APPLICATION OF CHAOS THEORY TO EVALUATE PEDESTRIAN BEHAVIOR USING DEEP LEARNING BASED VIDEO ANALYTICS IN DIFFERENT DIURNAL VARIATIONS

By

MD. MUHTASHIM SHAHRIER

BACHELOR OF SCIENCE IN CIVIL ENGINEERING

DEPARTMENT OF CIVIL ENGINEERING

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DHAKA-1000, BANGLADESH

MARCH 2025

APPLICATION OF CHAOS THEORY TO EVALUATE PEDESTRIAN BEHAVIOR USING DEEP LEARNING BASED VIDEO ANALYTICS IN DIFFERENT DIURNAL VARIATIONS

By

MD. MUHTASHIM SHAHRIER

Student ID: 1904067

A thesis submitted to the

Department of Civil Engineering

in partial fulfillment for the degree of

BACHELOR OF SCIENCE IN CIVIL ENGINEERING

DEPARTMENT OF CIVIL ENGINEERING
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DHAKA-1000, BANGLADESH

March 2025

BOARD OF EXAMINERS

The thesis/project titled "Application of Chaos Theory to Evaluate Pedestrian Behavior using Deep Learning based Video Analytics in Different Diurnal Variations" submitted by Md. Muhtashim Shahrier, Student No.: 1904067, has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Bachelor of Science in Civil Engineering on March 17, 2025.

DECLARATION

I hereby declare that this thesis/project is my own work and to the best of my knowledge it contains no materials previously published or written by another person or have been accepted for the award of any other degree or diploma at Bangladesh University of Engineering and Technology (BUET) or any other educational institution, except where due acknowledgement is made. I also declare that the intellectual content of this thesis is the product of my own work, and any contribution made to the research by others, with whom I have worked at BUET or elsewhere, is explicitly acknowledged.

Md. Muhtashim Shahrier

Dedicated

to

“My parents and my uncle for their constant support and unwavering encouragement”

ACKNOWLEDGEMENTS

I am truly grateful to Almighty Allah for granting me the strength, patience, and determination to complete this work. His blessings have been my constant source of guidance and support.

I would like to express my heartfelt gratitude to my supervisor, Prof. Dr. Md. Hadiuzzaman, for his unwavering support, invaluable advice, and encouragement throughout this journey. His expertise and guidance have played a crucial role in shaping this research, and I sincerely appreciate his mentorship.

I am also deeply thankful to Dr. Nazmul Haque, Lecturer, ARI, BUET, for his continuous guidance and for always being available to help me navigate any challenges I faced. His support has been instrumental in the success of this research, and I truly appreciate his contributions.

A special thanks to Muttahirul Islam for his valuable insights, encouragement, and assistance throughout my research. His willingness to share knowledge and provide guidance has been a great source of motivation, and I sincerely appreciate his support in making this work possible.

I am profoundly grateful to all the individuals mentioned above and to everyone else who has supported me in any way. Their help and encouragement have been truly invaluable, and I will always appreciate their contributions.

ABSTRACT

The increasing complexity of urban transportation systems demands innovative approaches to traffic monitoring and pedestrian behavior analysis. This study applies chaos theory to evaluate pedestrian movement using deep learning-based video analytics, focusing on different diurnal variations. A custom-trained YOLOv8 object detection model combined with DeepSORT tracking is employed to extract pedestrian trajectories from nighttime traffic footage. The custom-trained model achieved a mAP of 0.8 after 1500 epochs with a learning rate of 0.01. The extracted trajectories are analyzed using Lyapunov Exponents and Approximate Entropy, two mathematical indicators that quantify movement stability and unpredictability. The findings reveal that nighttime pedestrian movement exhibits significantly higher unpredictability compared to daytime due to poor visibility, inconsistent pedestrian pathways, and increased pedestrian-vehicle interactions. The mean Lyapunov exponent at night (0.3) indicates greater trajectory divergence than in the daytime (0.1), highlighting chaotic pedestrian behavior. Similarly, Approximate Entropy values show increased movement irregularity at night, reflecting frequent hesitation, abrupt stops, and unpredictable directional changes. These variations suggest that nighttime pedestrian movement is more erratic, posing increased risks in mixed-traffic environments.

The study highlights the fundamental differences in pedestrian behavior between daytime and nighttime, emphasizing how external conditions influence movement stability. By leveraging chaos-based analytical techniques, this research offers a deeper understanding of the irregularities and non-linear patterns inherent in pedestrian dynamics. The integration of deep learning and chaos theory allows for a more precise quantification of pedestrian unpredictability, offering valuable insights into urban mobility trends. These findings contribute to the broader discourse on pedestrian movement analysis, presenting a data-driven approach to evaluating traffic complexity and behavioral adaptation in varied environmental conditions.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivations

The rapid expansion of urban centers presents a critical challenge for modern transportation systems. Projections suggest that by 2050, nearly 70% of the global population will reside in urban areas (Bretzke, 2013). This accelerating urbanization has led to severe transportation issues in developing countries such as Indonesia, China, Nigeria, and Bangladesh (Ahmed & Islam, 2014; Alama, 2021; Rukmana, 2018; Shen, 1997). Consequently, the necessity for efficient traffic management systems has never been more pressing. Such systems play a vital role in alleviating congestion, enhancing road safety, and optimizing overall urban mobility (de Souza et al., 2017). However, the effectiveness of these systems is largely dependent on the accuracy and availability of extensive traffic-related data.

The manual methods of collecting traffic related data have been used for a long time. But the process is very time consuming and labor intensive. There has been a trend to automate this process on the last couple of decades by creating new technologies and techniques. With the help of Intelligent Transportation System (ITS), computer-based methods are being prioritized for collecting these traffic data. Also, development of deep learning and computer vision has created a new path for traffic detection. Several deep learning architectures are being used to develop new technologies for traffic data collection and object detection. Deep learning-based techniques for detection are more precise than traditional methods.

One of the most significant challenges in traffic data collection is ensuring the accuracy of the data which has been collected. Traditional methods such as manual observation, surveys, and sensor-based systems often suffer from issues like human error, limited coverage, and high operational costs. Real-time data collection and analysis of these data is crucial for managing traffic effectively which is not possible with these methods. On the other hand, deep learning-based systems can provide more reliable and real-time solutions by automatically detecting vehicles, pedestrians, and other traffic elements with a higher degree of accuracy. This is particularly beneficial in urban areas where traffic patterns are complex and can change rapidly.

Combining deep learning with computer vision makes it possible to develop advanced systems for traffic detection and monitoring. These systems can identify a wide range of objects and characteristics, such as vehicle types, speeds, and congestion levels, allowing for better decision-making in traffic management. Furthermore, as the field of deep learning evolves, more advanced models are emerging that can handle noisy, incomplete, or ambiguous data more effectively. The ability to process real-time data and adapt to changing traffic conditions allows for more proactive and responsive traffic management, leading to smoother traffic flow and fewer accidents.

The ability to process large-scale, real-time data also opens new possibilities for enhancing road safety. Deep learning models can detect and classify potential risks on the road, such as accidents, potholes, or erratic driving behavior, that might otherwise go unnoticed. Once successfully detected, this information can be sent instantly to traffic control centers. This enables faster response times and potentially preventing accidents before they occur. Additionally, these systems can monitor driving behaviors, such as speed violations or dangerous lane changes, and trigger automated alerts or enforcement measures, contributing to safer driving environments.

One of the most notable advancements in traffic monitoring is the application of object detection models such as YOLO (You Only Look Once). YOLO is a state-of-the-art deep learning-based computer vision algorithm, and it has revolutionized the way traffic data is collected. YOLOv8 is the version which is being used widely because of its simplicity. By detecting and classifying objects in real-time from video feeds, YOLOv8 is capable of accurately identifying vehicles, pedestrians, and traffic signs with minimal processing time. This allows for near-instantaneous data capture and analysis, which is essential for live traffic management.

Despite their superiority, deep learning techniques face challenges such as the need for extensive labeled data, potential annotation errors, and high computational demands. Computer vision further complicates annotation, while model training requires powerful hardware. However, increasing dataset availability and GPU advancements have eased implementation. Practical challenges persist, including variability in data quality due to lighting, weather, and camera differences. Researchers are improving robustness using synthetic data and domain adaptation to enhance model performance across diverse conditions.

Traffic congestion in Bangladesh is worsening due to urbanization and increasing vehicles. AI-based solutions like YOLOv8 can enhance traffic monitoring, aiding real-time decision-making for better control and infrastructure. Dhaka's diverse traffic mix, including rickshaws and motorcycles, complicates detection due to size variations and inconsistent manufacturing. Poor lighting and infrastructure further hinder accuracy. Adaptive deep learning models are essential to address these challenges and improve traffic management, safety, and urban sustainability.

While there have been some studies regarding traffic detection in Bangladesh, there is no study where traffic detection model is being trained dedicatedly for nighttime traffic. This study aims to address this research gap.

While much attention has been given to vehicle detection, pedestrian behavior remains an equally crucial yet underexplored aspect of urban traffic systems. Unlike vehicles, pedestrian movement is inherently chaotic, characterized by unpredictable variations in speed, direction, and trajectory. This unpredictability is particularly present in cities like Dhaka, where heterogeneous traffic conditions result in frequent pedestrian-vehicle interactions. This often happens due to the absence of designated crossings or proper traffic controls.

Pedestrian chaos manifests in various ways, including jaywalking, sudden trajectory shifts, and erratic crossing behavior, particularly in high-density urban areas. These behaviors pose significant challenges for traffic management. Traditional management often struggle to account for the non-linearity of human movement. Moreover, pedestrian behavior is influenced by various external factors, such as visibility, road design, traffic density, and environmental conditions. The need for robust pedestrian tracking and behavioral analysis is therefore critical for improving urban safety and efficiency.

Pedestrian chaos is further influenced by socio-economic factors. In many developing nations, the lack of well-defined pedestrian infrastructure forces people to adapt to informal crossing patterns, often resulting in hazardous interactions with vehicles. Cultural factors also play a role in how pedestrians navigate traffic, as jaywalking may be more socially accepted in certain regions. Understanding these behavioral distinctions is crucial for designing pedestrian-friendly traffic management solutions.

The challenge of monitoring pedestrian chaos becomes even more pronounced in nighttime conditions. Reduced visibility, inadequate lighting, and glare from vehicle headlights significantly impact pedestrian detection accuracy. Traditional traffic surveillance systems often fail to account for these complexities, leading to increased accident risks during nighttime hours. Deep learning-based pedestrian tracking systems, such as those integrating YOLO with multi-object tracking algorithms like DeepSORT, offer a promising solution. These models can track pedestrian movements across multiple frames, preserving their identities even in challenging conditions such as occlusions or low-light environments.

Furthermore, pedestrian behavior changes significantly in low-light conditions. Pedestrians take longer to assess traffic conditions at night, leading to increased hesitation before crossing roadways. This hesitation, combined with sudden bursts of movement, results in erratic trajectories that are difficult to predict. Pedestrians also tend to walk closer to the curb or remain in shaded areas, making them less visible to drivers and traditional surveillance systems. These behavioral adaptations further complicate the task of pedestrian detection and chaos analysis.

The application of chaos theory in pedestrian behavior analysis provides an additional dimension to understanding movement unpredictability. By analyzing pedestrian chaos mathematically, researchers can identify high-risk areas, predict collision probabilities, and develop intervention strategies to enhance pedestrian safety.

This study aims to bridge these gaps by employing deep learning-based object detection for both vehicles and pedestrians while simultaneously analyzing pedestrian chaos using chaos theory-based methodologies. By doing so, it provides a more holistic approach to urban traffic monitoring, particularly under nighttime conditions where traditional methods often fail. Through this research, we seek to contribute to the advancement of intelligent transportation systems that can accommodate the complexities of real-world urban environments.

1.2 Objective of the Study

The main objectives of this study are to

1. Develop a custom-trained YOLOv8 model using a custom dataset for nighttime road users' detection in Bangladesh and extract pedestrian trajectories using the DeepSORT algorithm.
2. Analyze nighttime and daytime pedestrian trajectories using chaos theory, focusing on Lyapunov Exponent and Approximate Entropy.

1.3 Scope of the Study

The scope of this study focuses on developing a deep learning-based approach for nighttime road user detection and pedestrian trajectory analysis in Bangladesh's urban traffic environment. The research involves training a custom YOLOv8 model using a dataset specifically collected under low-light conditions in nighttime. The trained model is specifically for nighttime detection. The model will detect 16 classes which are selected based on BRTA. The classes are Ambulance, Auto-Rickshaw, Bicycle, Bus, C.N.G., Covered-Van, Cycle-Rickshaw, Human-Hauler, Microbus, Motor-Cycle, Pedestrian, Pick-Up, Private-Passenger-Car, Non-Motorized-Van, Special-Purpose-Vehicles, Truck. Additionally, the study implements the DeepSORT algorithm to extract pedestrian trajectories. A key aspect of the study is the comparative analysis of pedestrian movements during nighttime and daytime, utilizing chaos theory-based metrics such as the Lyapunov Exponent and Approximate Entropy to assess movement stability and unpredictability.

This research aims to bridge the gap in nighttime traffic monitoring by introducing a robust detection model tailored to Bangladesh's mixed-traffic conditions. By applying chaos theory to trajectory analysis, the study provides insights into pedestrian behavior, identifying areas with increased instability. The findings will contribute to improving traffic safety measures, such as better lighting and designated pedestrian crossings. However, the study is limited to a specific urban setting, and future research may be required to validate the model's generalizability across diverse environments.

1.4 Organization of the Thesis

This thesis is structured into seven chapters, each covering a specific aspect of the research process, from background and literature review to methodology, analysis, and conclusions.

Chapter 1 provides an introduction to the study, outlining the background, motivations, and significance of the research. The chapter also presents the research objectives, scope, and expected contributions.

Chapter 2 presents a comprehensive literature review, defining key terms and reviewing previous research. It highlights research gaps and establishes the foundation for the study.

Chapter 3 describes the research methodology, detailing the data collection methods, computational techniques, and analytical frameworks employed.

Chapter 4 focuses on data collection methods and preparation of dataset used in this study. It covers the selection of study sites, the structure of the video data, and preprocessing techniques.

Chapter 5 presents the model building process. It provides a detailed analysis of the model's performance, accuracy, and reliability in detecting and tracking traffic objects.

Chapter 6 discusses the interpretation of the results. It also includes the chaos theory analysis of pedestrian trajectories.

Chapter 7 concludes the thesis by summarizing the key findings of the study. It discusses the practical significance of the research and provides recommendations for future work.

The thesis concludes with an appendix containing pseudocodes used in the study.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Traffic detection has evolved significantly over the years, transitioning from manual observation techniques to advanced deep learning-based models. The integration of computer vision and artificial intelligence, particularly through YOLO models, has enhanced real-time traffic detection capabilities. However, challenges persist, especially in nighttime traffic detection, where factors such as glare, low illumination, and occlusions affect accuracy. In regions like Bangladesh, unique traffic conditions further complicate the application of standard models, necessitating customized approaches. This chapter provides an in-depth review of existing traffic detection techniques, advancements in YOLO-based models, and their applications in various scenarios, including nighttime conditions and Bangladesh's heterogeneous traffic landscape. Additionally, image enhancement techniques and tracking methodologies are explored to improve detection performance and pedestrian trajectory analysis. Understanding pedestrian chaos, particularly in mixed-traffic conditions, is crucial for enhancing traffic management systems, as unpredictable pedestrian behavior often contributes to congestion and accident risks, especially at night.

2.2 Traffic Detection Evolution

The evolution of traffic detection is an indication of the continuous technological advancements which improved road safety, traffic management, and urban planning. Different techniques were invented and implemented from time to time for traffic detection. The journey began in the early 20th century, with elementary methods such as manual counting, where traffic policemen or researchers would physically count vehicles passing through a specific point. These early efforts were primarily for traffic studies, and while effective, they were labor-intensive and prone to human error. In the late 1920s, there were some studies where statistical analysis of traffic data was done by manual counting by counting traffic in 5 minutes interval (Albright, 1991). In the mid of the 20th century, traffic detection was based on technologies like pneumatic road tubes and inductive loop detectors. In the late 20th century, several noninterfering technologies were invented. In the 1990s, researchers started to use video-based traffic detection techniques. With the rise of artificial intelligence in the 21st century, traffic detection systems incorporate machine learning and deep learning techniques. Techniques like object detection and classification, which are based on deep learning, are heavily employed in Intelligence Transportation System and vehicle detection (Berwo et al., 2023). The gradual development of these techniques has advanced the entire traffic detection system.

2.3 YOLO and YOLOv8

Computer vision is a field of artificial intelligence which focuses on enabling machines to understand and interpret visual information (Diwan et al., 2023). Computer vision relies on Convolutional Neural Networks (CNNs) for tasks such as object detection, image classification and object segmentation(Haupt & Nowak, 2006; Hussain, 2023). YOLO (You Only Look Once), invented by Joseph Redmon in 2016, represents a significant advancement in this field by reconceptualizing object detection as a regression problem, predicting bounding boxes and class probabilities in one unified evaluation. As described by Redmon et al. (2016), "a single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation". YOLO divides the input image into grid cells, with each cell predicting whether an object exists, its class, and the bounding box. The model progressively improves its accuracy by making multiple predictions per frame and selecting the most accurate one, discarding the others(Gothane, 2021). YOLO delivers results more quickly than other two-stage object identification methods due to its approach to the problem(Atik et al., 2022).

The training network used in YOLO was based on GoogleNet whereas a new classification model called DarkNet-19 was used as a primary network in YOLOv2(Jiang et al., 2022). YOLOv2 works by dividing an image into grid cells, performing classification and localization within each cell, and then applying anchor boxes to focus on objects like preset shapes. After thresholding to eliminate low-confidence detections, the final object locations are determined using Non-Max Suppression with Intersection over Union (IoU)(Gupta, 2020; Wen et al., 2021).

In "YOLOv3: An Incremental Improvement," Redmon and Farhadi presented updates to the original YOLO architecture. The paper introduces design modifications that enhance accuracy while maintaining the model's speed. YOLOv3, although slightly larger, achieves a mAP of 28.2 at 320 × 320 resolution, running in 22 ms, making it as accurate as SSD but three times faster. The model also performs well on the AP50 metric, achieving 57.9 AP50 in 51 ms, compared to RetinaNet's 57.5 AP50 in 198 ms, demonstrating a significant speed advantage(Redmon & Farhadi, 2018). YOLOv3 used DarkNet-51 as architecture backbone(Terven et al., 2023).

The YOLOv4 model, developed by Alexey Bochkovskiy and colleagues, advances the YOLO series by integrating a range of improvements to enhance both speed and accuracy. This iteration incorporates various techniques such as CSPNet, PANet, and SAM, achieving significant gains in performance while maintaining real-time processing capabilities(Bochkovskiy et al., 2020).

Glenn Jocher, founder and CEO of Ultralytics developed the YOLOv5 model using Pytorch instead of Darknet framework. The architecture of YOLOv5 introduced a strided convolution layer with a large window size to decrease memory usage and computational costs(Hussain, 2024; Terven et al., 2023). The YOLOv5 has five variants (YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), and YOLOv5x (extra-large)) to incorporate different hardware and software requirements(Hussain, 2024).

YOLOv6, introduced by Li et al. (2022), enhances YOLO's efficiency for industrial applications by integrating advanced design and optimization techniques. It achieves notable performance, including 43.5% AP at 495 FPS with YOLOv6-S and a quantized version reaching 43.3% AP at 869 FPS. YOLOv7, developed by Wang et al. (2023), introduces a trainable bag-of-freebies approach that combines flexible training tools with a new architecture and compound scaling method. It sets a new benchmark for real-time object detection, achieving a top accuracy of 56.8% AP at 30 FPS or higher on a GPU V100.

YOLOv8, developed by Ultralytics, marks a significant advancement in the YOLO series, building on the strengths of its predecessors while introducing novel architectural improvements to enhance performance. YOLOv8 stands out for its exceptional speed and accuracy, all while maintaining a streamlined design that is adaptable to various applications and hardware platforms(Sohan et al., 2024). There are five YOLOv8 variants: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large)(Hussain, 2024). The architectural backbone used for YOLOv8 is C2f module which is an adapted version of the CSPLayer (Hussain, 2024; Sohan et al., 2024; Terven et al., 2023; T. Wu & Dong, 2023). YOLOv8 has shown improvement in performance when dealing with smaller object as YOLOv8 utilizes CIoU (Zheng et al., 2020) and DFL (X. Li et al., 2020) loss functions for bounding box predictions and binary cross-entropy for classification (Terven et al., 2023; T. Wu & Dong, 2023).

