages were installed correctly and could be imported without errors. PyTorch, Ultralytics, and the required Python modules were imported into the training environment. PyTorch was used as the underlying deep learning framework, while Ultralytics provided the implementation of the YOLO model and training pipeline.

The installed PyTorch version was checked to confirm that the expected version was available in the environment. In addition, a random tensor was generated using PyTorch to verify that the library was functioning correctly and capable of performing basic tensor operations. The existence of the YOLO model weight file was also checked before attempting to load the model.

The Ultralytics package version was then displayed to verify the installed framework version. This version check was performed to ensure that the training environment was properly configured and that the required YOLO model implementation was available.

The following checks were performed during environment verification:

- Importing PyTorch and checking its installed version.
- Generating a random PyTorch tensor to verify that PyTorch was functioning correctly.
- Importing the OS module and checking whether the required model weight file existed.
- Importing Ultralytics and checking its installed version.
- Importing the YOLO class from Ultralytics.
- Loading the YOLO model to verify that the model architecture and weight file could be accessed successfully.

These preliminary checks helped identify potential package, version, or model-file issues before the computationally intensive training process was started.

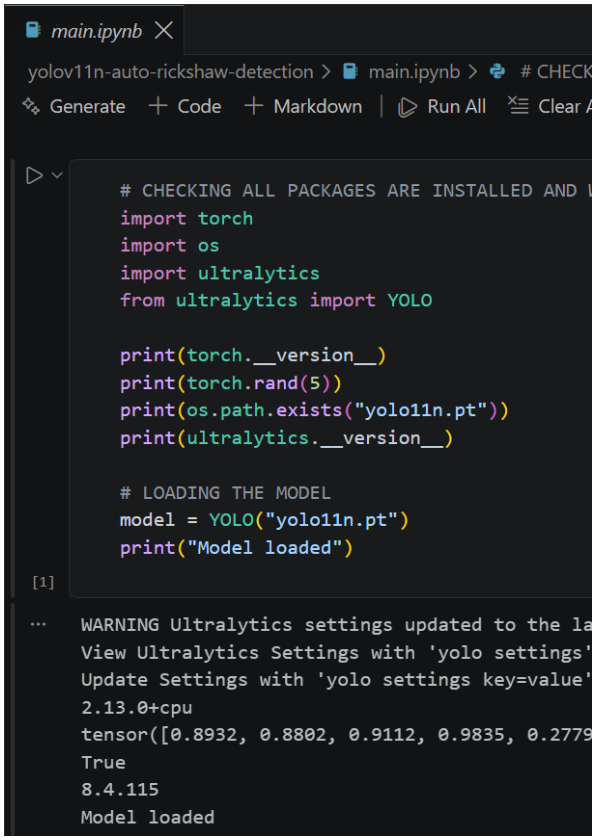


Figure 6.3: Training Environment and YOLO11n Model Verification

6.4 Model Loading

After confirming that the required packages were installed and functioning correctly, the YOLO model was loaded using the YOLO class provided by the Ultralytics framework. The model weight file was specified during initialization, and successful loading was verified through a confirmation message.

This step ensured that the selected YOLO model could be initialized correctly within the configured training environment before proceeding to dataset preparation and training.

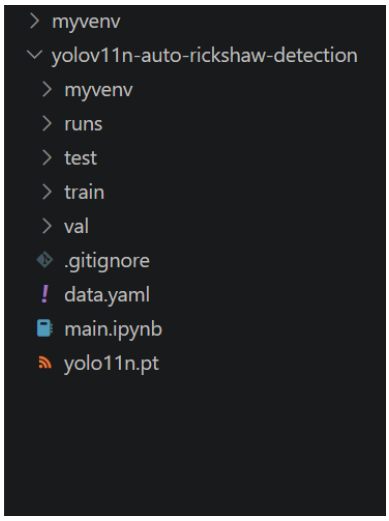


Figure 6.4: Directory Structure of the Proposed YOLO11n-Based Auto-Rickshaw Detection Project

6.5 Dataset Configuration

Before initiating training, the dataset was configured using a data.yaml file. This file provided the YOLO training framework with the necessary information about the dataset structure, class names, and locations of the training and validation data.

For this study, two classes were defined:

- Class 0: auto_rickshaw
- Class 1: non_auto_rickshaw

The dataset configuration was therefore defined with a total of two classes. The data.yaml file also specified the paths to the training and validation datasets so that the YOLOv11n training procedure could automatically locate the required images and corresponding annotation files.