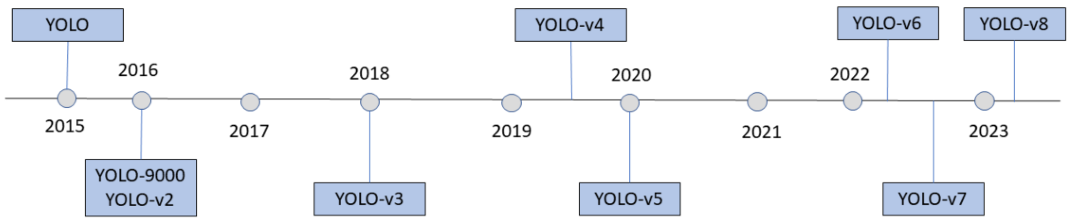


Figure 2.1: Timeline of YOLO advancements (Sohan et al., 2024)

2.4 Traffic detection with Yolo

YOLO models have been widely adopted for traffic and vehicle detection due to their real-time capabilities and high accuracy. There are several studies where Yolo model is used for detection related to transportation. Sang et al. (2018) proposed YOLOv2_Vehicle, an improved vehicle detection model based on YOLOv2, incorporating k-means++ clustering for anchor boxes, normalization of bounding box dimensions, and multi-layer feature fusion, achieving a 94.78% mAP and effective generalization across different datasets. The model's limitations include a relatively small vehicle type and data volume, with future work aimed at expanding the dataset and further enhancing detection accuracy and speed. Chen & Lin (2019) introduced YOLO v3-live, an optimized version of YOLOv3-tiny for real-time vehicle detection on embedded devices, achieving 87.79% mAP before quantization and 69.79% mAP with a detection speed of 28 FPS after quantization. Xu et al. (2019) improved YOLOv3 to enhance vehicle detection in aerial images by increasing network depth and utilizing top-level feature maps for better detail and accuracy. The algorithm outperforms existing state-of-the-art methods for aerial vehicle detection. Rodríguez-Rangel et al. (2022) developed a system to estimate vehicle speeds in real-time using monocular cameras, extracting features with YOLOv3 and Kalman filters, and found that a Linear Regression Model (LRM) provided the best accuracy with a Mean Absolute Error (MAE) of 1.694 km/h and 0.956 km/h on different lanes. However, the study faced challenges in data sampling and processing time. Y. Zhang et al. (2022) proposed an enhanced YOLOv5-based method for vehicle detection that addresses false detections caused by occlusions by introducing the Flip-Mosaic algorithm, which improves the network's sensitivity to small targets. The model, trained on a diverse dataset, achieved higher accuracy in various traffic scenarios, particularly in high-speed environments. However, the study noted the limitations due to the scarcity of highway surveillance data and the need to improve robustness under adverse weather conditions. Padilla Carrasco et al. (2023) introduced a modified YOLOv5 model designed for tiny vehicle detection in parking areas viewed from an overhead perspective. The model incorporates a multi-scale mechanism and attention modules, improving detection precision to 96.34%, a significant increase from the 63.87% achieved by the baseline YOLOv5. This modification reduces the number of trainable parameters and maintains competitive recall and mAP values, although it slightly decreases detection speed compared to smaller YOLOv5 profiles.

There are several studies in the recent times where YOLOv8 is used for traffic detection. Soylu & Soylu (2023) developed a traffic sign recognition system using YOLOv8, achieving a high mAP-50 score of 0.995 and mAP-50-95 score of 0.802 with the YOLOv8-nano model. This version was selected for its efficiency in accuracy, speed, and model size. Telaumbanua et al. (2023) applied the YOLOv8 model for vehicle detection and identification across 9 classes, achieving 77% accuracy during training and 96% during testing. Although effective, the study highlighted challenges with occlusion and suggested future improvements through expanded datasets and enhanced data augmentation. Al Mudawi et al. (2023) proposed a method for vehicle detection and classification in aerial image sequences using a five-stage model. After preprocessing images for noise reduction, the YOLOv8 algorithm detected vehicles, followed by feature extraction using SIFT, KAZE, and ORB. The Deep Belief Network (DBN) classifier was then used for classification, achieving 95.6% accuracy on the VEDAI dataset and 94.6% on VAID. M. Wang & Ren (2023)proposed SSB-YOLO, an improved YOLOv8 model for vehicle detection that integrates Shuffle Attention and spatial-channel reconstruction to enhance performance. With Wise-IoU optimization for better bounding box regression, the model achieved a 1.6% mAP@50 increase over YOLOv8n, improving accuracy, robustness, and reducing computational costs. H. Wang et al. (2024) introduced YOLOv8-QSD, an anchor-free network for self-driving cars, enhancing YOLOv8 with structural reparameterization and BiFPN for improved small object detection. It achieves 64.5% accuracy and 7.1 GFLOPs on the SODA-A dataset, offering superior speed and precision compared to YOLOv8. Fei et al. (2024) proposed a lightweight vehicle detection algorithm based on YOLOv8 by integrating FasterNet, SimAM attention module, and a small target detection head, achieving improved accuracy and speed in complex traffic scenarios. Their method enhanced the mAP and FPS by 3.06 percentage points and 3.36%, respectively, effectively addressing the challenge of detecting small vehicles. X. Chen (2024) enhanced the YOLOv8 network by incorporating a CBAM attention mechanism and a soft-NMS mechanism to improve detection of small, occluded, and multiple objects in complex urban traffic scenes. Their approach increased model accuracy by 2.6% and reduced GFLOPs consumption by 0.8. B. Wang et al. (2024) improved the YOLOv8n model by integrating the BiFormer attention mechanism, a small-scale target detection header, a weighted bidirectional feature pyramid structure, and WIoUv3 as a loss function, leading to significant increases in mAP and mAP50:95 by 4.7 and 6.2 percentage points, respectively. Bakirci (2024) explored the capabilities of YOLOv8 for vehicle detection in Intelligent Transportation Systems, highlighting its decoupled head structure and C2f module. The study demonstrated YOLOv8's 18% improvement in precision and superior processing speed compared to YOLOv5, revealing its potential and limitations in real-time applications. The YOLOv8 model has progressively changed the whole traffic detection scenario due to its accuracy and simplicity.

2.5 Image Enhancement and Glare Removal

Image enhancement is a crucial part of deep learning-based vehicle detection methods. Glaring from headlight in the nighttime is also a big problem in traffic detection. There are several studies where various image enhancement and glare removal techniques are developed. Singh & Banga (2013) developed a nighttime image enhancement method for surveillance cameras by applying the moving target extraction technology and illumination estimation theory with image fusion. Their approach recovers scene information and highlights vehicle details, achieving a high PSNR (Peak Signal to-noise ratio) of 65.9 dB and a low MSE (Mean Squared Error) of 0.128, demonstrating its effectiveness over earlier techniques. M. Chen et al. (2016) proposed a method to detect leading vehicles at night using multisensor data fusion by integrating millimeter wave radar and image enhancement algorithms to improve visibility and reduce rearend accidents. Their approach effectively enhances nighttime driving safety by accurately detecting and positioning preceding vehicles, though they acknowledge the need for further optimization to enable real-time processing and broader application. Kuang et al. (2016) introduced a vehicle detection method for nighttime images by combining a region-of-interest extraction approach with an enhanced image processing technique based on improved multiscale retinex. Their method achieved a 93.34% detection rate, outperforming other models by effectively detecting vehicles even when blurred, occluded, or presented in various challenging conditions. Kuang et al. (2017) developed a novel nighttime vehicle detection system combining a bioinspired image enhancement approach, drawing inspiration from retinal mechanism of eye, with a weighted feature fusion technique. Their method achieved a 95.95% detection rate at 0.0575 false positives per image in detecting vehicles of various types, sizes, and under challenging conditions like occlusions and blurring. Ye et al. (2020) proposed a decompose-refine network architecture for glare removal in single images, consisting of a glare detection subnetwork and a glare removal subnetwork, effectively enhancing the perceptual quality of de-glared images. K. Xu et al. (2020) introduced an improved multitask cascaded CNN (IMC-CNN) for detecting small and fuzzy vehicles, employing contrast limited adaptive histogram equalization (CLAHE) and multi-scale Retinex (MSR) for image enhancement. Shao et al. (2021) proposed a cascade detection network called FteGanOd that enhances nighttime vehicle detection by integrating a feature translate-enhancement (FTE) module with an object detection (OD) module. This method effectively improves vehicle detection accuracy by enhancing features and suppressing background interference in complex lighting conditions. G. Li et al. (2021) developed a light enhancement network (LE-net) designed to improve low-light images for safer connected autonomous vehicle (CAV) driving, particularly in extremely low-light conditions. The LE-net was trained on synthetic low-light images generated from daytime images and demonstrated superior performance in enhancing image details while minimizing noise, outperforming other methods in both qualitative and quantitative assessments. Zhao et al. (2021) introduced a vehicle detection method designed for low illumination conditions by employing an image enhancement algorithm to improve contrast and detection accuracy. The approach first enhances image contrast using an adaptive contrast stretching algorithm, followed by noise reduction through bilateral filtering. The detection process utilizes a system based on Haar features and an AdaBoost classifier. Experimental results demonstrate that the method significantly enhances image contrast and vehicle detection accuracy, achieving an 87.04% detection rate under low illumination. Mandal et al. (2021) proposed a night-vision system that dims the high-beam headlights of oncoming vehicles while enhancing the road view, using region segmentation, local enhancement techniques, and adaptive Gaussian filtering. The system shows superior performance in real-time tests compared to existing methods, offering better visibility for both manual and autonomous vehicles. Cheng et al. (2023) presented an enhanced GAN-based algorithm for improving highway traffic images under complex weather conditions, integrating an attention mechanism and multiscale feature fusion to reduce noise and enhance detail. Yoon & Cho (2023) proposed a technique to improve object detection in low-light environments by inverting low-light images and applying a haze removal algorithm to enhance contrast, followed by using an adaptive 2D Wiener filter to reduce noise. Their method significantly enhances image perception, with a 12.73% reduction in image quality evaluator scores and an 18.63% increase in vehicle detection average precision compared to existing techniques. Khrisne et al. (2023) proposed an enhanced image processing approach for low-light traffic surveillance using ZeroDCE combined with a glare reduction stage, significantly improving image quality with PSNR rising from 28.91 to 31.07 and SSIM from 0.55 to 0.88. Du et al. (2024) introduced a dual-branch network for nighttime vehicle detection that processes RGB images to handle both glare reduction and glare acknowledgment. Their method, which improves average detection accuracy by 4.7% over the YOLOv5 baseline on the VD-NUS-G dataset, effectively manages glare without the need for additional infrared devices.

2.6 Traffic Detection at Nighttime with YOLO

Traffic detection at nighttime presents a unique challenge due to poor visibility and varying lighting conditions. There are several studies which explored the aspect of traffic detection at nighttime using YOLO models. Ho et al. (2020) investigated vehicle detection at night using YOLOv4 combined with image preprocessing techniques, finding that training on the original night data achieved the highest efficiency at 64.51% mAP. Miao et al. (2020) developed a nighttime vehicle detection method using MSR-enhanced YOLOv3, which outperformed Faster R-CNN and SSD with an average precision of 93.66%. Huang et al. (2021) introduced M-YOLO, a lightweight deep neural network model based on YOLOv3, which utilizes MobileNet v2 for feature extraction and the EIoU loss function for optimization, achieving an average precision of 94.96% and processing ten frames per second in constrained environments. This model is 18.45% more accurate and three times faster than YOLOv3. Jhong et al. (2021) proposed a nighttime object detection system for IoV (Internet of Vehicles) applications using a lightweight deep learning model. The pedestrian detection and vehicle detection algorithm is based on YOLO architecture. Y. Wu et al. (2023) developed an improved nighttime detection algorithm based on YOLOv7-tiny, incorporating Ghostnet V2, FReLU activation, Wise-10 U boundary box loss, SimAM, C3, and ODConv modules to enhance both detection accuracy and speed. The optimized algorithm reduces floating-point operations by 59% and achieves 47 fps with a minor decrease in mAP, demonstrating effective performance improvements. Yin et al. (2023) introduced Light-YOLO, a lightweight vehicle detection model based on YOLO, designed to enhance accuracy and real-time performance in low-light conditions. The model incorporates a polarized self-attention-enhanced feature pyramid network, Swift spatial pyramid pooling, and an anchor mechanism with focal loss, achieving significant improvements in mAP and detection speed while reducing parameters and maintaining real-time efficiency. Zhang et al. (2024) developed a hierarchical contextual information (HCI) framework to enhance nighttime vehicle detection by integrating estimation, segmentation, and detection branches to improve performance over existing models like YOLOX and Faster R-CNN. Guo et al. (2024) introduced KSC-YOLOv5, an enhanced YOLOv5 model with SKAttention, CARAFE, and SIOU to improve nighttime vehicle detection, achieving a 2% increase in mAP to 91.2% and superior accuracy compared to other YOLO versions. However, the model's performance may still be affected by extreme low-light conditions and highly blurred vehicle images. Wei & Yu (2024) enhanced the YOLOv8n algorithm for nighttime vehicle detection by preprocessing images with the Retinex algorithm, introducing MishDyHead, and optimizing the network with FasterEMA_Block and MPDIoU loss, resulting in a 2.5% increase in Average Precision and a 6.7% reduction in parameters.

2.7 Traffic Detection in Bangladesh with YOLO

The traffic condition in Bangladesh is unique as it includes both motorized and non-motorized vehicles in highway at the same time causing heterogeneous traffic flow (Zubaer et al., 2020). Detection of heterogeneous traffic is challenging considering they do not follow lane behavior like homogeneous traffic (Mittal et al., 2018). There are several studies which applied YOLO model for traffic detection in Bangladesh. Several studies optimized and customized the model for Bangladeshi traffic. Rahman et al. (2020) presented a real-time detection system for wrong-way vehicles using the YOLO algorithm for object detection and a centroid tracking algorithm to monitor vehicle movements in traffic videos, demonstrating high accuracy under various environmental and weather conditions. Tabassum et al. (2020) proposed a vehicle detection system using transfer learning with CNN applied to the YOLO framework, achieving a 73% IoU (Intersection over Union) for detecting 15 types of native Bangladeshi vehicles using custom dataset. Datta et al. (2020) implemented a Faster R-CNN-based road object detection system for Bangladesh, achieving an 86.42% accuracy in identifying 19 object classes in various traffic conditions in real-time. Onim et al. (2020) developed a vehicle license plate detection and recognition system using YOLOv4 for Bangladesh, achieving a 90.50% mAP and presented a user-friendly GUI. Shah Junayed et al. (2021) developed a real-time front vehicle detection system using the YOLO model, which improves detection accuracy and surpass performance of other leading detectors on the DhakaAI dataset. The system achieved improvements in mAP (mean average precision) and FPS (frame per second) compared to RetinaNet, SSD, and Faster R-CNN. Shomee & Sams (2021) proposed a multi-step deep learning system for Bangladeshi license plate detection and recognition using YOLOv4 and ESRGAN, achieving a mean average precision of 98.35% and 98.09% with average prediction times of 23 ms and 35 ms. Lima et al. (2022) performed a comparative analysis of road sign detection using various YOLO and R-CNN architectures on the "BD Road Sign 2021 (BDRS 2021)" dataset. In the study, YOLOv3 and Faster R-CNN was found to be the most effective for detecting road signs in Bangladesh. Rafi et al. (2022) conducted a performance analysis of various YOLOv5 models (small, medium and large variant) for recognizing South Asian vehicles, finding that the YOLOv5 large variant outperforms others in accuracy. Bari et al. (2022) evaluated YOLOv3, YOLOv3-tiny, and YOLOv4-tiny models for detecting and counting traffic vehicles in Bangladesh, finding that YOLOv4-tiny outperforms the others in terms of recall, F1-Score, average FPS, and mAP by balancing speed and accuracy for real-time applications. Alamgir et al. (2022) compared YOLOv3, YOLOv5s, and YOLOv5x architectures for vehicle detection in traffic images from Bangladesh, finding YOLOv5x to be the most effective with 7% and 4% higher mAP, and 12% and 8.5% better accuracy than YOLOv3 and YOLOv5s, respectively. Fahim et al. (2023) developed a system for detecting Bangla number plates using the YOLO V8 which achieved a 98.4% mean average precision (mAP). They also addressed challenges such as variations in fonts, sizes, orientations, and lighting conditions. Saha et al. (2024) introduced the Bangladesh Native Vehicle Dataset (BNVD), a comprehensive dataset designed for autonomous vehicle detection in Bangladesh. The dataset includes 17 vehicle classes and 17,326 images with 81,542 fully annotated instances, addressing challenges like geographical diversity, illumination, and adverse weather conditions. BNVD's effectiveness was assessed using YOLO v5, v6, v7, and v8 models, achieving a mean average precision (mAP) of 0.848 at 50% intersection over union (IoU) and 0.643 across a range of IoUs.

2.8 DeepSORT: A Multi-Object Tracking Algorithm

DeepSORT (Deep Simple Online and Realtime Tracking) is an extension of the SORT algorithm, designed to enhance multi-object tracking by integrating deep learning for appearance-based tracking. This integration allows the algorithm to maintain object identities more robustly in occlusions or when objects are in close proximity (Wojke et al., 2017). The original SORT algorithm relied primarily on motion information. It was efficient but struggled with identity switches in crowded scenes. DeepSORT addresses this limitation by incorporating a deep association metric. It utilizes a pre-trained convolutional neural network to extract appearance features of detected objects. These features are then used to match objects across frames. It significantly reduced identity switches and improved the overall tracking performance (Wojke et al., 2017). Several studies have applied DeepSORT in various contexts. For instance, (X. Chen et al., 2022) proposed a pedestrian detection and tracking method combining an improved YOLO model with DeepSORT,. It achieved better handling of small and occluded targets in complex environments. Integrating DeepSORT with YOLOv5 has shown improved performance in tracking multiple objects in real-time scenarios (Razzok et al., 2023).

2.9 Pedestrian Tracking and DeepSORT Application

Pedestrian tracking is an essential component of intelligent transportation systems, urban safety, and autonomous navigation. It involves continuously identifying and following individuals across frames in video footage. Pedestrian tracking presents unique challenges because the unpredictability of human movement, occlusion, and changes in appearance.

Pedestrians exhibit highly dynamic and non-linear movement behaviors unlike vehicles. They frequently change direction, alter speed, and interact with other pedestrians and objects in their environment. This makes pedestrian tracking significantly more complex compared to vehicle tracking. Furthermore, occlusion is a major issue in pedestrian tracking. People often walk in groups, overlap with each other, or become momentarily hidden by obstacles such as vehicles, trees, or buildings.

To address these challenges, modern pedestrian tracking systems combine object detection with advanced tracking algorithms. Typically, a deep learning-based object detection model (such as YOLO, Faster R-CNN, or SSD) first identifies pedestrians in each frame. Then a tracking algorithm like DeepSORT maintains their identities across multiple frames. The integration of deep learning in both detection and tracking stages has significantly improved the accuracy of pedestrian tracking systems. For example, (X. Chen et al., 2022) proposed a method that combines an improved YOLO model with DeepSORT for pedestrian detection and tracking in intelligent vehicle systems. Their approach enhances the detection of small and occluded targets. It leads to more reliable tracking in complex environments. Another study by (Razzok et al., 2023) focused on improving the data association component of the DeepSORT algorithm. By introducing new cost matrices based on metrics such as intersections and distances, they achieved better performance in pedestrian tracking tasks in crowded scenes. (Sheng et al., 2024) introduced a multi-objective pedestrian tracking method that combines YOLOv8 with an improved DeepSORT algorithm. Their approach addresses challenges like occlusion and ID switching, achieving high accuracy and real-time performance in complex traffic scenarios. In the context of vehicle tracking, integrating YOLOv5 with DeepSORT has proven effective. This combination creates an end-to-end pipeline for vehicle detection and tracking, enhancing overall performance and accuracy, which is crucial for real-time traffic management applications (Kumar et al., 2023).

2.10 Stability Checking Indicators

Understanding the stability and predictability of dynamic systems is essential in various scientific and engineering fields. Several mathematical indicators are commonly used to assess the stability and chaotic behavior of such systems:

Lyapunov Exponent: This measures the rate at which nearby trajectories in a system diverge or converge. A positive Lyapunov exponent indicates chaos, where small differences in initial conditions lead to exponentially divergent outcomes, reflecting system sensitivity and unpredictability.

Approximate Entropy: This quantifies the complexity and regularity of fluctuations in a time series. Lower values suggest more predictable and regular patterns, while higher values indicate complexity and randomness. It's particularly useful in analyzing physiological data to detect irregularities and potential disorders.

These indicators provide valuable insights into the behavior of complex systems, aiding in the development of models and strategies for prediction and control.

2.11 Summary

This chapter provides a comprehensive review of traffic detection techniques, with a particular focus on deep learning-based object detection models like YOLO and their applications in nighttime traffic monitoring. The evolution of traffic detection is discussed, highlighting the transition from manual counting methods to advanced computer vision models such as YOLOv8, which offers improved accuracy and real-time detection capabilities. Studies related to traffic detection in Bangladesh are also reviewed, emphasizing the need for customized models to handle the country's unique heterogeneous traffic conditions. Finally, pedestrian tracking methods and chaos-based analysis techniques, such as Lyapunov Exponents and Approximate Entropy, are explored to better understand unpredictable pedestrian movements in mixed-traffic environments.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents the methodological framework used to analyze pedestrian nighttime trajectories, focusing on detecting chaotic behavior in pedestrian movements. The study leverages deep learning-based object detection models and tracking algorithms to extract pedestrian trajectories and analyze their stability using chaos theory.

The methodology is structured into three core areas: object detection, tracking, and chaos analysis. The custom trained model using YOLOv8 object detection model is employed for road users' including pedestrian identification in nighttime conditions. To maintain pedestrian identity across multiple frames, the DeepSORT tracking algorithm is used, integrating both motion and appearance-based features to enhance tracking reliability.

A key innovation in this study is the application of chaos theory to analyze pedestrian movements. Pedestrian behavior, especially in mixed-traffic conditions, often exhibits non-linearity and unpredictability. To quantify these characteristics, the study employs two key indicators of chaotic motion Lyapunov Exponents and Approximate Entropy. These mathematical measures allow for a detailed assessment of pedestrian trajectory stability, identifying locations where pedestrians display erratic or unpredictable behavior.

3.2 Object Detection

Pedestrian detection in nighttime conditions requires a robust object detection framework capable of handling low visibility, glare, and occlusions. To achieve this, a custom-trained YOLOv8 model is developed using a dataset specifically collected for this study, ensuring optimal performance in real-world nighttime traffic scenarios.

3.2.1 Custom Model Training Approach

To accurately detect pedestrian movements at nighttime, a custom-trained YOLOv8 model is developed using a dataset specifically collected and annotated for this study. Unlike pre-trained models, which are trained on general datasets, this approach ensures that the detection model is optimized for the unique conditions of nighttime traffic, including low illumination, glare, and occlusions in Bangladesh. The dataset is created from video recordings of real-world traffic videos, followed by manual annotation to label pedestrian instances and other road users. Using this custom dataset, the YOLOv8 model is fine-tuned through supervised training, ensuring improved accuracy in road users' detection under challenging nighttime conditions. The following section details the architecture of YOLOv8, highlighting the key components that make it well-suited for this task.

3.2.2 Overview of YOLOv8 Architecture

The architecture of YOLOv8 consists of three core components: Backbone, Neck, Head.

3.2.2.1 Backbone: Feature Extraction

YOLOv8 uses an advanced convolutional neural network (CNN) backbone. It is designed to extract multi-scale features from input images. YOLOv8 uses a backbone

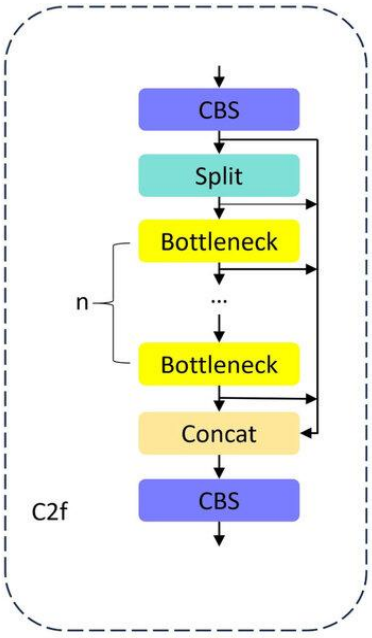


Figure 3.1: C2f Module Architecture (Shi et al., 2024)

like YOLOv5. But it introduced the C2f module (Cross-Stage Partial bottleneck with two convolutions) in place of the CSPLayer. The C2f module (Cross-Stage Partial bottleneck with two convolutions) is designed to enhance feature fusion by efficiently combining high-level and contextual information. It replaces the CSPLayer from YOLOv5, using a lighter yet powerful architecture to improve detection accuracy while reducing computational overhead. The backbone captures detailed hierarchical features, representing both low-level textures and high-level semantics crucial for object detection. It balances speed and accuracy by integrating advanced techniques like depth wise separable convolutions or other efficient layers. It does so by reducing computational load while maintaining strong feature representation for precise and efficient object detection.

3.2.2.2 Neck: Multi-Scale Feature Fusion

The neck module in YOLOv8 refines and integrates the multi-scale features captured by the backbone. It is done to enhance object detection across varying sizes and scales. It uses an improved version of the Path Aggregation Network (PANet), which facilitates efficient information flow between feature levels. PANet is designed to improve feature fusion by combining high-resolution and low-resolution features. This is crucial for detecting objects of various sizes with precision. The enhanced PANet in YOLOv8 incorporates optimizations to reduce memory usage and computational costs while maintaining high detection accuracy. This makes the neck a vital component for robust object detection tasks.

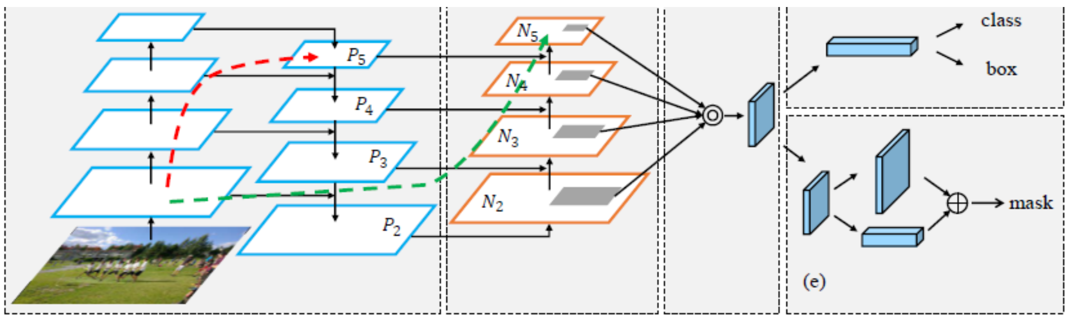


Figure 3.2: PANet Architecture (Liu et al., 2018)

3.2.2.3 Head: Anchor-Free Object Detection

The head module in YOLOv8 is responsible for producing the final predictions, including bounding box coordinates, object confidence scores, and class labels, based on the refined features from earlier stages. Unlike previous YOLO versions, YOLOv8 adopts an anchor-free approach to bounding box prediction. This simplifies the process, reduces hyperparameters, and improves adaptability to objects with varying aspect ratios and scales.

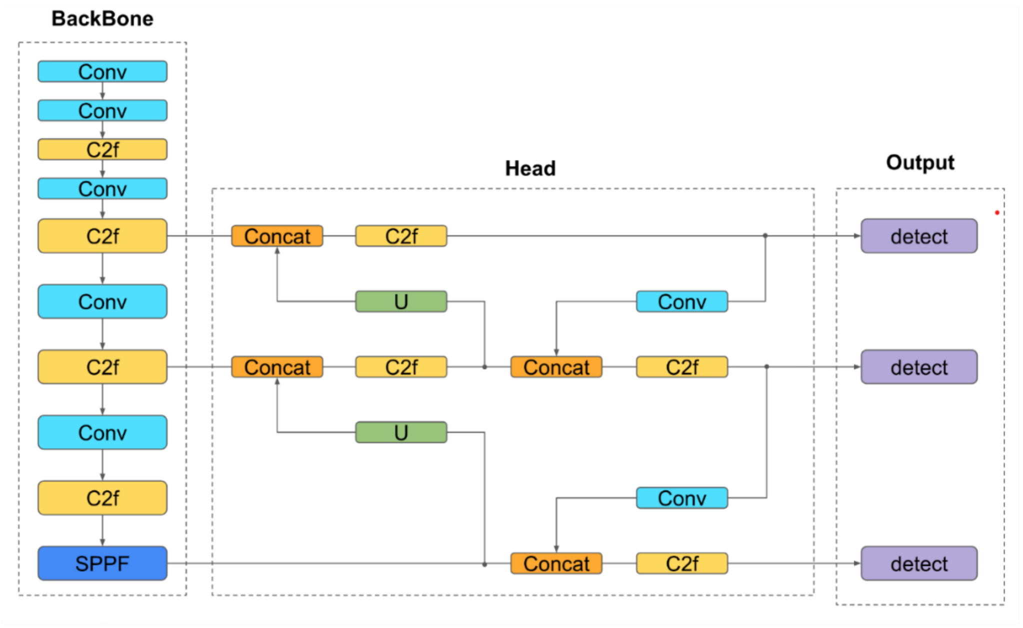


Figure 3.3: YOLOv8 architecture visualization (Sohan et al., 2024)

3.2.3 Justification for using YOLOv8 for Nighttime Road User Detection

The YOLOv8 model employs a decoupled head architecture, allowing objectness (Objectness refers to the probability that a proposed region or bounding box contains any object, as opposed to being background. It is a measure of how confident the model is that a given box contains an object of interest, regardless of its specific class.), classification, and regression tasks to be processed independently. This specialization improves overall accuracy by enabling each branch to focus on its specific task. In the output layer, YOLOv8 uses a sigmoid activation function for objectness scores, representing the probability of a bounding box containing an object. It also uses a softmax function for class probabilities, which assigns the likelihood of an object belonging to a particular class.

YOLOv8 is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box. Anchor boxes were a notoriously tricky part of earlier YOLO models, since they may represent the distribution of the target benchmark's boxes but not the distribution of the custom dataset.

3.2.4 Loss Functions for Object Detection

For training, YOLOv8 utilizes CIoU and Distribution Focal Loss (DFL) for bounding box loss and binary cross-entropy for classification loss. These loss functions significantly enhance detection performance, especially for smaller objects. This carefully designed head ensures YOLOv8 remains both accurate and efficient.

3.3 Multi-Object Tracking with DeepSORT

Accurately tracking pedestrians across multiple frames is essential for analyzing movement patterns and extracting meaningful trajectory data. To achieve this, DeepSORT is employed as the tracking algorithm, enhancing traditional SORT with deep feature embeddings for more reliable pedestrian identity tracking.

3.3.1 Overview of DeepSORT

DeepSORT (Deep Learning-based SORT) is an advanced algorithm for multiple objects tracking that builds upon the earlier SORT (Simple Online and Realtime Tracking) framework. It has become one of the most popular techniques in the computer vision field due to its ability to efficiently track multiple objects in real-time. DeepSORT is widely used in applications where object tracking is crucial.

The basic principle behind DeepSORT is to track objects across frames in a video by using both spatial and appearance information. It enhances the original SORT algorithm by integrating a deep learning component to extract appearance features. This significantly improves tracking performance in challenging scenarios, such as crowded environments or when objects are partially occluded.

3.3.2 Development of DeepSORT

The whole development of DeepSORT can be divided into two parts: SORT algorithm and DeepSORT enhancements.

3.3.2.1 SORT Algorithm

SORT is a lightweight tracking algorithm. It uses a Kalman filter for state prediction and the Hungarian algorithm for data association. The Kalman filter predicts the object's position in the next frame based on its previous state (position and velocity). The Hungarian algorithm is used to associate detected objects with the predicted states of tracked objects by minimizing the cost of assignment based on spatial proximity. SORT is efficient and works well in real-time. But it struggles in situations involving occlusions, re-identification of objects after they are temporarily lost, or when multiple objects have similar appearance characteristics.

3.3.2.2 DeepSORT Enhancement

DeepSORT improves upon SORT by incorporating a deep convolutional neural network (CNN) to extract appearance features of detected objects. These appearance features are critical for distinguishing between objects that may be spatially close or have similar motion patterns. By using a CNN, DeepSORT creates a robust feature representation for each object, which helps it maintain tracking accuracy, even when objects are occluded or change appearance. Additionally, DeepSORT uses these appearance features in conjunction with Kalman filtering and the Hungarian algorithm for more precise data association and object identification.

3.3.3 Architecture and Components of DeepSORT

DeepSORT consists of four components and each of them help in different stages of the tracking task.

3.3.3.1 Kalman Filter

The Kalman filter is used in both SORT and DeepSORT to predict the state of an object. In DeepSORT, the Kalman filter provides an estimate of the object's position and velocity in the next frame based on previous detections. This helps smooth the trajectory of tracked objects, even when some detections may be lost or delayed due to temporary occlusions. The filter's output is then used as the predicted location for data association.

3.3.3.2 Appearance Feature Extraction

DeepSORT uses a deep neural network to extract appearance features from the detected objects. These features capture the unique visual characteristics of each object and are essential for distinguishing between objects that are close to one another or have similar motion patterns. The extracted appearance embeddings are then used to match detected objects across frames, providing more reliable and accurate tracking.

3.3.3.3 Data Association (Hungarian Algorithm)

After the Kalman filter provides predicted locations for tracked objects, and the appearance features are extracted for detected objects, DeepSORT applies the Hungarian algorithm to associate detections with predicted object states. The Hungarian algorithm minimizes the cost of assignment by considering both the spatial proximity between predicted and detected object locations and the similarity of their appearance features. This approach helps resolve ambiguous situations, such as when two objects are in close proximity or when one object briefly disappears from view and reappears.

3.3.3.4 Tracking and ID Management

DeepSORT uses the Kalman filter to maintain the identity of each tracked object across frames. Once a detection is associated with a tracked object, the system updates the object's state and appearance model. If a new object appears in the frame, a new object ID is assigned, ensuring that each object maintains a unique identifier over time. This consistent ID assignment helps track objects across multiple frames, even in scenarios where they temporarily occlude each other or move in close proximity.

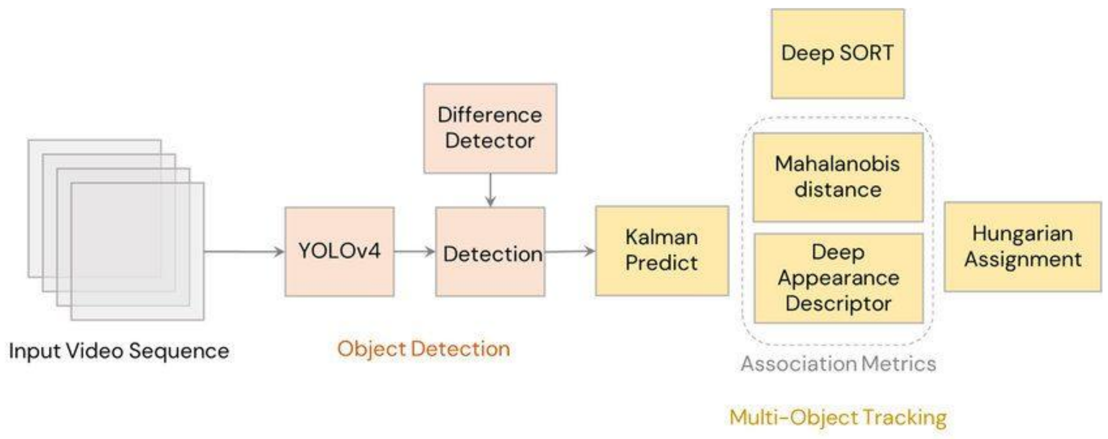


Figure 3.4: DeepSORT Architecture (Parico & Ahamed, 2021)

3.3.4 Overview of DeepSORT in Pedestrian Tracking

Tracking pedestrians in dynamic environments presents significant challenges due to erratic movement, occlusions, and varying environmental conditions such as lighting differences. To address these challenges, this study employs DeepSORT (Deep Simple Online and Realtime Tracking), an enhanced version of the original SORT (Simple Online and Realtime Tracking) algorithm.

DeepSORT works in conjunction with an object detection model, in this case, YOLOv8 (You Only Look Once, Version 8), to first detect pedestrians in individual video frames. The detected pedestrians are then assigned unique identities, which are maintained throughout the video sequence by associating detections in consecutive frames. The algorithm combines motion-based tracking with deep feature embeddings, enabling it to handle scenarios where pedestrians temporarily disappear from view due to occlusions or overlapping objects.

DeepSORT was chosen for this study due to several key advantages:

1. Robust Identity Association: Unlike traditional tracking-by-detection models that rely solely on object locations, DeepSORT enhances pedestrian identity preservation by incorporating deep feature representations.

2. Handling Occlusions: Pedestrians in real-world traffic often get momentarily hidden behind vehicles or other obstacles. DeepSORT's feature-based re-identification mechanism ensures that their trajectories are not lost.

3. Efficiency in Real-Time Scenarios: DeepSORT is computationally efficient, making it suitable for real-time pedestrian monitoring applications in urban traffic environments.

4. Scalability: The algorithm can track multiple pedestrians simultaneously, making it effective in high-density pedestrian areas.

Challenges in Pedestrian Tracking

Despite its advantages, pedestrian tracking with DeepSORT presents several challenges:

1. Lighting Conditions: At night, poor illumination and glare from vehicle headlights can degrade tracking accuracy.

2. Erratic Movement Patterns: Pedestrians exhibit unpredictable movement, frequently changing speed and direction, making trajectory association difficult.

3. Heavy Occlusions in Mixed-Traffic Environments: Vehicles, street furniture, and other pedestrians can obscure movement, leading to potential identity switches.

In the next section, we delve into the implementation process of DeepSORT for pedestrian trajectory extraction, detailing the methodology used to generate meaningful trajectory data for chaos-based analysis.

3.3.5 Implementation of DeepSORT for Trajectory Extraction

After training, the best.pt file is extracted, which serves as the pre-trained model for pedestrian detection. This file is used alongside DeepSORT to track pedestrians in video frames. The tracking process involves detecting pedestrians using YOLOv8, assigning unique IDs to each pedestrian, and maintaining these identities across multiple frames to generate reliable movement trajectories.

Step-by-Step Tracking Process

1. Load the YOLOv8 Model: The best.pt file is loaded to detect pedestrians in each frame of the video.

2. Apply DeepSORT for Tracking: The detections is passed into DeepSORT to track pedestrian movements across frames.

3. Assign Unique IDs: Each pedestrian is assigned an identity, which is maintained as they moved through the scene.

4. Handle Occlusions: In cases where pedestrians are momentarily blocked by vehicles or other obstacles, DeepSORT's appearance feature matching helps re-identify them upon reappearance.

5. Save Trajectory Data: The extracted pedestrian trajectories are stored in a CSV file for further analysis.

Pseudocode for the Tracking Pipeline is provided in the appendix.

To facilitate further analysis, only the pedestrian class's trajectory is saved in a CSV file. The CSV file format was structured as follows:

['Frame', 'ID', 'X', 'Y', 'Width', 'Height']

where (X, Y) represents the center coordinates of the pedestrian bounding box. This data is later utilized for analyzing movement stability using Lyapunov Exponent and Approximate Entropy (ApEn) Analysis.

3.4 Evaluation Metrics for Object Detection Models

Evaluating object detection models requires multiple performance metrics that assess the ability of the model to accurately detect, classify, and localize objects in images.

3.4.1 Mean Average Precision (mAP)

Mean Average Precision (mAP) is a widely used metric in object detection that evaluates both the precision and recall of a model across multiple classes. It measures how well the model detects objects at different confidence thresholds. It is calculated in the following ways:

Precision-Recall Curve: Precision is the proportion of correctly predicted objects among all detected objects. Recall is the proportion of correctly predicted objects among all actual objects.

Average Precision (AP): The area under the Precision-Recall curve for each class.

Mean Average Precision (mAP): The average of AP values across all classes.

Variants:

mAP@0.5: Measures AP when the Intersection over Union (IoU) threshold is set to 0.5.

mAP@0.5:0.95: Measures AP at different IoU thresholds ranging from 0.5 to 0.95 in steps of 0.05.

Interpretation:

Higher mAP indicates better detection performance. mAP@0.5 is easier to achieve, whereas mAP@0.5:0.95 is more stringent, requiring better localization.

3.4.2 Precision

Precision measures how many of the detected objects are correct.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{3.1}$$

Interpretation:

High Precision means that most detected objects are correct. Low Precision means many false positives (incorrect detections). Precision is crucial in applications where false detections can have serious consequences (e.g., autonomous driving).

3.4.3 Recall

Recall measures how many actual objects were correctly detected by the model.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{3.2}$$

Interpretation:

High Recall means the model is detecting most objects. Low Recall indicates the model is missing many objects (false negatives).

3.4.4 Precision-Recall (PR) Curve

A Precision-Recall (PR) curve visualizes the trade-off between precision and recall at different confidence thresholds.

Interpretation:

A well-performing model will have a PR curve closer to the top right (high precision and high recall). A model with low recall may still achieve high precision, but it may fail to detect many objects. The area under the PR curve (AUC-PR) is a good measure of overall model performance.

3.4.5 F1-Score

F1-score is the harmonic mean of precision and recall, balancing both metrics.

$$\mathrm{F1-score} = 2\,x\,\frac{Precision\ x\ Recall}{Precision + Recall} \qquad (3.3)$$

Interpretation:

High F1-score (close to 1.0) means a balanced model with good precision and recall. Low F1-score (close to 0.0) means the model is poor at detecting objects.

3.4.6 Intersection over Union (IoU)

IoU measures the overlap between the predicted bounding box and the ground-truth bounding box.

$$\mathrm{IoU} = \frac{Area\ of\ Overlap}{Area\ of\ Union} \qquad (3.4)$$

Interpretation:

IoU = 1 means perfect match. IoU = 0 means no overlap. IoU ≥ 0.5 (used in mAP@0.5) is considered a correct detection. IoU ≥ 0.75 (used in stricter evaluations) ensures highly precise localization.