A simplified representation of the dataset configuration is shown below:

path: /path/to/dataset
train: train/images
val: val/images
names:
 0: auto_rickshaw
 1: non_auto_rickshaw

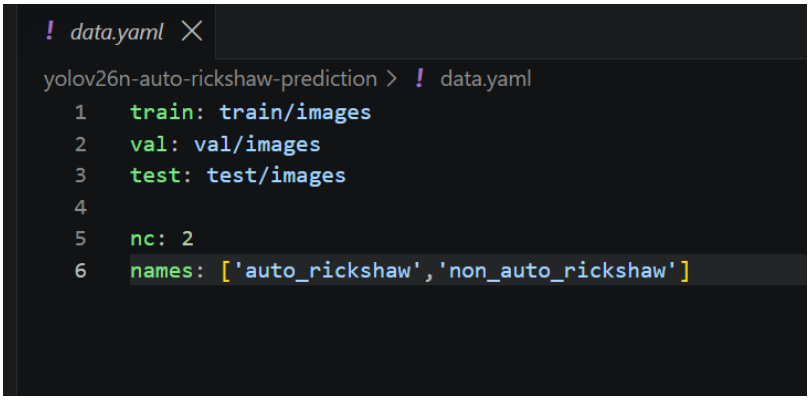


Figure 6.5: YOLO Dataset Configuration and Class Definition for YOLO26n

The actual dataset paths were configured according to the directory structure used in the experimental environment.

6.6 Model Training Procedure

After completing the environment verification, loading the YOLOv11n model, and configuring the dataset through the data.yaml file, the model was trained using the Ultralytics training interface. The training configuration was defined to control the number of training epochs, input image size, batch size, computational device, number of worker processes, and early-stopping patience.

The training process was executed using the following configuration:

```
train_results = model.train(
   data="./data.yaml",
   epochs=300,
   imgsz=512,
   batch=8,
   device="cpu",
   workers=0,
   patience=20
)
```

The data parameter specifies the location of the data.yaml file, which contains the dataset paths and the two class definitions, auto_rickshaw and non_auto_rickshaw. The epochs parameter was set to

300, meaning that the complete training dataset was processed up to 300 times during the training process.

The input image size was set to 512 × 512 pixels using the imgsz=512 parameter. This ensured that the input images were resized to a consistent spatial resolution before being processed by the model. A batch size of 8 was selected using batch=8, meaning that eight images were processed together in each training iteration.

The device="cpu" parameter specifies that the training was performed using the CPU rather than a GPU. The number of data-loading worker processes was set to 0 using workers=0. This configuration was used to ensure compatibility with the available computational environment and to avoid potential multiprocessing-related issues during training.

The patience=20 parameter was used to control early stopping. If the validation performance did not show improvement for 20 consecutive epochs, the training process could be stopped before reaching the maximum number of 300 epochs. This helped prevent unnecessary training when further iterations were no longer producing meaningful improvements in validation performance.

6.7 Model Evaluation

The performance of the proposed object detection system was evaluated using Precision, Recall, mAP@0.5, and mAP@0.5:0.95 on both the validation and test datasets. Three lightweight YOLO models, YOLO11n, YOLO12n, and YOLO26n, were trained and compared under two experimental configurations: 300 epochs with 640 × 640 image size and batch size 8, and 200 epochs with 512 × 512 image size and batch size 4. The training was performed using CPU with zero workers, as specified in the experimental configuration.

Model	mAP@0.5	mAP@0.5:0.95	Precision	Recall	F1-score
YOLO26n	78.59%	42.83%	76.09%	69.48%	72.64%
YOLO12n	69.63%	37.44%	69.05%	66.09%	67.54%
YOLO11n	83.27%	45.51%	85.32%	74.67%	79.64%

Table 6.7.1: Performance comparison of the models on the validation dataset

Model	mAP@0.5	mAP@0.5:0.95	Precision	Recall	F1-score
YOLO26n	79.23%	44.45%	78.55%	68.00%	72.90%
YOLO12n	62.37%	33.93%	63.62%	61.38%	62.48%
YOLO11n	83.93%	47.64%	84.19%	73.82%	78.66%

Table 6.7.2: Performance comparison of the models on the test dataset

The experimental results demonstrate that YOLO11n achieved the best overall detection performance among the three evaluated models. On the validation dataset, YOLO11n achieved an mAP@0.5 of 83.27%, mAP@0.5:0.95 of 45.51%, Precision of 85.32%, Recall of 74.67%, and an F1-score of 79.64%.

On the test dataset, YOLO11n further achieved the highest performance, with an mAP@0.5 of 83.93% and mAP@0.5:0.95 of 47.64%. Its Precision, Recall, and F1-score were 84.19%, 73.82%, and 78.66%, respectively. These results indicate that YOLO11n provides a better balance between object localization and classification compared with YOLO12n and YOLO26n under the same experimental configuration.

YOLO26n demonstrated the second-best performance, obtaining a test mAP@0.5 of 79.23%, mAP@0.5:0.95 of 44.45%, and an F1-score of 72.90%. Although YOLO26n performed better than YOLO12n, its overall detection accuracy remained lower than YOLO11n.

YOLO12n produced the lowest performance in the present experiment. Its test mAP@0.5 was 62.37%, mAP@0.5:0.95 was 33.93%, and F1-score was 62.48%. The comparatively lower Precision and Recall indicate that YOLO12n was less effective at detecting the target objects under the current dataset and training configuration.

Therefore, based on the test-set results, particularly the mAP@0.5, mAP@0.5:0.95, and F1-score, YOLO11n was selected as the best-performing model for the proposed object detection system. It achieved the highest values for all major evaluation metrics among the three models.

CHAPTER 7
TESTING AND RESULT ANALYSIS

7.0 Testing and Result Analysis

After completing the training and validation stages, the trained YOLOv11n model was tested on an independent test dataset containing previously unseen images. The purpose of this stage was to examine the model's practical detection capability and analyze its behavior under different road environments, object sizes, viewing angles, and background conditions.

The qualitative testing results demonstrate that the trained YOLOv11n model successfully detected auto-rickshaws in various urban road scenes. The detected objects were represented using bounding boxes, class labels, and confidence scores. The model was able to identify both relatively large and small auto-rickshaws appearing at different distances and positions within the images. Representative detection results are shown in Fig. 5.8.1.