3.4.7. Confidence-Score-Based Metrics

Precision-Confidence Curve

Shows how precision changes as the model's confidence threshold varies. Higher confidence values usually lead to higher precision but lower recall.

Recall-Confidence Curve

Shows how recall changes as the model's confidence threshold varies. Higher confidence thresholds often reduce recall but improve precision.

3.4.8. Loss Functions During Training

Bounding Box Loss (CIoU)

CIoU (Complete IoU) loss measures how well the predicted bounding box aligns with the ground truth, considering IoU, distance between box centers, aspect ratio differences Lower CIoU loss indicates better object localization.

Classification Loss (Binary Cross-Entropy)

Determines how well the model predicts the correct class labels for objects. Lower classification loss means the model is more confident in its class predictions.

Distribution Focal Loss (DFL)

Helps refine bounding box predictions by giving more weight to difficult-to-predict cases.

Table 3.1 Summary of Metrics and Their Significance

Metric	**Measures**	**Ideal Value**	**Interpretation**
mAP@0.5	Detection accuracy at IoU 0.5	High	Good object detection capability
mAP@0.5:0.95	Detection accuracy across IoUs	High	More stringent evaluation of localization quality
Precision	Correctness of detected objects	High	Fewer false positives
Recall	Percentage of actual objects detected	High	Fewer false negatives
F1-Score	Balance between precision and recall	~1	Strong performance in both metrics
IoU	Bounding boxes overlap accuracy	High (>0.5)	Good localization of objects
PR Curve	Trade-off between precision & recall	AUC close to 1	Well-balanced model performance
Confidence Curves	Relationship between precision/recall & confidence	Steady decline	Optimal confidence threshold selection
Bounding Box Loss	Localization accuracy	Low	Better object localization
Classification Loss	Correct class predictions	Low	Fewer misclassifications

3.5 Chaos Theory in Pedestrian Trajectory Analysis

Pedestrian movement, especially in mixed-traffic conditions, often exhibits unpredictable and nonlinear behavior, making traditional analysis methods insufficient. Chaos theory provides a mathematical framework to quantify this unpredictability, offering deeper insights into movement stability and disorder.

3.5.1 Chaos Theory

Chaos theory is a branch of mathematics that deals with the behavior of nonlinear dynamic systems that are highly sensitive to initial conditions. It explains how small changes in initial conditions can lead to drastically different outcomes. This phenomenon is popularly known as Butterfly Effect. Despite this unpredictability, chaotic systems are deterministic, meaning that their behavior is governed by underlying rules or equations.

Chaos theory is widely applied in various fields, including physics, engineering, biology, economics, and transportation. In transportation studies, it is particularly useful in analyzing pedestrian movement, which exhibits complex, dynamic, and often unpredictable patterns. The analysis of pedestrian trajectories aligns well with the principles of chaos theory. This is especially effective at nighttime when external factors such as low visibility and sudden vehicle movements influence behavior.

A chaotic system exhibits the following key properties:

1. Sensitivity to Initial Conditions: Small variations in initial conditions can cause significant deviations in system behavior over time.

2. Nonlinearity: The system’s output is not directly proportional to its input due to feedback loops and interactions.

3. Deterministic Nature: Despite appearing random, chaotic systems follow deterministic laws.

4. Strange Attractors: The system follows a specific yet irregular trajectory within a confined space, known as an attractor.

5. Fractal Structure: Many chaotic systems exhibit self-similarity across different scales.

3.5.2 Lyapunov Exponents for Stability Measurements

The Lyapunov exponent is a fundamental concept in dynamical systems and chaos theory, used to measure the sensitivity of a system to initial conditions. It provides a quantitative assessment of how small perturbations in the system's state evolve over time, offering insight into stability, predictability, and chaotic behavior.

The Lyapunov exponent determines whether small differences in an initial state amplify or diminish over time. If the exponent is positive, small perturbations grow exponentially, indicating a chaotic system where long-term predictions become nearly impossible. If it is negative, perturbations decay, suggesting a stable and predictable system. An exponent close to zero indicates a neutral or marginally stable system where perturbations neither grow nor shrink significantly. Mathematically,

$$\lambda = \lim_{t \to \infty} \frac{1}{t} \ln \frac{||\delta X(t)||}{||\delta X(0)||} \tag{3.5}$$

where:

- $\delta X(0)$ represents an initial small perturbation in the system,
- $\delta X(t)$ is the perturbation after time t, and
- λ (the Lyapunov exponent) quantifies the rate at which two nearby trajectories diverge over time.

If $\lambda>0$, the system is chaotic; if $\lambda<0$, the system is stable.

3.5.2.1 Application of Lyapunov Exponents in Pedestrian Trajectory Analysis

Pedestrian movement is inherently dynamic and can be influenced by multiple factors, including traffic density, obstacles, lighting conditions, and individual behavioral choices. Traditional methods of analyzing pedestrian trajectories often rely on statistical and rule-based models, which may fail to capture the complex and chaotic nature of pedestrian movement in unstructured environments.

The application of the Lyapunov exponent in pedestrian trajectory analysis provides several advantages:

1. Measuring Stability vs. Chaos in Pedestrian Motion: Pedestrians following predictable, structured pathways (e.g., crossing at designated areas or walking along sidewalks) tend to exhibit negative Lyapunov exponents, indicating stable movement patterns. Pedestrians who display irregular, erratic movement (e.g., sudden direction changes, hesitation, jaywalking) exhibit higher Lyapunov exponents, signifying chaotic behavior. By computing Lyapunov exponents for individual pedestrian trajectories, we can quantify how structured or unpredictable a pedestrian's motion is, allowing for better understanding and prediction of pedestrian-vehicle interactions.

2. Detecting Anomalies in Pedestrian Flow

1. High Lyapunov exponent values may indicate locations where pedestrians face unexpected disruptions, such as obstacles, poor visibility, or sudden vehicle encounters.

2. Clusters of high Lyapunov exponents in specific areas may suggest unsafe pedestrian crossings, highlighting the need for improved pedestrian infrastructure, such as better lighting or clearer signage.

3. Comparing Daytime and Nighttime Pedestrian Behavior

1. Daytime pedestrian movement is generally more structured, with well-defined pathways and minimal obstructions. This results in lower Lyapunov exponents, confirming that daytime pedestrian motion is more predictable.

2. Nighttime pedestrian movement, on the other hand, tends to be more chaotic due to low visibility, fewer controlled crossings, and increased interactions with vehicles. Higher Lyapunov exponents at night suggest greater unpredictability, reinforcing the need for enhanced pedestrian safety measures.

3.5.2.2 Application in Smart Traffic Systems

The ability to measure chaos in pedestrian movement can improve AI-driven pedestrian detection models in intelligent transportation systems. By incorporating Lyapunov-based movement stability metrics, models can:

1. Predict pedestrian collision risks in real time.

2. Identify high-risk zones where pedestrian behavior is highly erratic.

3. Adapt pedestrian signal timings based on real-time pedestrian flow dynamics.

3.5.2.3 Lyapunov Exponent Calculation

Steps for calculating the Lyapunov Exponent are provided below

1. Preprocessing of Trajectory Data

The pedestrian tracking results stored in CSV format are loaded, extracting relevant columns such as Frame, Object ID, X, and Y coordinates. Each pedestrian's movement is represented as a time-series trajectory consisting of center positions (X, Y) across frames. The dataset is grouped based on unique Object IDs, ensuring that each pedestrian's trajectory is independently analyzed.

2. Time-Delay Embedding for Phase Space Reconstruction

Since real-world pedestrian motion is inherently nonlinear, time-delay embedding is applied to reconstruct trajectories in a higher-dimensional space. The embedding process involves choosing an appropriate delay parameter (τ) and embedding dimension (m) to capture meaningful motion patterns. This provides a better representation of pedestrian dynamics and helps in tracking how movement patterns evolve over time.

3. Finding the Nearest Neighbor and Measuring Divergence

For each trajectory, nearest neighbor points are identified in the reconstructed space. The separation between these points is tracked over time to assess how trajectories evolve. This step is critical in quantifying the sensitivity of pedestrian movement to small disturbances, which directly correlates with movement stability.

4. Applying Rosenstein's Algorithm for Lyapunov Exponent Calculation

The Rosenstein Algorithm is used for computing the largest Lyapunov exponent ($\lambda 1$) from the pedestrian trajectories. This method is chosen because:

1. Handles Short Data Lengths: Pedestrian movement data is often limited in time (short trajectories), and Rosenstein's method works efficiently in such conditions.

2. Computationally Efficient: Unlike other Lyapunov exponent computation methods (e.g., Wolf's algorithm), Rosenstein's approach requires minimal computational overhead, making it ideal for real-time pedestrian trajectory analysis.

3. Robust to Noise: Given that real-world tracking data can be noisy due to occlusions and imperfect detections, Rosenstein's algorithm provides reliable estimates of movement stability.

Steps of the Rosenstein Algorithm

1. Reconstructing the Phase Space

The pedestrian trajectories are transformed using time-delay embedding to create a phase space representation. The embedding dimension (m) and time delay (τ) are selected based on standard heuristics.

2. Finding the Nearest Neighbors

For each trajectory point, the closest neighbor in phase space is located, excluding temporally close points to avoid trivial correlations. This ensures that pedestrian movement variations are not falsely correlated due to consecutive frames.

3. Computing the Logarithmic Divergence

The distance between neighboring trajectory points is monitored over time. The mean separation of initially close trajectories is computed, forming a divergence curve.

4. Estimating the Lyapunov Exponent (λ1)

A linear fit is applied to the logarithmic divergence curve. The slope of this linear fit represents the Lyapunov exponent, quantifying how quickly two initially close trajectories diverge.

5. Interpreting the Results

A positive λ1 value confirms chaotic pedestrian motion. A negative λ1 value suggests stable, predictable movement.

Visualization and Interpretation of Results

A histogram of Lyapunov exponents is generated to analyze the distribution of movement stability across different pedestrian trajectories. Comparisons are made between daytime and nighttime pedestrian behavior to assess the impact of external conditions on movement predictability. Key observations are drawn regarding traffic interactions, pedestrian hesitation, and spontaneous directional changes.

The results from this analysis provide deeper insights into how pedestrians navigate urban environments and which factors contribute to movement instability. This information is essential for improving pedestrian safety measures, optimizing urban infrastructure, and enhancing traffic management strategies.

3.5.3 Approximate Entropy for Quantifying Complexity

Approximate Entropy (ApEn) is a statistical measure used to quantify the predictability and complexity of a time series. It was introduced to assess the regularity of physiological and financial data, but its applications extend to chaotic systems, movement dynamics, and behavioral analysis.

ApEn is particularly useful for identifying patterns and irregularities in sequential data. It measures the likelihood that similar patterns of observations will remain similar over subsequent time steps. Lower ApEn values indicate a highly structured and predictable system, while higher ApEn values suggest greater randomness and irregularity.

Mathematically, ApEn is defined as:

$$ApEn(m, r, N) = \Phi^m(r) - \Phi^{m+1}(r) \tag{3.6}$$

where:

m is the embedding dimension (number of previous points considered),

r is the tolerance (threshold for similarity),

N is the total number of data points, and

$\Phi^m(r\ )$ is a measure of the probability that two sequences of length **m** remain similar when extended to **m+1**

Where $\Phi^m(r\ )$ is defined as

$$\Phi^m(r) = \frac{1}{N - m + 1} \sum_{i=1}^{N-m+1} \ln C_i^m(r) \tag{3.7}$$

Where

$C_i^m(r)$ is the relative frequency of template vectors being similar (within tolerance r) to vector *X(i)*

And *X(i)* represents a sequence of m consecutive data points from the original time series.

A higher ApEn value means a more unpredictable system, while a lower value means a more structured system.

3.5.3.1 Application of Approximate Entropy in Pedestrian Analysis

Pedestrian movement is inherently dynamic and influenced by multiple environmental and behavioral factors. Traditional trajectory analysis methods often focus on speed, direction, and acceleration, but they do not capture the level of unpredictability in movement behavior. Approximate Entropy provides a quantitative way to assess the regularity of pedestrian movement, making it valuable for safety analysis, urban planning, and traffic management. The application of the Approximate Entropy in pedestrian trajectory analysis provides several advantages:

1. Measuring Stability vs. Unpredictability in Pedestrian Motion

Approximate Entropy (ApEn) is a measure of complexity and irregularity in a timeseries dataset. It evaluates how similar patterns of movement remain consistent over time, making it a valuable tool for identifying predictable vs. unpredictable pedestrian behavior.

Low ApEn values ($\approx 0.2 - 0.5$) indicate structured movement, where pedestrians follow clear paths with minimal variations in speed or direction. Moderate ApEn values ($\approx 0.5 - 0.8$) suggest some degree of variability, likely due to minor adjustments in movement while navigating obstacles or adjusting to environmental cues. High ApEn values (≥ 0.8) indicate erratic pedestrian behavior, characterized by frequent stops, sudden direction changes, or hesitation, often seen in nighttime conditions or high-risk areas.

Unlike the Lyapunov Exponent, which identifies global trajectory stability, ApEn provides a more granular view of pedestrian motion by analyzing moment-to-moment fluctuations. This makes it particularly useful in detecting hesitation before crossing, assessing pedestrian confidence in movement, and understanding behavioral differences between daytime and nighttime walking patterns.

2. Identifying Pedestrian Hesitation and Decision-Making

Pedestrians exhibit hesitation when crossing roads or navigating obstacles. Hesitation results in fluctuations in walking speed and direction, leading to higher ApEn values. ApEn can be used to quantify how much uncertainty exists in pedestrian decision-making, especially in high-risk areas like uncontrolled intersections and jaywalking scenarios.

3. Daytime vs. Nighttime Pedestrian Behavior

Daytime movement is generally more structured, leading to lower ApEn values, as pedestrians can see clearly and follow established pathways. Nighttime movement tends to be more erratic, leading to higher ApEn values, due to factors like low visibility, unclear crossings, and pedestrian-vehicle interactions.

4. Application in Traffic Safety and Urban Planning

1. Detection of High-Risk Zones: Areas with consistently high ApEn values indicate locations where pedestrians exhibit frequent hesitations or erratic movement, signaling a need for better infrastructure (e.g., improved crosswalks, lighting, traffic signals).

2. Optimizing Pedestrian Crossings: By analyzing ApEn values at crosswalks vs. mid-block crossings, city planners can assess where pedestrian behavior is most unpredictable and design interventions accordingly.

3. AI-Based Pedestrian Prediction Models: Smart traffic systems can incorporate ApEn-based unpredictability metrics to improve pedestrian behavior forecasting and collision prevention algorithms.

3.5.3.2 Approximate Entropy Calculation

Steps for calculating Approximate Entropy are provided below

1. Preprocessing of Trajectory Data

The pedestrian tracking results stored in CSV format are loaded, extracting relevant columns such as Frame, Object ID, X, and Y coordinates. Data is sorted by Object ID and Frame to ensure that pedestrian trajectories are processed in sequential order. New columns for Velocity and Direction Change are initialized to capture movement characteristics.

2. Computing Velocity and Direction Change

The velocity of each pedestrian is computed based on the displacement between consecutive frames, using the Euclidean distance formula:

$$V = \sqrt{dx^2 + dy^2}\, X\, 30 \tag{3.8}$$

where dx and dy are the differences in X and Y positions between consecutive frames, and 30 accounts for the frame rate.

The direction change is calculated using the arctan2 function, which determines the angular variation in movement over successive frames:

$$\Delta\theta = arctan^2(dy, dx) - previous\ arctan^2(dy, dx) \tag{3.9}$$

This allows for measuring both speed variations and angular fluctuations in pedestrian movement.

3. Setting ApEn Parameters

The embedding dimension (m) is set to 2 to ensure that the analysis captures patterns based on two consecutive movement states. The tolerance (r) is set to 0.2 × Standard Deviation of the Signal, ensuring that ApEn adapts to variations in velocity and direction change across different pedestrians.

4. Applying the Approximate Entropy Algorithm

The ApEn function is applied separately to the velocity time series and the direction change time series for each pedestrian. The entropy values are stored only if they are valid (i.e., non-NaN values), ensuring robust data analysis.

Visualization and Interpretation of Results

Histograms of Approximate Entropy values are generated for both velocity and direction change. The distribution of ApEn values is analyzed to compare structured vs. chaotic pedestrian movement. Special attention is given to differences between daytime and nighttime pedestrian behavior, revealing how visibility and external factors influence movement predictability.

To effectively quantify movement irregularity, Approximate Entropy is computed separately for both velocity and direction change. This dual analysis allows for a comprehensive understanding of how pedestrians adjust their movement patterns in response to their surroundings.

Velocity-Based Approximate Entropy (ApEn_Velocity)

1. Low ApEn_Velocity values (0.2 - 0.5) indicate that pedestrians maintain a consistent walking speed, often observed in daytime conditions where visibility is high, and paths are well-defined.

2. Moderate ApEn_Velocity values (0.5 - 0.8) suggest some variation in speed, typically seen when pedestrians slow down or accelerate based on traffic signals, pedestrian crossings, or interactions with vehicles.

3. High ApEn_Velocity values (≥ 0.8) suggest erratic speed changes, often associated with hesitation, avoidance maneuvers, or abrupt stops, particularly in poorly lit nighttime environments.

Direction Change-Based Approximate Entropy (ApEn_DirectionChange)

1. Low ApEn_DirectionChange values (0.2 - 0.5) indicate stable movement directions, where pedestrians move along a defined path with minimal deviations.

2. Moderate ApEn_DirectionChange values (0.5 - 0.8) suggest some directional shifts, likely due to avoiding obstacles or adjusting walking paths in crowded areas.

3. High ApEn_DirectionChange values (≥ 0.8) indicate frequent and sudden changes in direction, which may occur due to pedestrian hesitation before crossing, uncertainty in navigation, or unexpected vehicle interactions.

By analyzing both velocity-based ApEn and direction change-based ApEn, a clear picture of pedestrian behavior emerges, allowing for more precise identification of movement irregularities and potential risk factors in urban pedestrian environments.

3.5.4 Comparison Between Stable and Chaotic Movements

Stable pedestrian movements are characterized by predictable and smooth trajectories, where individuals follow structured pathways with minimum deviations. Such movement is often observed in designated crosswalks or sidewalks with clear visibility. In contrast, chaotic pedestrian movement exhibits irregular trajectories, frequent direction changes, and sudden stops, often influenced by environmental factors such as traffic congestion, poor lighting, or unexpected obstacles. High Lyapunov Exponent values indicate sensitive dependence on initial conditions, meaning small disturbances lead to significant trajectory deviations. Similarly, Approximate Entropy values are higher in chaotic movements, reflecting the increased randomness and unpredictability in pedestrian behavior. By analyzing these indicators, this study distinguishes between stable and chaotic movement patterns, helping to identify locations and conditions that contribute to pedestrian unpredictability.

3.5.5 Justification for Using Chaos Theory in This Study

Traditional pedestrian trajectory analysis often relies on statistical and rule-based models that assume linear and predictable movement patterns. However, in real-world urban environments, especially in mixed-traffic conditions, pedestrian behavior is inherently nonlinear and influenced by multiple unpredictable factors. Chaos theory provides a powerful analytical framework to quantify this unpredictability through metrics like Lyapunov Exponents and Approximate Entropy, which measure sensitivity to initial conditions and randomness in movement. By applying these methods, this study captures complex pedestrian dynamics that conventional models fail to address. It enables a deeper understanding of pedestrian safety risks, traffic flow disruptions, and necessary infrastructural improvements.

3.6 Summary

This chapter outlines the methodological framework used for pedestrian nighttime trajectory analysis, integrating deep learning-based object detection, tracking, and chaos theory. A custom-trained YOLOv8 model was developed using a dedicated dataset to improve road user detection accuracy under nighttime conditions. DeepSORT was employed for multi-object tracking, ensuring reliable pedestrian identity tracking across frames. To analyze movement stability, Lyapunov Exponents and Approximate Entropy were applied, allowing the study to differentiate between stable and chaotic pedestrian trajectories. The methodology provides a structured approach to detecting, tracking, and analyzing pedestrian motion, addressing the challenges of nighttime traffic monitoring and advancing intelligent transportation research.

CHAPTER 4

DATA COLLECTION AND PREPROCESSING

4.1 Introduction

This chapter focuses on the data collection and preprocessing process, which forms the foundation for training the pedestrian detection model. The chapter begins by detailing the site selection criteria, emphasizing the choice of an urban highway location with high pedestrian and vehicular interaction. The data collection process is then outlined, including the setup of cameras, recording conditions, and the timeline for capturing nighttime pedestrian movement. The recorded video data undergoes preprocessing steps, including frame extraction, annotation, and dataset structuring. Annotation plays a critical role in ensuring high-quality labeled data, and this chapter explains the methodology used for precise object labeling, including bounding box formatting and dataset splitting for training, validation, and testing. Additionally, data augmentation techniques are applied to enhance the model's robustness to real-world variations. These steps collectively ensure a high-quality dataset, optimized for training the YOLOv8-based pedestrian detection model.