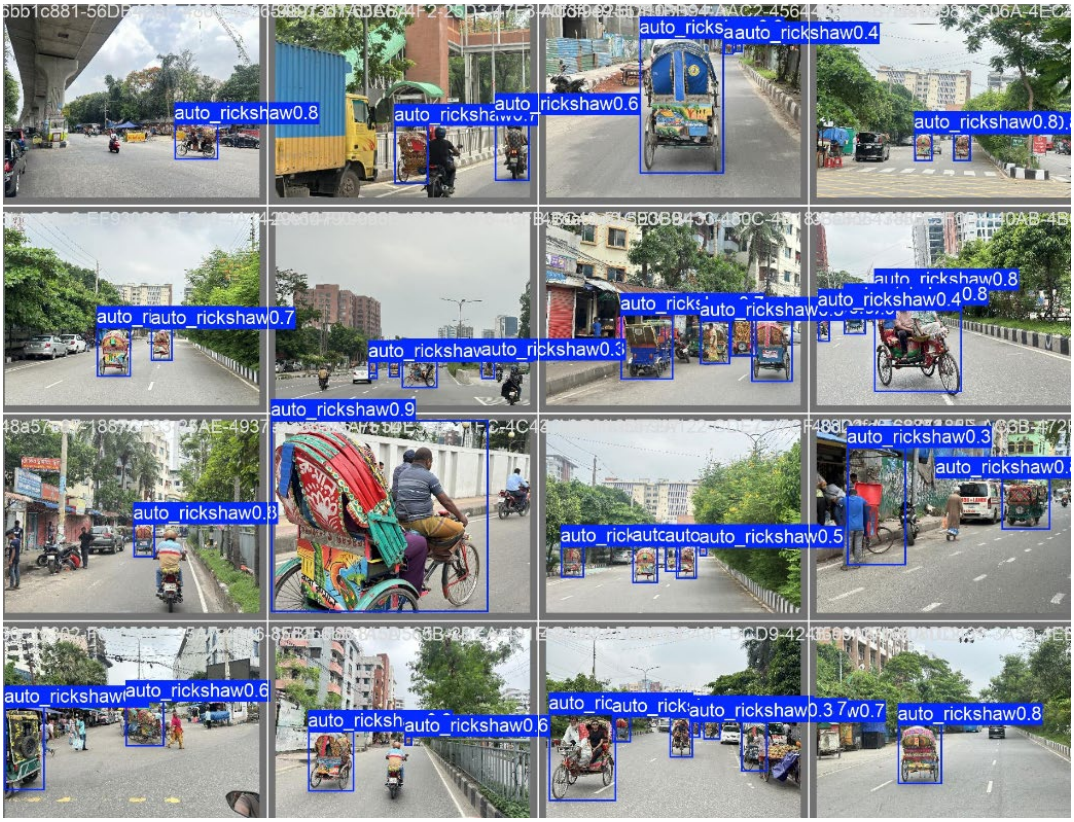


Fig. 7.0.1: Representative detection results of the trained YOLOv11n model on unseen test images

The training and validation curves were further analyzed to understand the learning behavior of the model. As presented in Fig. 5.8.2, the training box loss, classification loss, and DFL loss generally decreased during the training process. The reduction in these losses indicates progressive improvement in bounding-box localization, object classification, and regression performance. The Precision and Recall curves also show an overall improving trend throughout training. Similarly, the mAP@0.5 and mAP@0.5:0.95 curves increased during the training process and reached higher values toward the later stages, indicating improved detection performance.

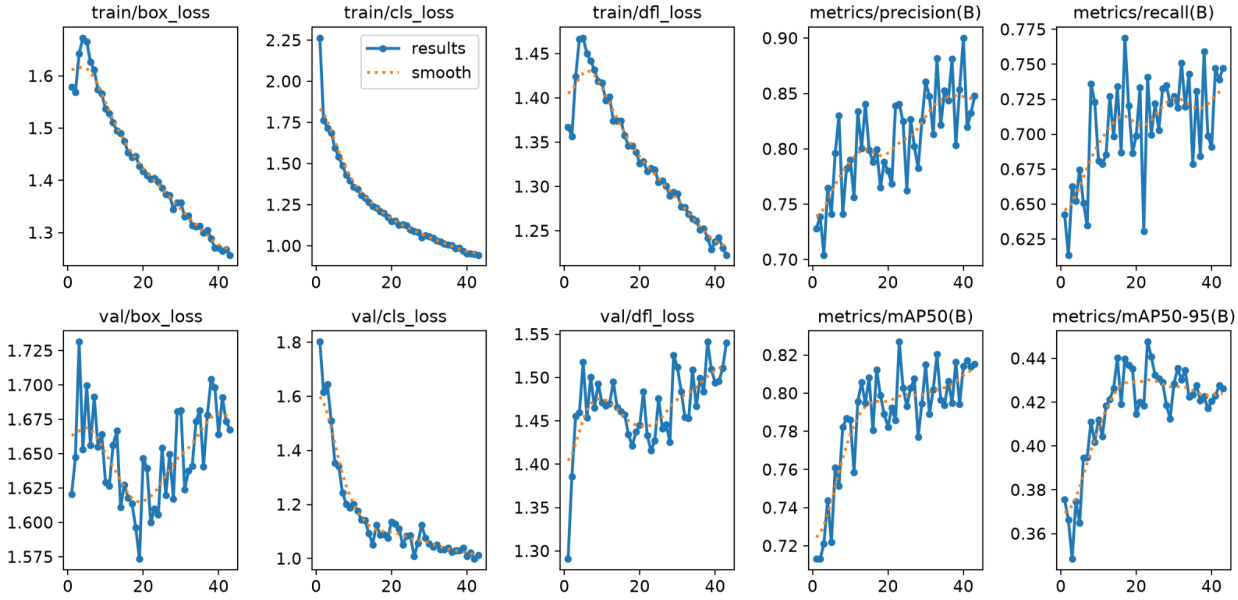


Fig. 7.0.2: Training and validation performance curves of the YOLOv11n model

The F1-confidence curve was analyzed to determine an appropriate confidence threshold for balancing Precision and Recall. As shown in Fig. 5.8.3, the F1-score reached approximately 0.79 at a confidence threshold of 0.457. This indicates that a confidence threshold around 0.457 provides a favorable balance between correctly detected auto-rickshaws and false or missed detections for the evaluated dataset.