4.2 Site Description

4.2.1 Description of the site:

The site for video data collection was selected to be at Shanarpar Foot Over Bridge, Narayanganj, Dhaka. Shanarpar is a developing place in between Demra & Narayangonj. It is under shiddirgonj thana of Narayangonj district. The latitude and longitude of the place is 23.694809099733607, 90.48994490238138. The foot over bridge is near Shanarpar Bus Station which is at 20 meters distance. The foot over bridge is in the N1 highway.

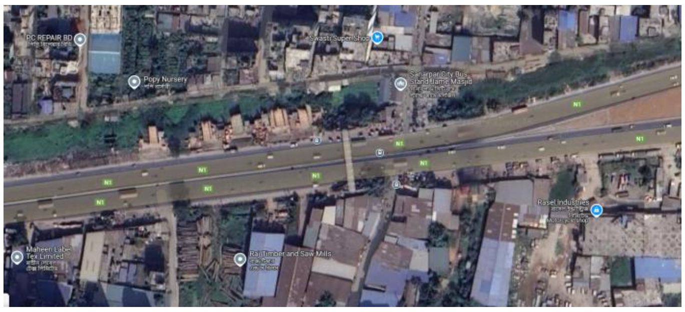


Figure 4.1: Satellite View of the Location

The Dhaka–Chittagong Highway, also known as the N1, is Bangladesh's most important transportation route. It connects the capital city of Dhaka with the southern port city of Chittagong. The highway is 465 kilometers long (*RMMS::*, n.d.). The highway begins at Jatrabari in Dhaka and extends all the way to Teknaf in Cox's Bazar. Along different segments, it is also referred to as the Chittagong–Cox's Bazar Highway or the Cox's Bazar–Teknaf Highway. Currently, the highway has four lanes, but work is underway to expand it to eight lanes to accommodate the increasing traffic. As the country's busiest road, the N1 plays a crucial role in the nation's economy and is a key focus for development projects.

The surroundings of the selected section along the N1 Highway at Shanarpar Bus Stand include a mix of commercial and industrial establishments. To the north, there are small businesses such as nurseries and repair shops, while to the south, various industries and institutions like textile mills and vocational institutes are present. The area is densely packed with urban infrastructure, including warehouses and workshops, creating a vibrant mix of economic activity.

4.2.2 Justification for Site Selection for Video Data Collection

The site was chosen strategically to ensure a comprehensive representation of diverse traffic conditions. This site was ideal as it is situated on the periphery of Dhaka City. Its unique blend of vehicular and pedestrian activity enabled to capture a broad range of transportation dynamics.

Diverse Vehicle Types

One of the primary reasons for selecting this location was the opportunity to observe all types of vehicles, both motorized and non-motorized. As part of the Dhaka-Chittagong Highway, a major transportation artery in Bangladesh, the site facilitates the movement of long-distance and local traffic. Motorized vehicles such as buses, trucks, cars, motorcycles, and auto-rickshaws frequently traverse this route, reflecting both urban and inter-city traffic patterns. Simultaneously, non-motorized vehicles, including rickshaws and bicycles, are commonly seen, especially around the bus stand. This mix of traffic provides a diverse dataset, crucial for understanding traffic behavior in a context where traditional and modern transportation coexist.

Pedestrian Activity

The Shanarpar Bus Stand is a bustling node that attracts significant pedestrian activity. Being a hub for local transportation, the area witnesses constant foot traffic as passengers board and disembark from buses and other vehicles. Additionally, the presence of nearby commercial establishments and residential areas contributes to regular pedestrian movement. This aspect was particularly important for our study, as pedestrian dynamics are a critical component of traffic analysis in urban settings, especially in regions like Bangladesh, where mixed traffic conditions are prevalent.

Influence of the Bus Stand

The presence of the bus stand further enriches the site's traffic diversity. The stand serves as a collection and dispersal point for various modes of transport, creating a microcosm of urban traffic interactions. Non-motorized vehicles, such as cycle rickshaws and vans, frequently gather here to provide last-mile connectivity for passengers. The interplay of these vehicles with motorized traffic offers valuable insights into congestion patterns, modal interactions, and overall traffic behavior.

Proximity to Dhaka-Chittagong Highway

The site's location along the Dhaka-Chittagong Highway was another significant factor in its selection. As a major national highway, the movement of a variety of vehicles, including heavy vehicles are prominent here. This ensures the inclusion of long-distance traffic patterns in the data. Moreover, the highway serves as a critical corridor for economic activity, making it an essential area for studying traffic flow and resilience in the broader transportation network.

4.3 Camera Setup and Video Recording

4.3.1 Camera Specifications and Placement

To ensure effective and comprehensive traffic data collection, two Imou WiFi Security Cameras were installed on opposite sides of the Shanarpar foot over bridge. This strategic placement allowed for detailed coverage of vehicular movement in both directions while also capturing the surrounding environment, including pedestrian activity and adjacent roadside areas.

The elevated point provided by the foot overbridge ensured unobstructed views of the highway, making it possible to monitor a wide range of traffic patterns. The cameras were positioned to maximize their field of view, capturing motorized vehicles such as buses, cars, motorcycles, and trucks, along with non-motorized vehicles like rickshaws and bicycles. Additionally, the recordings included pedestrians crossing or walking near the bus stand, making the data particularly valuable for analyzing the complex interactions between vehicles and pedestrians.

The Imou WiFi Security Cameras were chosen for their high-resolution capabilities (1080p Full HD), ensuring crystal-clear recordings essential for detailed analysis. Their range of 10 to 20 meters allowed for effective monitoring of the road and nearby areas. The cameras were also capable of capturing activity under low-light conditions due to their night vision feature, ensuring consistent data collection throughout the nighttime.

4.3.2 Recording Conditions

The data collection for this study was conducted on March 6, 2023, between 5:00 PM and 10:00 PM. Videos were captured in DAV format using stationary cameras positioned at strategic points along the Dhaka-Chittagong Highway. The video files are organized in folders named according to the start time in a 24-hour format, ensuring a systematic structure for easy navigation. For instance, files from 5:00 PM are stored in the folder named 17.00, indicating the start time of 17:00 in 24-hour notation. Each video is 5 minutes long, capturing detailed snapshots of traffic and pedestrian activity during the collection period. This approach ensured consistent data coverage for analysis. Additionally, separate cameras were positioned to capture the opposite direction of traffic and pedestrian movement.

4.3.3 Video Storage and Organization

These videos are stored in folders with a similar structure, as demonstrated by the file path E:\NightOwl\New folder\17\17.00.00-17.05.00[R][0@0][0].dav. This dual-camera setup enabled a comprehensive analysis of both directions, providing a complete view of the traffic dynamics and pedestrian behavior. The organized data collection structure, combined with the chosen timeframe, ensured the capture of varied traffic and pedestrian patterns during evening and early nighttime hours, critical for analyzing behaviors under differing visibility and traffic density conditions.

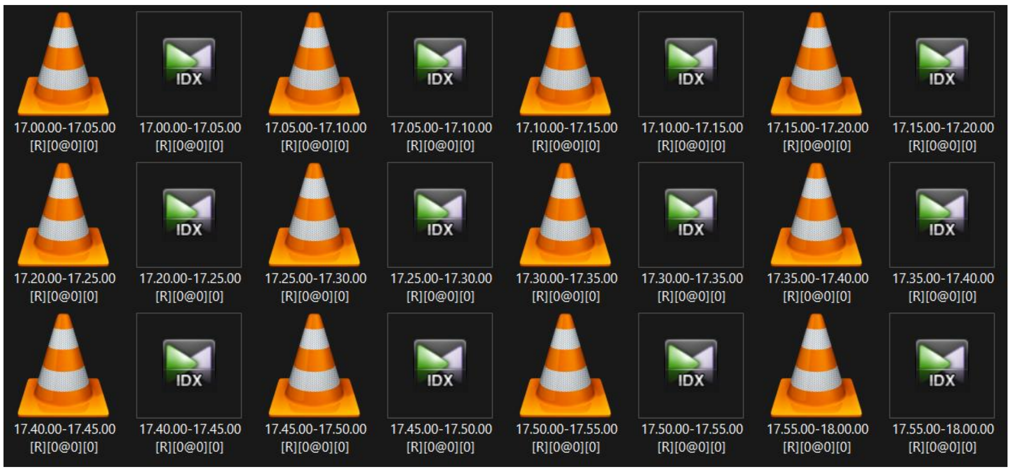

17.00.00-17.05.00 [R][0@0][0]
17.00.00-17.05.00 [R][0@0][0]
17.05.00-17.10.00 [R][0@0][0]
17.05.00-17.10.00 [R][0@0][0]
17.10.00-17.15.00 [R][0@0][0]
17.10.00-17.15.00 [R][0@0][0]
17.15.00-17.20.00 [R][0@0][0]
17.15.00-17.20.00 [R][0@0][0]
17.20.00-17.25.00 [R][0@0][0]
17.20.00-17.25.00 [R][0@0][0]
17.25.00-17.30.00 [R][0@0][0]
17.25.00-17.30.00 [R][0@0][0]
17.30.00-17.35.00 [R][0@0][0]
17.30.00-17.35.00 [R][0@0][0]
17.35.00-17.40.00 [R][0@0][0]
17.35.00-17.40.00 [R][0@0][0]
17.40.00-17.45.00 [R][0@0][0]
17.40.00-17.45.00 [R][0@0][0]
17.45.00-17.50.00 [R][0@0][0]
17.45.00-17.50.00 [R][0@0][0]
17.50.00-17.55.00 [R][0@0][0]
17.50.00-17.55.00 [R][0@0][0]
17.55.00-18.00.00 [R][0@0][0]
17.55.00-18.00.00 [R][0@0][0]

Figure 4.2: File Organization

4.4 Video Processing and Frame Extraction

4.4.1 Conversion of Video Files

The recorded videos were originally saved in DAV format, a proprietary video file format commonly used in surveillance systems. To ensure compatibility with video processing and machine learning frameworks, these files were converted to the MP4 format using VLC Media Player. MP4 was chosen due to its widespread compatibility, efficient compression, and preservation of visual quality, making it suitable for further processing.

The conversion process was carried out systematically, ensuring no loss of critical information. The steps involved:

1. Loading the DAV files into VLC Media Player.

2. Selecting the appropriate codec to maintain resolution and frame rate consistency.

3. Batch processing multiple videos for efficiency.

4. Verifying that the converted MP4 files retained the expected resolution (1280×720) and frame rate (30 FPS).

This conversion was crucial for the frame extraction and annotation process, as most deep learning frameworks and annotation tools do not natively support DAV files. By converting the files to MP4, the dataset became more accessible for further analysis, ensuring smooth integration with YOLOv8 training pipelines.

4.4.2 Frame Extraction Methodology

To create a structured dataset for pedestrian detection, frames were extracted from the converted MP4 videos at regular intervals. This process ensured that the dataset captured diverse pedestrian movements, lighting conditions, and vehicle interactions over time. Extracting frames at fixed intervals reduces redundancy while maintaining temporal consistency, which is crucial for training an effective object detection model.

For this study, frames were extracted every 5 seconds, ensuring a balance between dataset diversity and computational efficiency. A higher extraction rate would lead to excessive similarity between frames, while a lower rate might result in missing key pedestrian movements. The extraction process was automated using Python and OpenCV, enabling seamless and systematic processing of the recorded videos.

Pseudocode for frame extraction is provided in the appendix.

4.4.3 Dataset Size and Frame Count

After completing the frame extraction process, a total of 6,953 images were generated from the recorded videos. These images were stored in JPG format, ensuring high-quality visuals while maintaining manageable file sizes for efficient processing. The dataset was structured in a well-organized folder hierarchy, with frames categorized based on their time of capture to facilitate easy annotation and retrieval.

The extracted dataset provides a diverse representation of nighttime pedestrian activity, capturing variations in lighting conditions, pedestrian movements, vehicle interactions, and occlusions. This diversity is essential for training a robust YOLOv8 model, ensuring it can generalize well across different real-world nighttime scenarios.

4.5 Annotation and Dataset Preparation

4.5.1 Dataset processing

The dataset was annotated using the Roboflow platform, leveraging Smart Polygon annotation to improve the precision of object boundaries. Unlike traditional rectangular bounding boxes, which may include unnecessary background pixels, Smart Polygon annotation allows for a more accurate representation of objects with irregular shapes, such as motorcycles, bicycles, and pedestrians. This approach enhances detection accuracy by ensuring the model learns more precise object features. During the annotation process, 16 object classes were selected for training based on Bangladesh Road Transport Authority (BRTA) data.

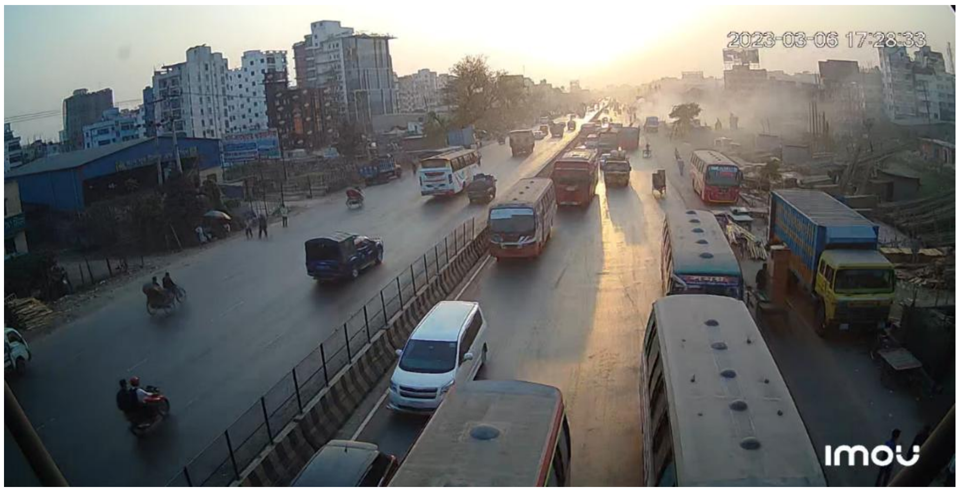


Figure 4.3: Sample Frame 1

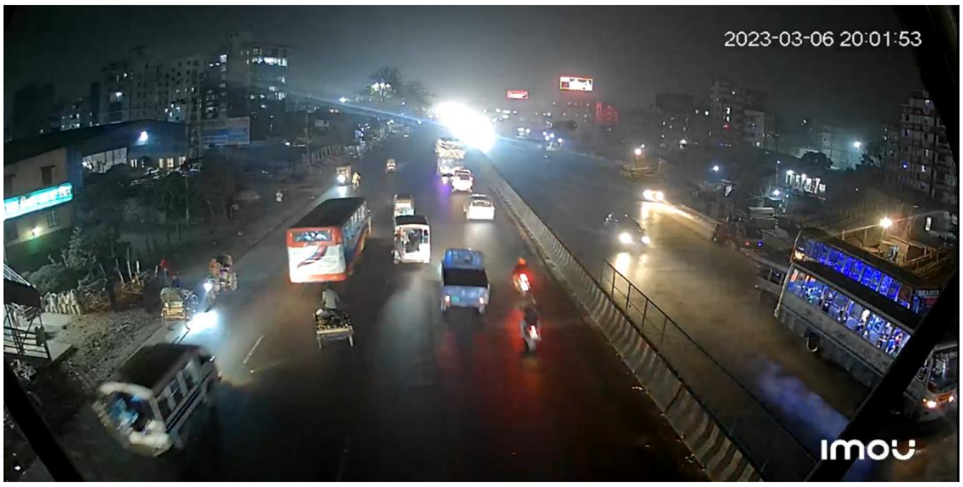


Figure 4.4: Sample Frame 2

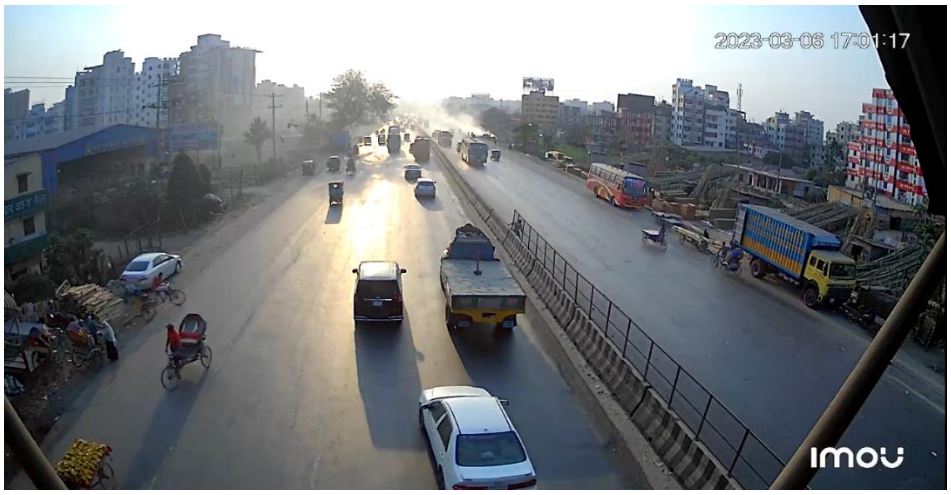


Figure 4.5: Sample Frame 3

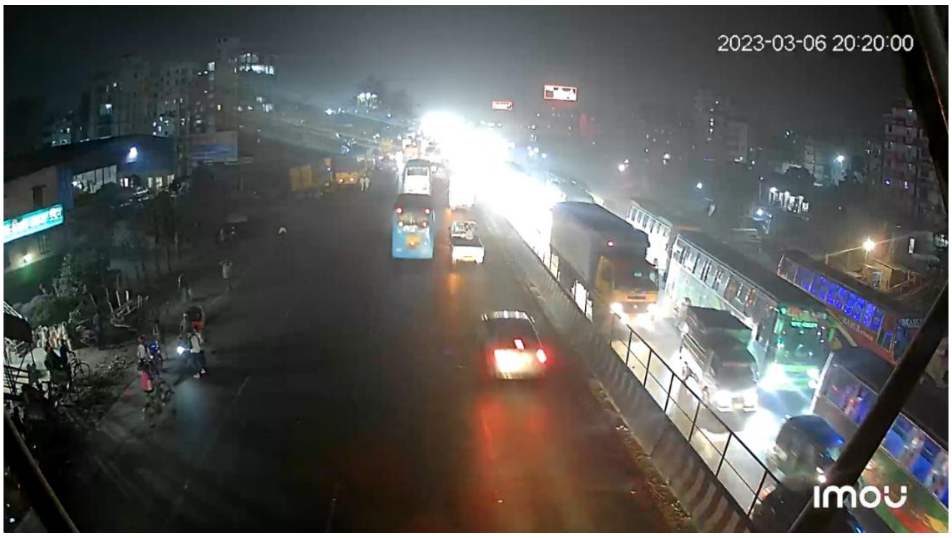


Figure 4.6: Sample Frame 4

These classes include:

1. Ambulance
2. Auto-Rickshaw
3. Bicycle
4. Bus
5. C.N.G.
6. Covered-Van
7. Cycle-Rickshaw
8. Human-Hauler
9. Microbus
10. Motor-Cycle
11. Pedestrian
12. Pick-Up
13. Private-Passenger-Car
14. Non-Motorized-Van
15. Special-Purpose-Vehicles

16. Truck

Sample images of every class is provided below:

Table 4.1: Class Images

Class	Image
Ambulance	
Auto-Rickshaw	
Bicycle	
Bus	
C.N.G.	
Covered-Van	

Cycle-Rickshaw	
Human-Hauler	
Microbus	
Motor-Cycle	
Pedestrian	
Pick-Up	
Private-Passenger-Car	

Non-Motorized-Van	
Special-Purpose-Vehicles	
Truck	

To ensure high-quality annotations, objects were labeled as long as they were visibly distinguishable to the human eye, even under nighttime conditions. This method reduces misclassification and confusion among classes, particularly in low-visibility settings. Once the annotation process was completed, the dataset was formatted and structured for model training. Initially, no preprocessing or data augmentation was applied, but all images were resized to 640 × 640 pixels, the standard input dimension for YOLOv8. Furthermore, no train-test-validation split was performed at this stage. Instead, the dataset consisted of image files and corresponding label files.

4.5.2 Label Formatting for YOLOv8

YOLOv8 requires a specific label format where each label is stored as a text file containing:

<Class ID> <X Coordinate of center> <Y Coordinate of center> <Width> <Height>

where:

- <Class ID> represents the object category.
- <X Coordinate of center> <Y Coordinate of center> are the normalized coordinates of the bounding box center.
- <Width>, <Height> represent the normalized width and height of the bounding box relative to the image dimensions.

This format is widely used in the YOLO family because it is efficient, compact, and well-suited for fast object detection. Each line in the text file represents one detected object in the image.

However, since Smart Polygon annotation was used, a Python script was implemented to convert the polygon-based labels into the required YOLOv8 bounding box format by computing the bounding box coordinates from polygon annotations.