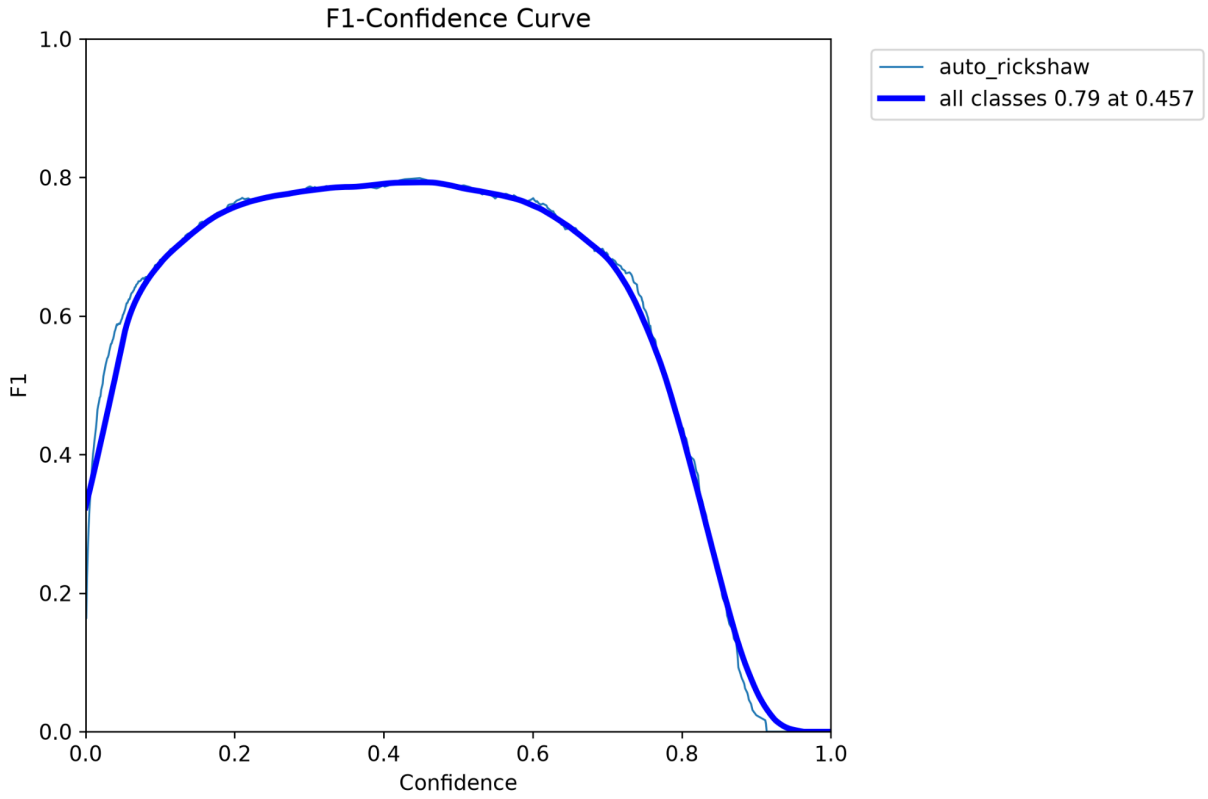


Fig. 7.0.3: F1-score versus confidence threshold for the YOLOv11n model

The normalized confusion matrix was also examined to understand the classification behavior of the model. As shown in Fig. 5.8.4, approximately 97% of the true auto-rickshaw instances were correctly associated with the auto_rickshaw class, while approximately 3% were associated with the background category. The matrix also indicates some false-positive predictions involving background regions. This suggests that the model performs well in recognizing the target class, although visually similar background regions may occasionally result in incorrect detections.