Pseudocode for converting the labels is provided in the appendix.

4.6 Dataset Splitting for Model Training

Once the dataset was structured and labeled, it was divided into three subsets to ensure optimal model training and evaluation:

Training Set – Used to train the model and learn patterns from the dataset.

Validation Set – Used to fine-tune hyperparameters and prevent overfitting.

Test Set – Used for final evaluation, ensuring the model generalizes well to unseen data.

A YAML configuration file was created to specify the dataset paths for YOLOv8 training. The dataset was split using an 80-10-10 ratio for training, validation, and testing, respectively. This ensures a balanced distribution of images, preventing bias in model evaluation.

Justification for Dataset Splitting

When developing machine or deep learning models, it is essential to divide the dataset into three distinct subsets: training, validation, and test sets. This separation ensures that the model is trained effectively, fine-tuned appropriately, and evaluated rigorously to generalize well on unseen data.

Training Set: The training set is used to train the machine learning model by adjusting its internal parameters to learn patterns and relationships in the data. It typically constitutes the largest portion of the dataset, allowing the model to generalize well when exposed to new data.

Validation Set: The validation set is used to fine-tune hyperparameters and prevent overfitting. During training, the model's performance is evaluated on the validation set to ensure that it is not memorizing the training data but rather learning meaningful patterns. Techniques such as cross-validation help optimize model performance before final testing.

Test Set: The test set is a separate subset used to assess the final performance of the trained and validated model. This set serves as an unbiased evaluation metric to measure the model’s generalization ability on unseen data, ensuring its reliability for real-world applications.

By splitting the dataset into these three subsets, the model's performance can be assessed effectively.

4.7 Data Augmentation for Model Robustness

To improve the generalization ability of the detection model, an augmented dataset was created using Roboflow. Augmentation helps to artificially expand the dataset by applying transformations, making the model robust to real-world variations such as lighting changes, occlusions, and camera distortions.

Data augmentation is a widely used technique in deep learning to artificially expand a dataset by applying transformations such as cropping, rotation, brightness adjustments, and noise injection. In object detection, augmentation enhances model generalization, mitigates overfitting, and improves robustness to variations in real-world conditions. By modifying images while preserving label consistency, augmentation enables models to learn invariant representations of objects, leading to higher detection accuracy (Shorten & Khoshgoftaar, 2019).

The primary reasons for using augmentation in object detection tasks include:

Increasing Dataset Diversity: Augmentation helps generate diverse variations of training samples, which is essential when the available dataset is limited. This improves the model's ability to generalize across different real-world scenarios (Shorten & Khoshgoftaar, 2019).

Reducing Overfitting: Deep learning models, especially CNN-based object detectors, are prone to overfitting when trained on small datasets. Augmentation introduces variability, preventing the model from memorizing specific patterns and instead focusing on general features (Wong et al., 2016).

Improving Model Robustness: Variations in lighting, perspective, and background clutter can affect detection performance. Augmentations such as brightness changes, noise addition, and geometric transformations help the model become invariant to such changes (Yun et al., 2019).

Handling Class Imbalance: When certain object classes have fewer samples in the dataset, augmentation can artificially balance the distribution by increasing the number of samples for underrepresented classes (Inoue, 2018).

Simulating Real-World Conditions: Object detection models deployed in real-world applications, such as autonomous vehicles or surveillance, must handle occlusions, viewpoint changes, and environmental conditions. Augmentation prepares models to handle such variations effectively (Zoph et al., 2019).

Augmentation Techniques Applied

The following augmentation techniques were applied using the Roboflow platform:

Crop:

Implementation: Cropping was applied with a minimum zoom of 0% and a maximum zoom of 20%.

Purpose: This augmentation simulates objects at different scales and positions within an image. It ensures the model is robust to variations in object size and framing.

Impact: Cropping improves the model's ability to detect objects in varying spatial contexts.

Rotation:

Implementation: Images were rotated randomly between -15° and +15°.

Purpose: Rotation helps the model recognize objects from different angular perspectives.

Impact: This augmentation improves generalization, enabling better detection in tilted or rotated views.

Shear:

Implementation: Shearing was applied horizontally and vertically within a range of ±10°.

Purpose: Shearing creates a distortion effect, simulating changes in perspective or camera angles.

Impact: It helps the model learn to detect objects under skewed or distorted conditions.

Grayscale:

Implementation: 15% of the images were converted to grayscale.

Purpose: Simulates low-color or grayscale environments, ensuring the model does not overly depend on color features.

Impact: The model becomes resilient to color variations.

Hue:

Implementation: Random adjustments were made to hue within the range of -15° to +15°.

Purpose: Introduces variability in color tones, simulating different lighting conditions.

Impact: Enhances generalization under varying lighting environments.

Saturation:

Implementation: Saturation was adjusted between -25% and +25%.

Purpose: Alters color intensity to simulate diverse real-world scenarios.

Impact: Prepares the model to handle data with varying vividness.

Brightness:

Implementation: Brightness adjustments ranged from -15% to +15%.

Purpose: Simulates different lighting intensities to improve adaptability.

Impact: Reduces sensitivity to lighting variations.

Exposure:

Implementation: Exposure adjustments were applied in the range of -10% to +10%.

Purpose: Simulates underexposed and overexposed environments.

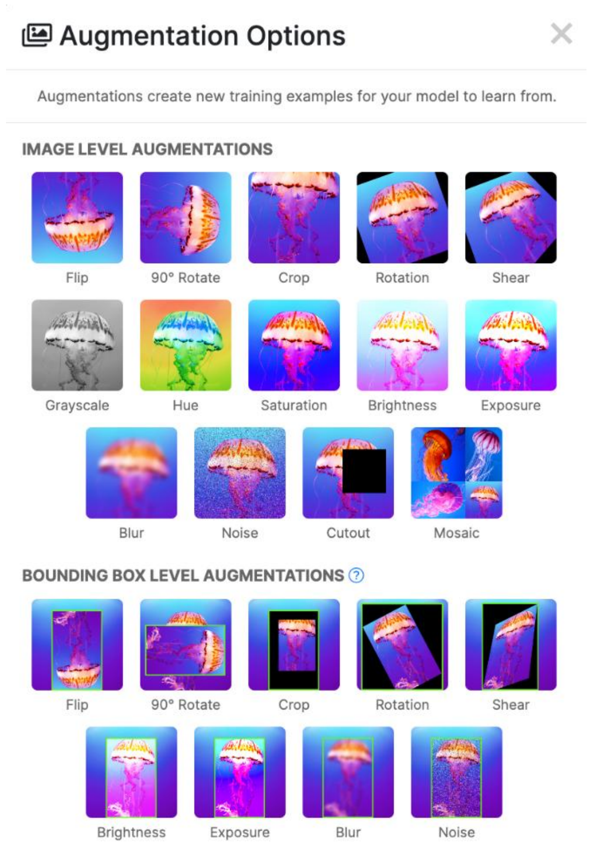
Impact: Prepares the model for challenging lighting conditions.

Noise:

Implementation: Noise was added to up to 0.1% of the pixels.

Purpose: Mimics sensor noise and image artifacts.

Impact: Enhances robustness to imperfections in input data.


Augmentation Options

Augmentations create new training examples for your model to learn from.

Figure 4.7: Augmentation Options in Roboflow

Exclusion of Mosaic and Cutout: Mosaic and Cutout augmentations were not applied manually as these techniques are already integrated into the training pipelines of modern object detection models like YOLOv8. Their manual application during preprocessing could lead to:

Over-augmentation: Redundant transformations may reduce dataset diversity.

Training instability: Conflicts between internal and external augmentation strategies.

4.8 Summary

This chapter outlined the data collection and preprocessing pipeline, which is essential for training an accurate pedestrian detection model. The dataset was collected from Shanarpar Foot Over Bridge, an urban highway site chosen for its diverse traffic conditions. High-resolution video recordings were converted, processed, and frames were extracted systematically to create a structured dataset.

The dataset was then annotated using Smart Polygon labeling, ensuring precise object boundaries for 16 traffic-related classes. After formatting the labels for YOLOv8, the dataset was split into training, validation, and test sets, maintaining an optimal balance for model training. Finally, data augmentation techniques were applied to improve model generalization and adaptability to nighttime conditions.

These preprocessing steps ensure that the YOLOv8-based detection model is trained on a high-quality dataset, enabling robust detection and trajectory analysis in real-world urban environments.

CHAPTER 5

MODEL BUILDING AND VALIDATION

5.1 Introduction

This chapter focuses on the model training and evaluation process for detection in nighttime conditions. The primary objective is to develop a robust deep learning model using YOLOv8 that can accurately detect road users at nighttime. The chapter covers the experimental setup, model training process, hyperparameter selection, validation strategies, and model performance evaluation. Various optimization techniques, loss functions, and computational considerations are also discussed to ensure an efficient and accurate detection model.

5.2 Experimental Design

5.2.1 Hardware and Software Specifications

The model training was conducted using the following hardware and software specifications:

- GPU: NVIDIA RTX 4070 Ti Super (16GB GDDR6X VRAM)
- RAM: 32GB DDR4
- Deep Learning Framework: PyTorch 2.0
- Other Tools: Roboflow for annotation, OpenCV for image preprocessing

Deep learning models for traffic detection, particularly those based on object detection frameworks like YOLOv8, require substantial computational power for efficient training and inference. To ensure optimal performance and reduced processing time, the NVIDIA RTX 4070 Ti Super (16GB GDDR6X) GPU was utilized in this study.

Technical Specifications of the GPU

The NVIDIA RTX 4070 Ti Super, a part of NVIDIA's Ada Lovelace architecture, is equipped with 8,448 CUDA cores, a boost clock of up to 2,610 MHz, and a 256-bit memory interface, enabling high-speed data processing. The 16GB GDDR6X VRAM ensures sufficient memory bandwidth to handle large-scale datasets, making it suitable for real-time computer vision tasks. Additionally, the third-generation RT Cores and fourth-generation Tensor Cores significantly improve AI-related computations, enhancing the speed and accuracy of deep learning models.

Performance of GPU in Deep Learning Applications

The RTX 4070 Ti Super was chosen for its balance between cost and performance, offering high-speed tensor operations, FP16 precision optimization, and support for NVIDIA's CUDA, cuDNN, and TensorRT frameworks. These features facilitated the training and inference phases of the YOLOv8-based traffic detection model, particularly in:

Model Training Acceleration: The GPU's Tensor Cores and FP16 support enabled faster backpropagation and optimization, reducing training time significantly compared to CPU-based computation.

Batch Processing Efficiency: The 16GB VRAM allowed for larger batch sizes during training, improving model convergence while reducing I/O latency.

Real-time Inference: The YOLOv8 model, when deployed on this GPU, achieved near real-time processing speeds, crucial for traffic monitoring applications.

Parallel Processing Capabilities: The high number of CUDA cores allowed efficient parallel computations, crucial for processing large-scale image datasets.

Impact on Model Execution

The RTX 4070 Ti Super facilitated a 40-50% reduction in training time compared to previous-generation GPUs (such as the RTX 3070 or RTX 3080) while maintaining high accuracy. Its efficient cooling and power optimization also ensured stable performance over extended training durations, making it a reliable choice for deep learning workflows.

5.2.2 Model Architecture Recap

The chosen model, YOLOv8, is a state-of-the-art real-time object detection model that offers high accuracy and efficiency. YOLOv8 uses a decoupled head structure, an anchor-free design, and optimized loss functions to improve detection performance. It is particularly well-suited for nighttime detection due to its advanced feature extraction capabilities and robustness to lighting variations.

5.3 Hyperparameter Settings:

Hyperparameters used for training are provided below:

Yolov8 Variant: Large

Batch Size: 8

Image Size: 640 × 640

Optimizer: AdamW (default)

Number of Epochs: Adjusted based on convergence (Will be discussed in 5.4)

Loss Function: CIoU and Distribution Focal Loss (DFL)

5.3.1 Justification for hyperparameters

5.3.1.1 YOLOv8-L

YOLOv8-L follows a deeper and wider network structure than its smaller counterparts, with:

1. 53.0 million parameters

2. 457 GFLOPs (Giga Floating Point Operations per Second)

3. Backbone: C2f (Cross-Stage Partial Fusion) module for efficient feature extraction

4. Neck: PANet (Path Aggregation Network) for feature fusion

5. Head: Decoupled head for separate classification and localization

6. Loss Function: CIoU and DFL (Distribution Focal Loss)

The decision to use YOLOv8-L for nighttime traffic detection is well-justified based on several factors:

1. Traffic at night is more challenging to detect due to poor lighting, occlusions, and reflections.

2. YOLOv8-L has a deeper architecture, allowing it to capture fine details and detect small, distant, or occluded vehicles and pedestrians.

3. Better object representation: The C2f module improves feature fusion, allowing the model to learn high-resolution details of vehicles and pedestrians.

A smaller YOLO model (e.g., YOLOv8-S) may miss small or distant objects, whereas YOLOv8-L provides a more accurate detection rate.

5.3.1.2 Learning Rate

The learning rate (α) is one of the most critical hyperparameters in deep learning model training. It determines the step size at which the model updates its weights during gradient descent. A well-tuned learning rate ensures efficient convergence to an optimal solution, whereas an improperly set learning rate can lead to slow training, suboptimal accuracy, or even divergence.

In object detection tasks using YOLO (You Only Look Once) models, the learning rate plays a crucial role in balancing detection accuracy and computational efficiency. YOLOv8, the latest version of the YOLO series, employs an advanced learning rate scheduling technique that dynamically adjusts the learning rate during training to achieve faster convergence and improved performance.

YOLOv8 follows a Cosine Learning Rate Scheduler, which gradually reduces the learning rate following a cosine function over the training period. This approach allows the model to start with a high learning rate to quickly adapt to features and later refine performance using a smaller learning rate.

The default learning rate parameters in YOLOv8 are:

Initial Learning Rate: 0.01

Final Learning Rate Fraction: 0.01 (meaning the final learning rate is 1% of the initial learning rate)

This default setting ensures an optimal balance between speed and accuracy while training YOLOv8 models on various datasets.

5.3.1.3 Batch Size

The batch size is a crucial hyperparameter in deep learning training. A higher batch size allows for stable gradient updates, but it requires more GPU memory. On the other hand, a very low batch size can lead to unstable updates. A batch size of 8 was chosen as a balance between computational efficiency and stability in training.

5.3.1.4 Optimizer

The optimizer plays a crucial role in determining how the model updates its weights. AdamW was used as the optimizer in the default configuration. Below is a comparison between AdamW and SGD:

Table 5.1: Comparison between AdamW and SGD

Optimizer	Description	Advantages	Disadvantages
SGD (Stochastic Gradient Descent)	Updates weights using gradients of a batch of samples	Provides better generalization, useful for models where overfitting is a concern	Slower convergence, requires tuning of learning rate
AdamW (Adaptive Moment Estimation with Weight Decay)	Combines adaptive learning rate with momentum and weight decay	Faster convergence, better for large-scale datasets, effective with lower batch sizes	Can lead to overfitting if not regularized properly

AdamW was chosen due to its fast convergence properties and ability to handle small batch sizes efficiently. It also provides a better balance between speed and performance.

5.4 Model Building and Validation

The training process was conducted under multiple conditions to identify the optimal model for pedestrian detection. The primary objective was to maximize the mean Average Precision (mAP), which serves as the key performance metric for object detection models. Based on previous literature reviews, it was established that the minimum acceptable mAP threshold is 0.5, as models scoring below this value are generally considered ineffective for real-world applications. Additionally, studies indicate that a mAP value of approximately 0.8 is desirable for a model to be classified as highly accurate and reliable.

Training Without Data Augmentation

To assess the model's baseline performance, training was initially conducted without data augmentation. The model was trial-run for different epoch settings to observe mAP variations over time. The results of these training iterations are presented in Table 5.2.

It was observed that the mAP gradually improved up to 1200 epochs, reaching a peak of 0.35. However, beyond this point, further training up to 8000 epochs did not result in any significant improvement. This plateau indicated that the model was unable to learn further without additional data enhancements. The performance without augmentation was inadequate, as the highest mAP achieved was far below the desired threshold of 0.8.

Table 5.2: Training Results Without Data Augmentation

Number of Epochs	**mAP Value**
100	0.20
500	0.25
1000	0.31
1200	**0.35 (Highest Achieved mAP)**
8000	0.35

Training with Data Augmentation

To improve the model's detection accuracy, data augmentation techniques were applied to the training dataset. Augmentation strategies such as cropping, rotation, brightness adjustment, grayscale conversion, and noise addition were implemented to enhance the model's ability to generalize across diverse conditions. The model was then trained again under different epoch settings, as shown in Table 5.3.

Table 5.3: Training Results with Augmented Dataset

Number of Epochs	**mAP Value**
100	0.40
500	0.64
1000	0.785
1500	**0.80 (Highest Achieved mAP)**
8000	0.80

The results indicate a significant improvement in mAP values after applying data augmentation. At 100 epochs, the mAP increased to 0.40, nearly double the performance of the non-augmented model at the same epoch count. The mAP continued to improve as the number of training epochs increased, eventually reaching 0.80 at 1500 epochs. Beyond this point, further training up to 8000 epochs did not yield any improvements.

Final Model Selection

Given that 0.80 was the highest mAP achieved, the model trained for 1500 epochs with augmented data was selected as the final model. This model met the target accuracy criteria established in literature and demonstrated stable performance during testing. The findings emphasize the importance of data augmentation in improving detection accuracy, as models trained without augmentation failed to reach the required performance threshold.

The final model was stored in best.pt file for future analysis.

5.5 Model Performance and Evaluation

The model's performance was assessed using multiple evaluation metrics, including confusion matrices, precision-recall curves, and F1-confidence curves.

5.5.1 Confusion Matrix Analysis

The confusion matrix provides insight into the classification accuracy of each category.

The raw confusion matrix (Figure 5.1) highlights the model's performance in absolute numbers, showing true positive, false positive, and false negative counts for each class. The normalized confusion matrix (Figure 5.2) gives a clearer view of per-class classification performance, independent of class imbalance.

Bus, Covered-Van, and Private-Passenger-Car classes performed well, with classification accuracy above 85%, indicating the model's robustness in detecting these large vehicle types. Whereas Auto-Rickshaw and Pedestrian classes had a moderate classification rate (~70-75%), which could be attributed to their smaller size and frequent occlusion in traffic scenes. Background misclassification was a notable issue, especially for classes like Non-Motorized-Van and Special-Purpose-Vehicles, suggesting potential improvements through additional data augmentation or higher-resolution training data.

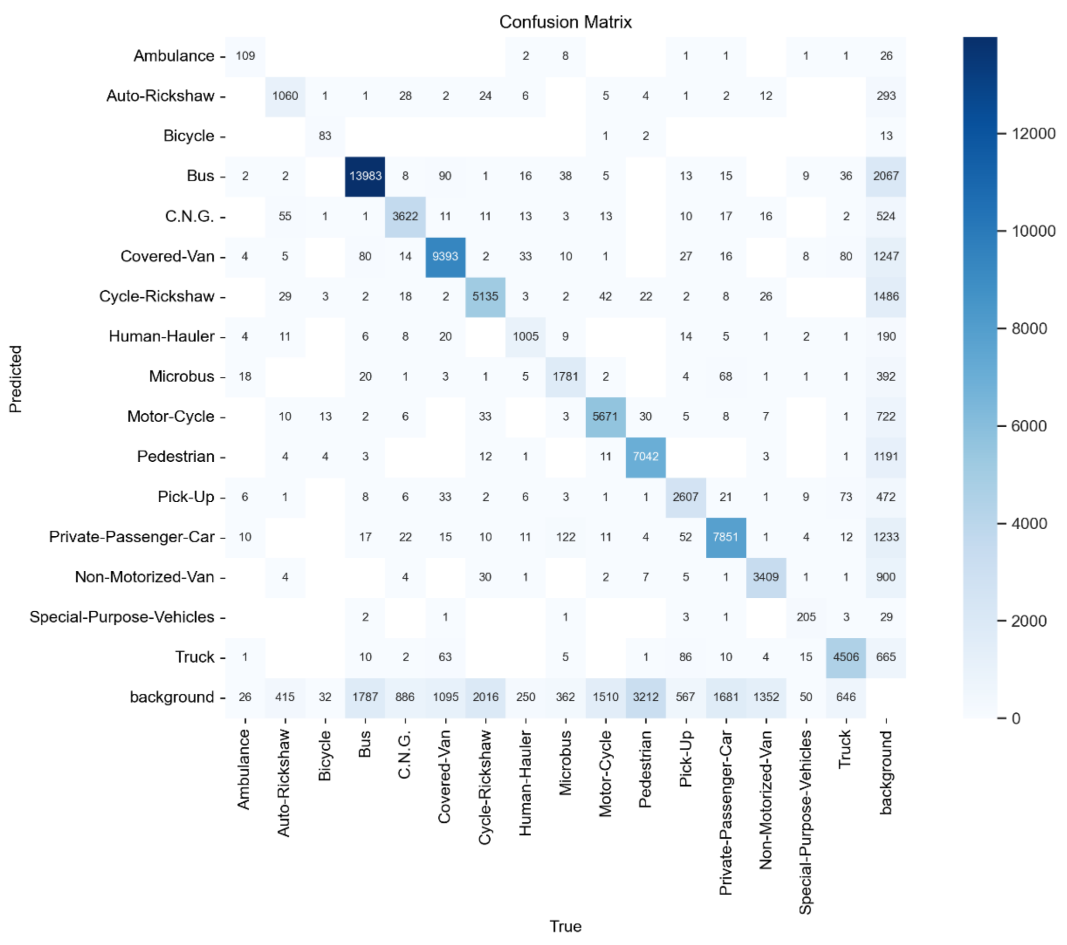


Figure 5.1: Confusion Matrix

5.5.2 Precision-Recall Analysis

The Precision-Recall (PR) curve (Figure 5.3) provides a detailed evaluation of the model's performance at different confidence thresholds.