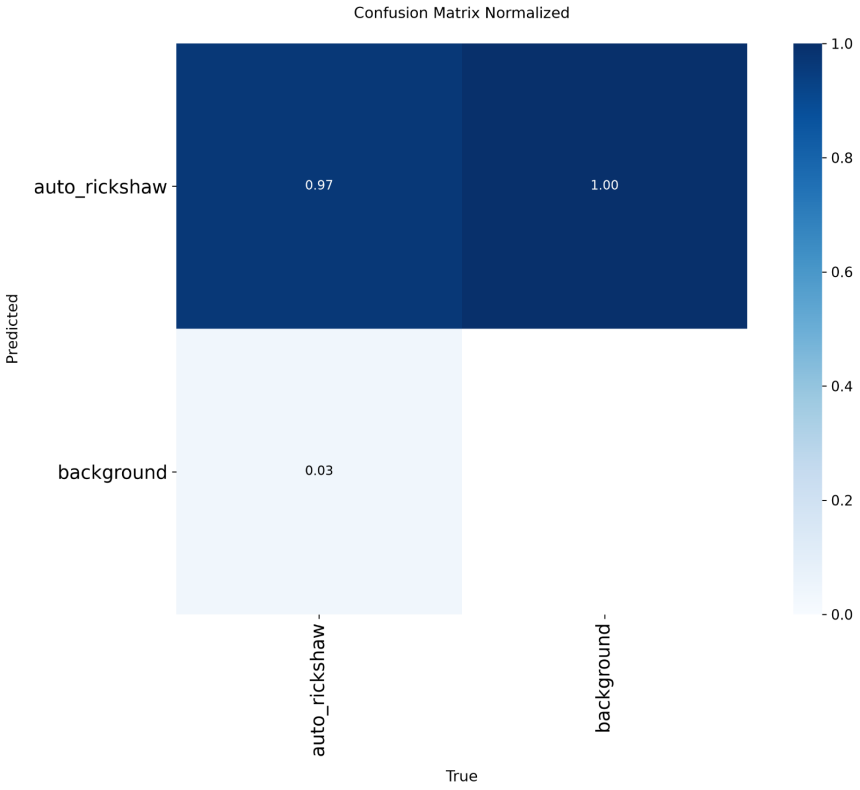


Fig. 7.0.4: Normalized confusion matrix of the YOLOv11n model

The quantitative comparison of YOLO11n, YOLO12n, and YOLO26n was presented earlier in the Evaluation Metrics section. Based on that comparison, YOLO11n achieved the strongest overall performance on the test dataset, with mAP@0.5 of 83.93%, mAP@0.5:0.95 of 47.64%, Precision of 84.19%, Recall of 73.82%, and F1-score of 78.66%. These results, together with the qualitative detection results and training curves, demonstrate that YOLO11n provides reliable detection performance for the proposed auto-rickshaw detection task.

Overall, the testing results indicate that the proposed YOLO11n model can effectively detect auto-rickshaws in diverse urban traffic scenes. The combination of strong detection metrics, satisfactory F1-score, and consistent qualitative detection results supports the selection of YOLO11n as the final model for the proposed system.

CHAPTER 8
CHALLANGES AND LIMITATION

8.0 Challenges & Limitation

The proposed system faced several challenges during dataset preparation, model training, and testing. The original dataset contained 1,331 images, which was expanded to 3,671 images using data augmentation. Although augmentation improved data diversity, the dataset remains relatively limited for representing all possible real-world traffic conditions.

Another challenge was the variation in lighting, vehicle size, viewing angle, background, and traffic density across urban scenes. Such variations can lead to missed detections or false positives. The confusion matrix also indicates that a small portion of auto-rickshaw instances were associated with the background class.

The models were trained using a CPU-based environment, which limited training speed and extensive experimentation. In addition, the current study primarily evaluates auto-rickshaw detection; the complete real-time traffic congestion alert mechanism and live CCTV deployment require further development and validation.

CHAPTER 9
CONCLUSION AND FUTURE WORK

9.0 Conclusion

This research developed a lightweight YOLOv11n-based auto-rickshaw detection framework for urban traffic surveillance in Dhaka City. A dataset of 3,671 augmented images was prepared, annotated into two classes, and divided into training, validation, and testing sets using a 70:15:15 ratio. YOLO11n, YOLO12n, and YOLO26n were trained and comparatively evaluated using Precision, Recall, mAP@0.5, mAP@0.5:0.95, and F1-score.

Among the evaluated models, YOLO11n achieved the best overall performance, obtaining 83.93% mAP@0.5, 47.64% mAP@0.5:0.95, 84.19% Precision, 73.82% Recall, and 78.66% F1-score on the test dataset.

The results demonstrate that YOLO11n can effectively detect auto-rickshaws in diverse urban traffic scenes while maintaining a lightweight architecture. Therefore, YOLO11n was selected as the final model and provides a foundation for future traffic-density estimation, congestion monitoring, real-time alerts, and live CCTV-based deployment.