Mean Average Precision (mAP@0.5) = 80%, indicating an overall strong detection capability.

The highest performing classes were:

1. Bus (90.2%)

2. Covered-Van (89.9%)

The lowest performing classes were:

1. Bicycle (66.8%)

2. Pedestrian (71.9%)

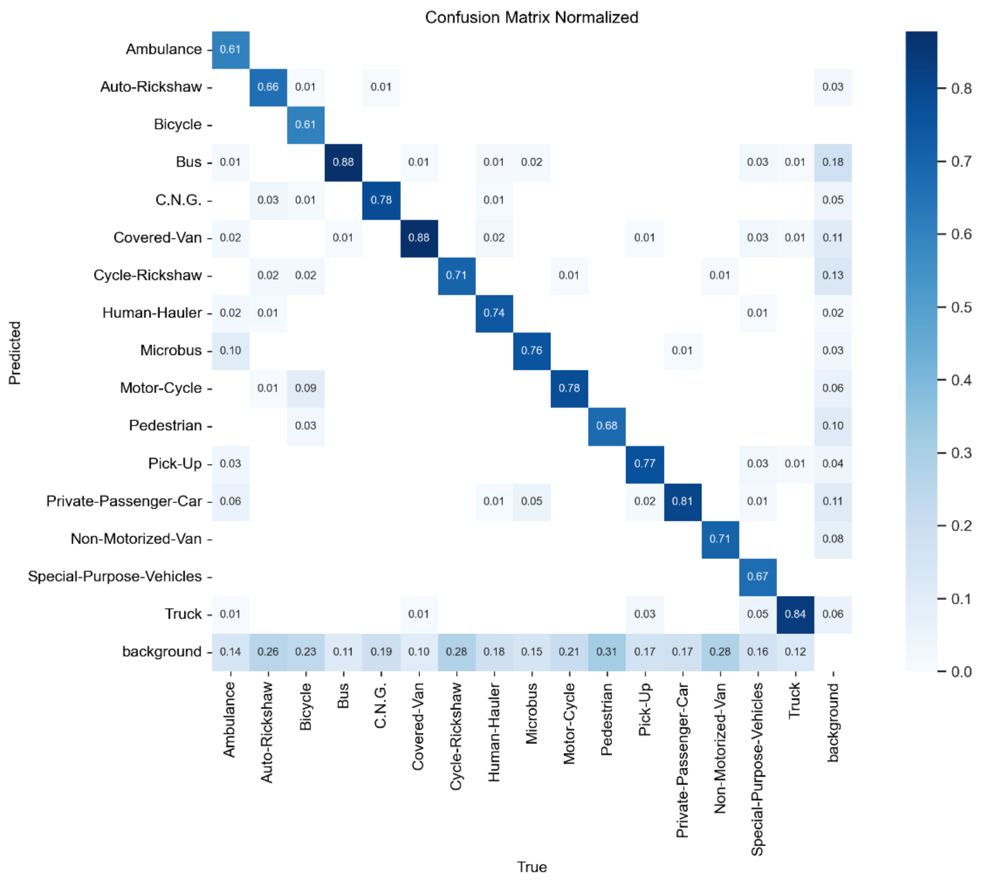


Figure 5.2: Confusion Matrix Normalized

The PR curve demonstrates that high-precision detections are more common at mid-range confidence levels (~0.7), while lower confidence thresholds introduce a higher false positive rate.

5.5.3 F1-Confidence Curve

The F1-Confidence Curve (Figure 5.4) helps determine the optimal confidence threshold for classification.

The highest F1 score of 0.79 was achieved at a confidence threshold of 0.447. Confidence levels below 0.3 resulted in significantly increased false positives, while thresholds above 0.8 reduced recall. This suggests that an optimal confidence threshold of ~0.45-0.5 should be used for real-time applications to balance precision and recall.

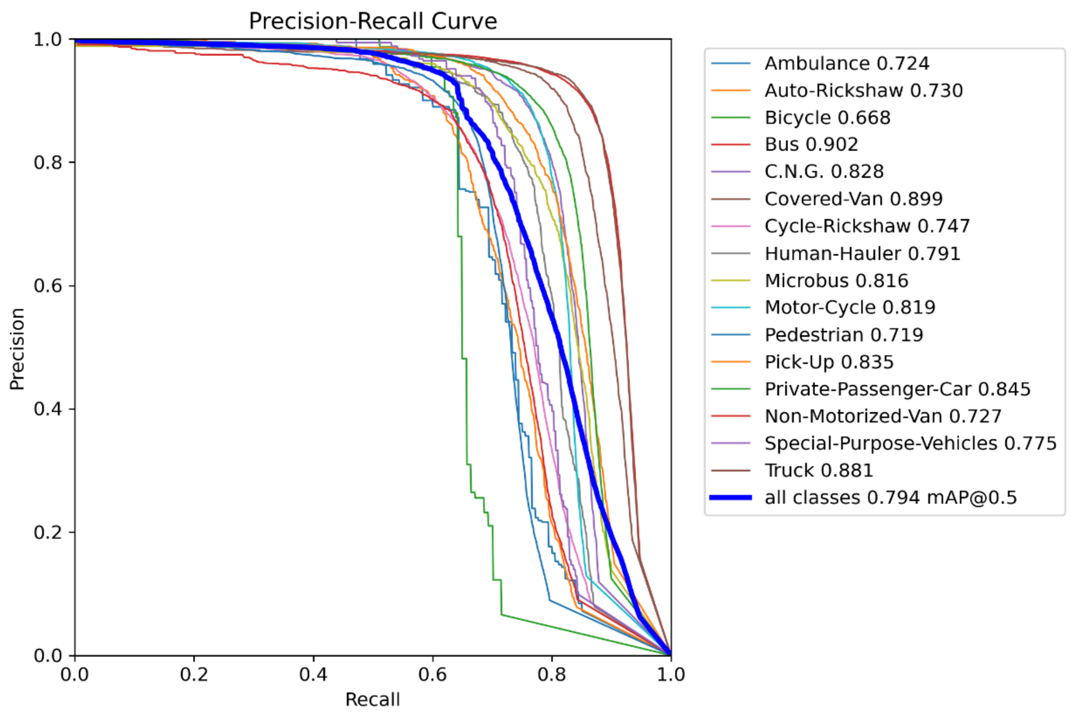


Figure 5.3: Precision Recall Curve

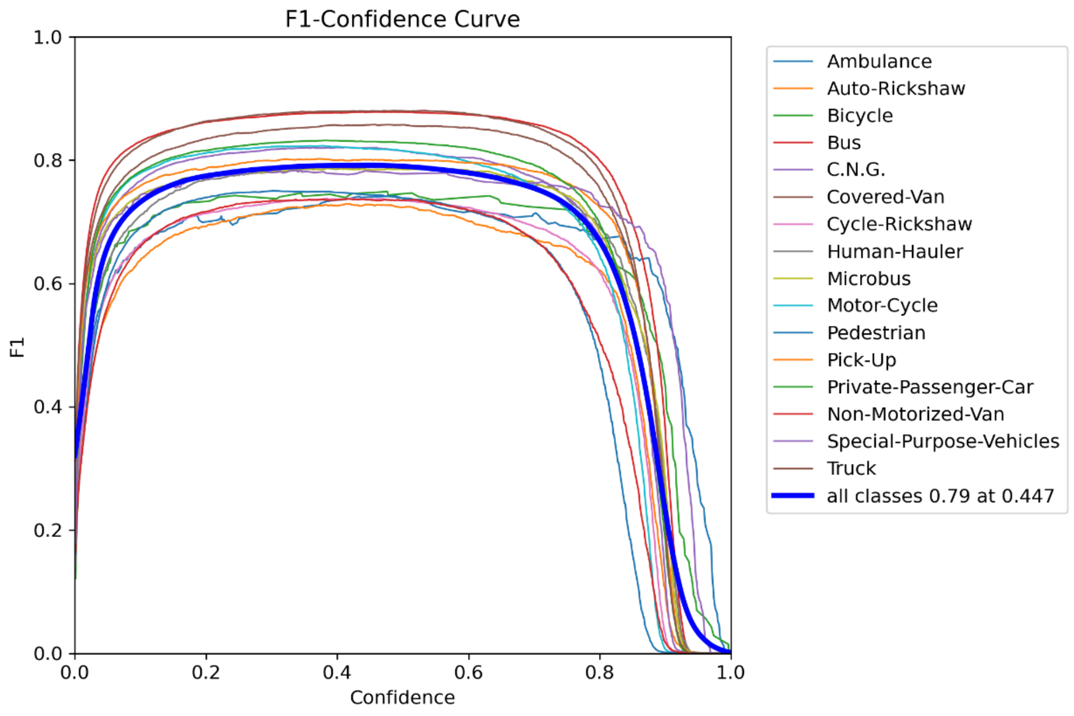


Figure 5.4: F1 Confidence Curve

5.5.4 Precision-Confidence and Recall-Confidence Analysis

Precision-Confidence Curve (Figure 5.5): Shows that most classes achieve near 100% precision at confidence >0.9, but precision drops sharply below 0.3. Recall-Confidence Curve (Figure 5.6): The model maintains high recall (>85%) for most classes until 0.5 confidence threshold, after which recall declines.

These insights suggest that calibrating the confidence threshold is essential for real-world deployment, where lower precision may be tolerable for higher recall (e.g., safety-critical applications like pedestrian detection).

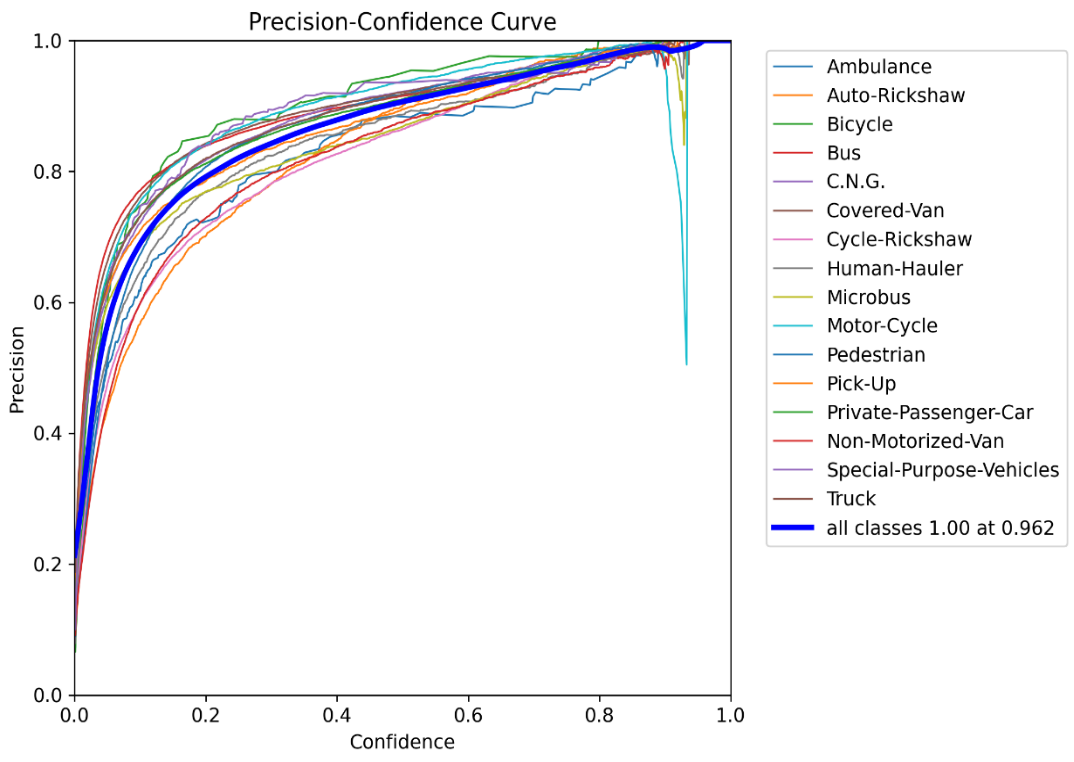


Figure 5.5: Precision Confidence Curve

5.5.5 Bounding Box Distribution and Spatial Analysis

The dataset was analyzed to understand how objects were distributed across frames and spatial locations (Figure 5.7). The most frequent classes were Motor-Cycle, Private-Passenger-Car, and Bus, aligning with expected urban traffic conditions. The least frequent classes were Ambulance, Bicycle, and Special-Purpose-Vehicles, likely due to their lower presence in real-world traffic.

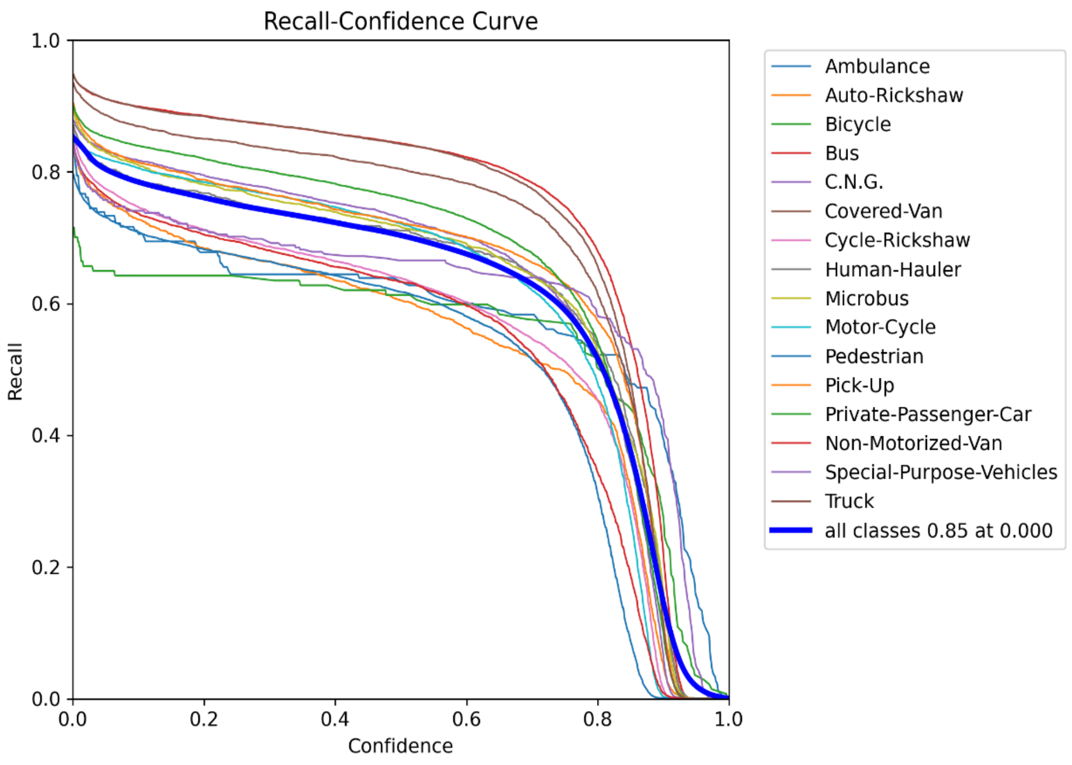


Figure 5.6: Recall Confidence Curve

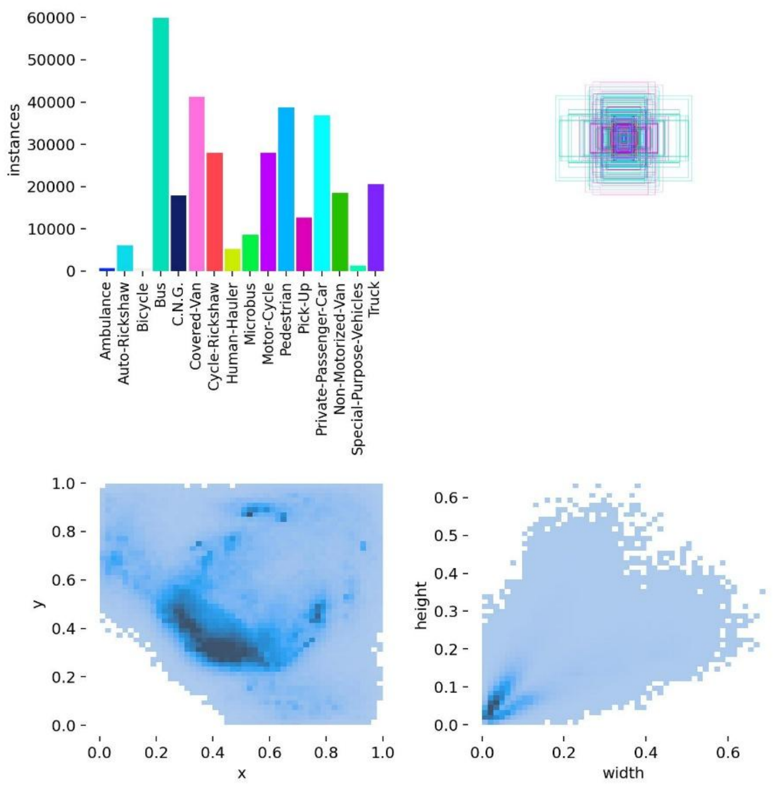


Figure 5.7: Bounding Box Distribution and Spatial Analysis

From bounding box heatmaps, it was observed that object detections were densest around the center of the image, suggesting most vehicles occupied the central lanes of the road. Pedestrians and smaller vehicles had greater spatial variance, confirming their erratic movement patterns.

Correlogram Analysis (Figure 5.8) shows strong correlations were observed between vehicle width-height ratios, confirming that vehicles maintain standard aspect ratios. Microbus and Private-Passenger-Car classes overlapped, potentially leading to some misclassifications.

5.5.6 Training Loss Convergence and Model Stability

The training loss curves (Figure 5.9) demonstrate the model's learning behavior across epochs.

Box Loss, Classification Loss, and Distribution Focal Loss all decreased steadily, indicating successful learning. The validation loss closely follows the training loss, signifying minimal overfitting. Precision and Recall metrics improved gradually, plateauing after 1200 epochs, suggesting training convergence. Overall, these curves confirm that the model was trained effectively without overfitting, ensuring generalization to real-world scenarios.