9.1. Future work

Although YOLO11n achieved the best overall performance among the evaluated YOLO11n, YOLO12n, and YOLO26n models on the 3,671-image auto-rickshaw and non-auto-rickshaw dataset, several opportunities remain for improving the robustness and practical applicability of the proposed detection system.

1. Expand and diversify the dataset: Future work will focus on collecting more images from different locations, weather conditions, lighting environments, traffic densities, and camera viewpoints. Increasing the diversity of the dataset can improve the model's ability to generalize to real-world scenarios.
2. Address challenging detection conditions: Auto-rickshaws may be partially occluded, appear at different scales, or be captured under poor illumination. Future studies can investigate specialized augmentation techniques and preprocessing methods to improve detection under these challenging conditions.

3. Evaluate model efficiency for real-time deployment: Future research can investigate the deployment of YOLO11n on resource-constrained and edge devices such as Raspberry Pi, NVIDIA Jetson, or other embedded platforms. In addition to accuracy, inference speed, memory consumption, power consumption, and latency should be evaluated.
4. Improve class imbalance handling: If the number of auto-rickshaw and non-auto-rickshaw samples is uneven, future work can investigate class-balancing techniques, targeted data augmentation, and alternative sampling strategies to improve performance for underrepresented classes.
5. Explore object tracking: The current work focuses on object detection. A future extension could integrate YOLO11n with object-tracking algorithms to continuously track auto-rickshaws across video frames. This could support applications such as traffic monitoring, counting, and movement analysis.
6. Test on independent real-world datasets: To verify the generalization capability of the model, future experiments should evaluate the trained model on external datasets collected from different locations and camera systems. This would provide stronger evidence of its robustness beyond the current dataset.
7. Extend the detection categories: Future work could distinguish between different types of auto-rickshaws or related three-wheeled vehicles. Additional categories could enable more detailed traffic analysis and intelligent transportation applications.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[2] R. M. Alamgir *et al.*, "Performance Analysis of YOLO-Based Architectures for Vehicle Detection from Traffic Images in Bangladesh," *2022 25th International Conference on Computer and Information Technology (ICCIT)*, pp. 982–987, 2022.

[3] S. Tabassum, M. S. Ullah, N. H. Al-Nur, and S. Shatabda, "Poribohon-BD: Bangladeshi Local Vehicle Image Dataset with Annotation for Classification," *Data in Brief*, vol. 33, p. 106465, 2020.

[4] B. Saha *et al.*, "Bangladeshi Native Vehicle Detection in Wild," *arXiv preprint arXiv:2405.12150*, 2024.

[5] S. A. Fahim, "Finetuning YOLOv9 for Vehicle Detection: Deep Learning for Intelligent Transportation Systems in Dhaka, Bangladesh," *arXiv preprint arXiv:2410.08230*, 2024.

[6] H. M. Hossain *et al.*, "Evaluating YOLO Architectures: Implications for Real-Time Vehicle Detection in Urban Environments of Bangladesh," *arXiv preprint arXiv:2509.05652*, 2025.

[7] R. M. Nabil, T. Kundu, and U. Karmoker, "Real-Time Three-Wheeler Traffic Detection on Expressways Using YOLO Model," *7th International Conference on Civil Engineering for Sustainable Development (ICCESD)*, 2024.

[8] S. D. Sukanto *et al.*, "Detecting Unauthorized Vehicles using Deep Learning for Smart Cities: A Case Study on Bangladesh," *arXiv preprint arXiv:2510.26154*, 2025.

[9] M. Ali *et al.*, "Real-Time Object Detection for Unmanned Vehicles in Bangladesh: Dataset, Implementation and Evaluation," *The Journal of Engineering*, 2024.

[10] M. H. Udoy, A. Rahman, and M. A. S. Zamee, "Real-Time Traffic Vehicle Detection in Bangladesh Using YOLO," B.Sc. thesis, Islamic University of Technology, Bangladesh, 2021.

[11] S. Majumder *et al.*, "Real-Time Wrong-Way Driving Detection System," *IET Image Processing*, 2026.

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

[13] G. Jocher, J. Qiu, M. Liu, S. Lyu, F. C. Akyon, and M. E. Kalfaoglu, "Ultralytics YOLO26: Unified Real-Time End-to-End Vision Models," *arXiv preprint arXiv:2606.03748*, 2026.