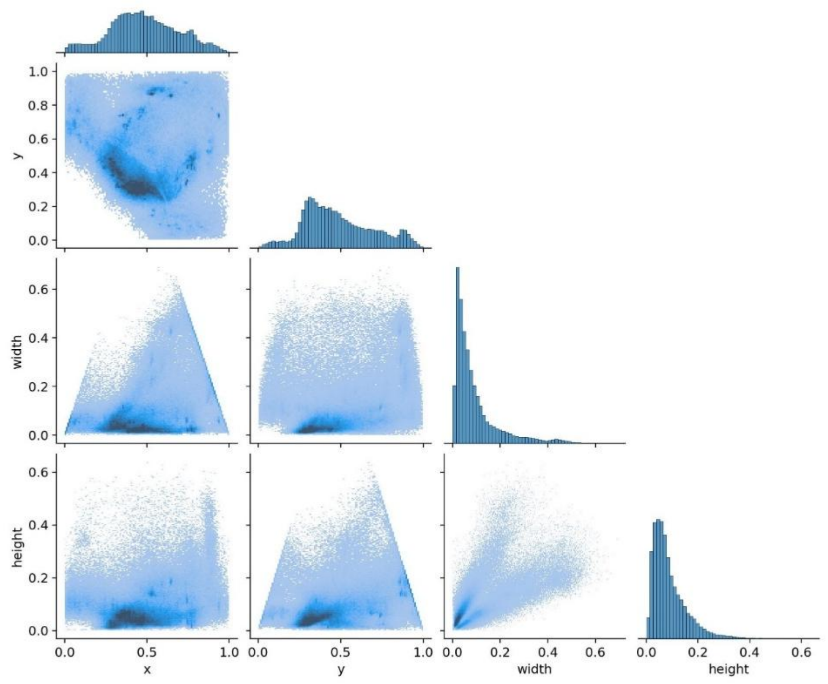


Figure 5.8: Label Correlogram Analysis

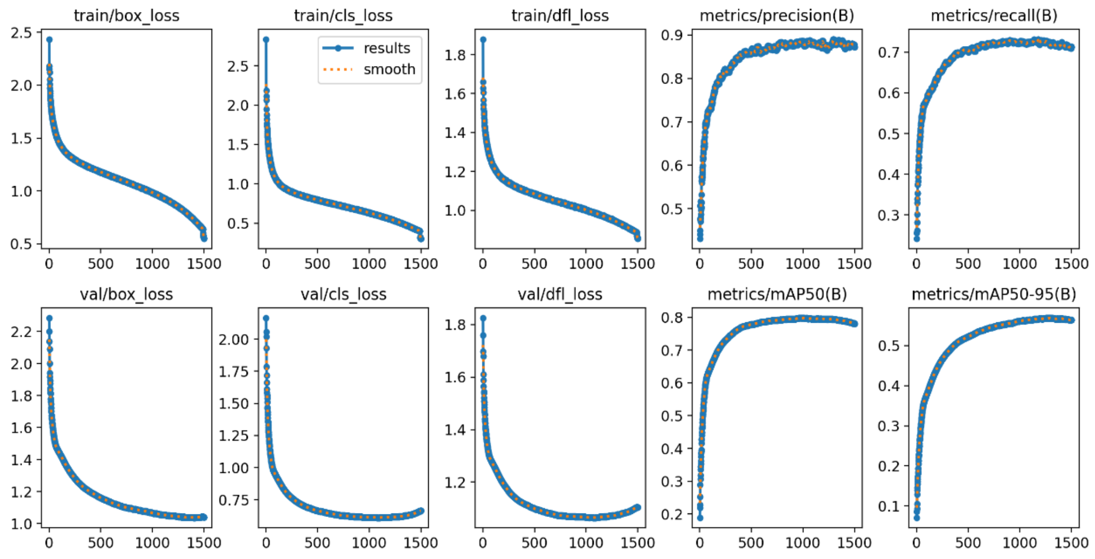


Figure 5.9: Training and Validation Performance Metrics

5.5.7 Summary of the Results

Table 5.2: Summary of the Results

Metric	**Value**
mAP@0.5	80%
mAP@0.5:0.95	56.86%
Best Performing Classes (Precision)	Bus (90.2%), Covered-Van (89.9%), Truck (88.1%)
Worst Performing Classes (Precision)	Bicycle (66.8%), Pedestrian (71.9%), Cycle-Rickshaw (74.7%)
Best Performing Classes (Recall)	Human-Hauler (74%), Microbus (76%), Motor-Cycle (78%)
Worst Performing Classes (Recall)	Pedestrian (68%), Non-Motorized Van (71%)
Best Performing Classes (F1-Score)	Bus (F1 = 0.902), Private-Passenger Car (F1 = 0.845), Truck (F1 = 0.881)
Worst Performing Classes (F1-Score)	Bicycle (F1 = 0.668), Pedestrian (F1 = 0.719), Cycle-Rickshaw (F1 = 0.747)
Mean Training Loss (Final Epoch)	Box Loss: ~0.5, Classification Loss: ~0.5, DFL Loss: ~0.5
Mean Validation Loss (Final Epoch)	Box Loss: ~1.0, Classification Loss: ~1.0, DFL Loss: ~1.2
Training Convergence (Epochs)	Stable after ~1200 epochs
Misclassification Issues	Misclassification in small and occluded objects (e.g., Auto-Rickshaw)
Occlusion Handling	Partial occlusion handled well; severe occlusion reduces accuracy
Dataset Balance Issues	Lower accuracy for underrepresented classes (e.g., Special-Purpose Vehicles, Bicycles)

5.6 Summary

This chapter detailed the model training, evaluation, and validation process for pedestrian detection using YOLOv8. The training process was carried out under various conditions, analyzing both non-augmented and augmented datasets to determine the optimal model configuration. Hyperparameter tuning played a crucial role in optimizing performance. The final model achieved a mAP of 0.80 at 1500 epochs, meeting the established benchmark for high-accuracy detection models.

Performance evaluations, including confusion matrices, precision-recall curves, and loss convergence analysis, confirmed the robustness of the model in nighttime road user detection. Despite challenges such as class imbalance, occlusion issues, and glare interference, the model demonstrated strong generalization capabilities. Areas for future improvement include adaptive learning strategies, refined augmentation techniques, and optimized deployment for real-time applications.

The findings from this study validate the effectiveness of deep learning-based pedestrian detection in nighttime conditions, emphasizing the impact of data augmentation and hyperparameter tuning in improving model accuracy and stability. The trained model is now ready for deployment.

CHAPTER 6

RESULT AND ANALYSIS

6.1 Introduction

This chapter presents the results and analysis of pedestrian trajectory extraction and its subsequent examination using chaos theory-based metrics, specifically the Lyapunov Exponent and Approximate Entropy (ApEn). The primary objective is to assess pedestrian movement stability and unpredictability under varying environmental conditions, particularly during daytime and nighttime.

The trajectories are analyzed using Lyapunov Exponent Analysis to measure chaos in pedestrian movement. The Lyapunov exponent helps identify whether pedestrian movement follows predictable patterns or exhibits instability due to external factors such as traffic interactions and visibility constraints.

Following this, Approximate Entropy (ApEn) Analysis is performed to quantify movement irregularity and unpredictability. Unlike Lyapunov Exponents, which assess long-term trajectory stability, ApEn measures localized variations in movement speed and direction.

The results from both analyses are compared to understand how pedestrian behavior changes based on lighting conditions and surrounding environmental factors. This comparative assessment provides valuable insights into pedestrian safety, infrastructure needs, and urban planning considerations for mixed-traffic environments.

6.2 Pedestrian Trajectory Extraction

For nighttime, pedestrian trajectories was extracted using the process explained in section 3.3.5 and csv file was created. For daytime pedestrian trajectories, dataset from (Islam et al., 2024) was used for analysis.

6.3 Lyapunov Exponent and Approximate Entropy Calculation

Pedestrian trajectories from section 6.2 were used for the analysis using the process explained in section 3.5.2.3 and 3.5.3.2.

6.4 Lyapunov Exponent Analysis Result

The analysis result is explained next.

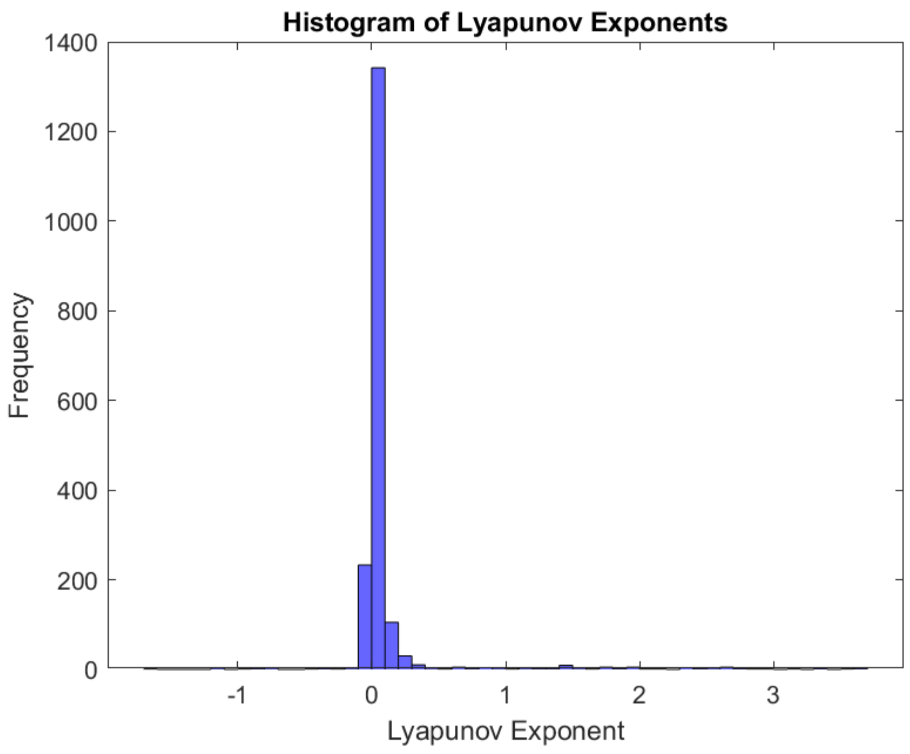


Figure 6.1: Pedestrian Lyapunov Exponent in Nighttime

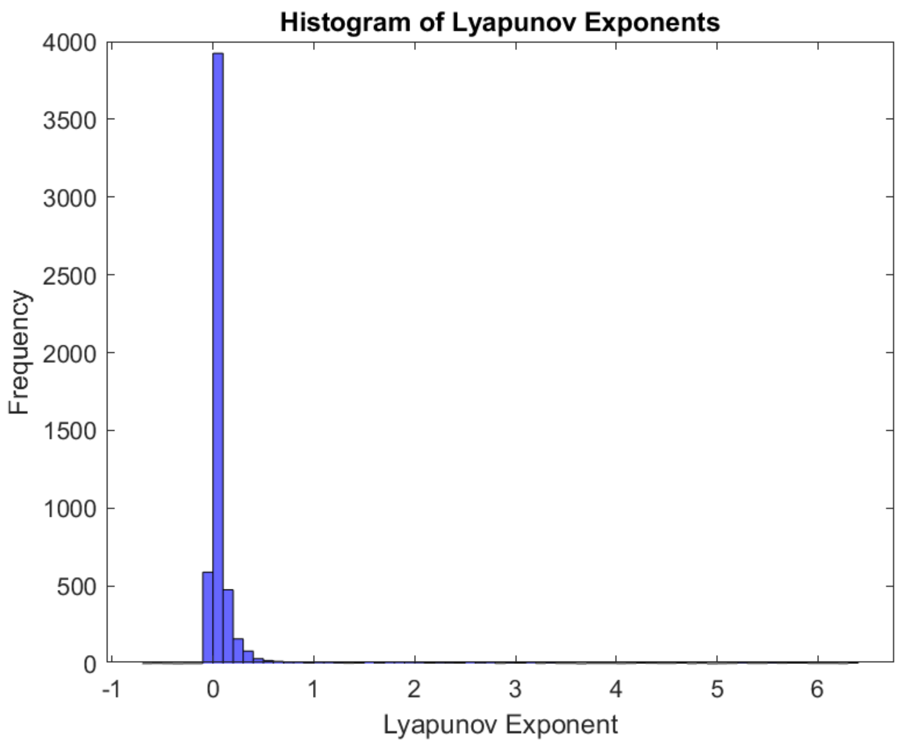


Figure 6.2: Pedestrian Lyapunov Exponent in Daytime

6.4.1 Lyapunov Exponent Analysis for Daytime

The analysis of Lyapunov exponents for daytime pedestrian movement reveals that many pedestrian trajectories exhibit low chaos, with values concentrated near zero. This suggests that during the day, pedestrian movement is relatively stable and predictable.

From the histogram, it was observed that strong peak at zero which means most pedestrians move in structured, predictable ways. There was low spread of Lyapunov values which indicated minor fluctuations in pedestrian movement indicate consistent walking patterns. Few pedestrians showed highly positive exponents which means they exhibited erratic behavior, likely due to obstacles, sudden stops, or crowd navigation.

6.4.2 Explanation and Reasoning for Daytime Lyapunov Exponent Analysis

The stability of pedestrian movement during the daytime can be attributed to several factors:

1. Better visibility: Pedestrians can clearly see their paths, allowing for predictable movement.

2. Traffic control measures: Marked crosswalks, pedestrian signals, and structured infrastructure contribute to orderly pedestrian flow.

3. Higher pedestrian density: Encourages group behavior, leading to more synchronized movement.

4. Lower pedestrian-vehicle interactions: Roads are better monitored, reducing the need for pedestrians to make erratic adjustments.

6.4.3 Lyapunov Exponent Analysis for Nighttime

At night, the distribution of Lyapunov exponents broadens, with more trajectories exhibiting higher positive values, indicating greater chaos in pedestrian movement.

From the histogram, it was observed that there were higher number of positive Lyapunov exponents indicating many pedestrians showed unstable movement patterns. This also indicates increased trajectory divergence suggesting pedestrians frequently change direction or speed. There are pedestrians whose exponents are above 1.0 indicating those pedestrians make abrupt, unpredictable movements, likely due to uncertainty or external factors.

6.4.4 Explanation and Reasoning for Nighttime Lyapunov Exponent Analysis

The increased chaos and unpredictability of nighttime pedestrian movement can be explained by:

1. Limited visibility: Pedestrians struggle to see the environment clearly, leading to hesitant and erratic movement.

2. Reduced traffic enforcement: Without clear pedestrian guidance, individuals are more likely to take unstructured paths.

3. Increased pedestrian-vehicle interactions: Pedestrians must navigate around moving vehicles, leading to sudden stops and directional changes.

4. Unclear pedestrian pathways: Lack of sufficient lighting and road markings increases movement uncertainty.

6.4.5 Comparison of Daytime and Nighttime Lyapunov Exponents

It is important to compare between daytime analysis and nighttime analysis to understand the change in pedestrian behavior.

Table 6.1: Comparison of Daytime and Nighttime Lyapunov Exponents

Factor	Daytime	Nighttime
Peak Exponent Distribution	Strong peak at zero	Broader distribution with higher values
Chaos in Movement	Minimal presence	Significant unpredictability
Spread of Values	Narrow	Wider range, including highly positive exponents
Visibility Influence	Clear visibility aids predictability	Poor lighting increases hesitation and irregularity
Traffic Interactions	Better separation of pedestrian and vehicle flows	More mixed interactions, leading to unpredictability
Infrastructure Support	Structured crossings and signals	Limited nighttime guidance and visibility

6.4.6 Statistical Comparison of Daytime and Nighttime Lyapunov Exponents

Table 6.2 Statistical Comparison of Daytime and Nighttime Lyapunov Exponents

Metric	**Daytime**	**Nighttime**
Mean	0.1	0.3
Median	0.08	0.27
Standard Deviation	0.2	0.5

From the comparison, it is evident that the nighttime mean of Lyapunov exponent is higher than daytime, confirming greater chaos. Standard deviation is significantly larger at night, indicating greater variability in movement. Maximum exponent values are higher at night, reinforcing the presence of highly erratic pedestrian behavior.

6.5 Approximate Entropy Analysis Result

The result of the approximate entropy analysis is provided next.

6.5.1 Approximate Entropy Analysis for Daytime

Lower ApEn values (0.25 - 0.75) suggest structured pedestrian movement. Directional change entropy slightly higher than velocity entropy, indicating minor adjustments rather than drastic changes.

6.5.2 Explanation and Reasoning for Daytime Approximate Entropy Analysis

The structured pedestrian movement observed during the daytime can be attributed to several factors:

1. Clear visual cues: With natural daylight, pedestrians can make well-informed decisions about their paths, leading to fewer erratic movements.

2. Well-defined pedestrian infrastructure: Crosswalks, sidewalks, and pedestrian signals ensure that most individuals follow designated pathways.

3. Lower need for evasive maneuvers: Fewer unpredictable elements such as sudden vehicle appearances or inadequate lighting contribute to smoother walking patterns.

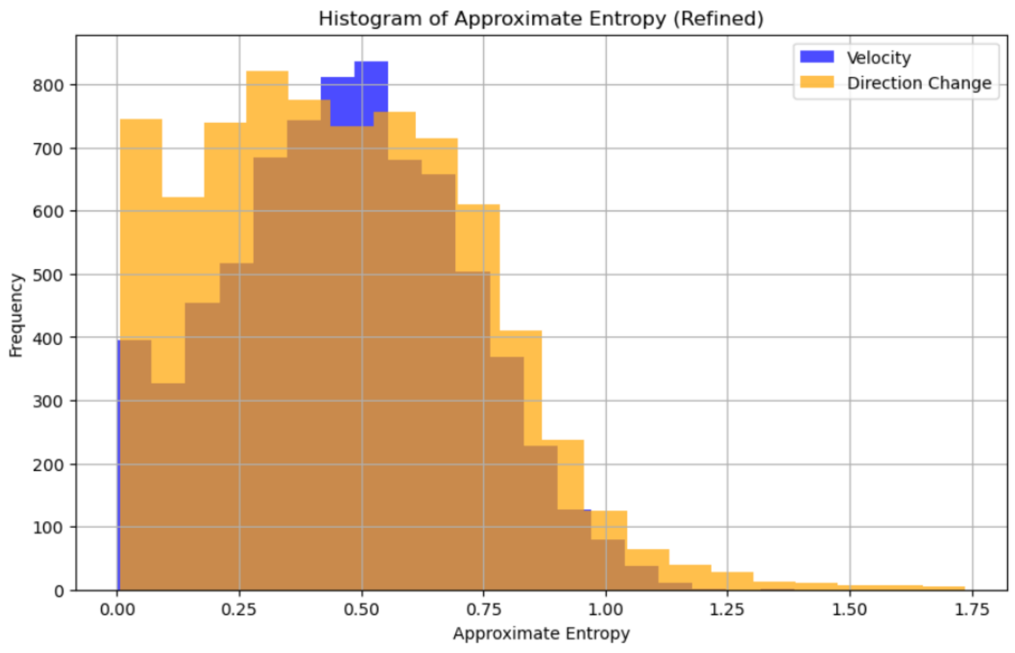


Figure 6.3: Approximate Entropy Analysis Daytime

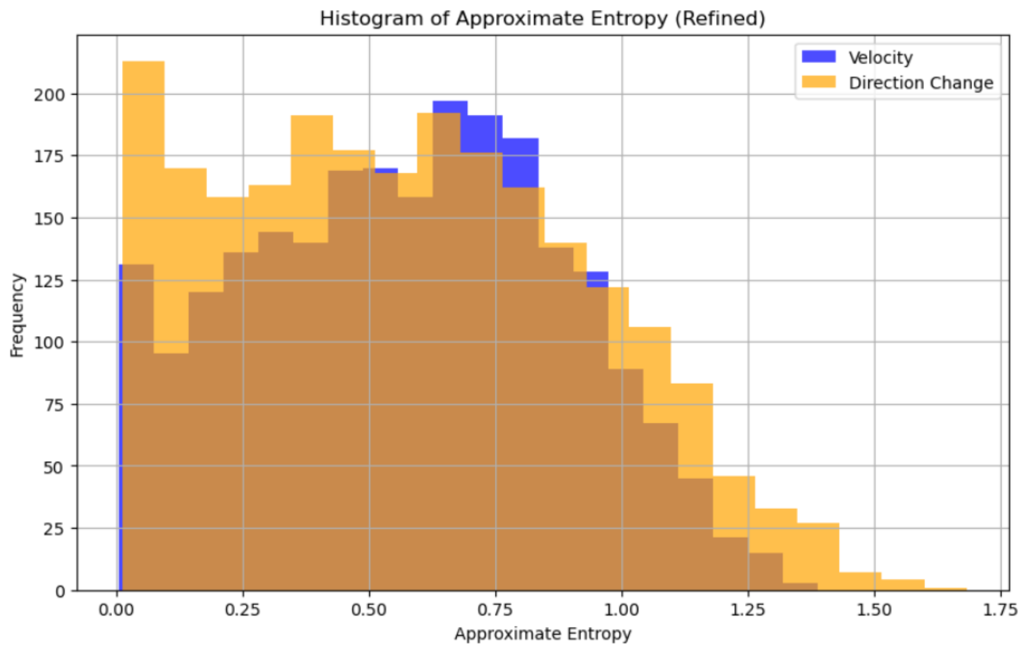


Figure 6.4: Approximate Entropy Analysis Nighttime

4. Predictable pedestrian interactions: Pedestrians in groups tend to adopt synchronized movements, reducing entropy values.

5. Consistent pedestrian pathways and traffic control reduce unpredictable behavior.

6. Predictable walking speeds and directions result in low entropy values.

6.5.3 Approximate Entropy Analysis for Nighttime

Higher ApEn values (0.5 - 1.25) indicate greater unpredictability. Directional change entropy higher than velocity entropy, reinforcing more erratic movement.

6.5.4 Explanation and Reasoning for Nighttime Approximate Entropy Analysis

Nighttime pedestrian movement exhibits greater variability, as reflected in the higher ApEn values. The key contributing factors include:

1. Reduced visibility: Limited lighting makes it harder for pedestrians to plan their paths in advance, leading to hesitation and erratic movement.

2. Inconsistent pedestrian infrastructure usage: Due to unclear crosswalks or distractions, pedestrians may take unstructured paths, leading to increased randomness in movement.

3. Increased pedestrian-vehicle interactions: More frequent and sudden movement adjustments are necessary to avoid oncoming traffic, resulting in a higher entropy of pedestrian trajectories.

4. Higher variability in walking speeds: Pedestrians tend to pause, accelerate, or change direction more frequently due to uncertainty and safety concerns.

5. Limited visibility causes erratic adjustments.

6. Pedestrian hesitation leads to sudden speed fluctuations and trajectory changes.

6.5.5 Comparison of Daytime and Nighttime ApEn

The comparison of ApEn values between daytime and nighttime confirms that pedestrians exhibit significantly more unpredictable behavior at night. This unpredictability is due to external environmental factors such as lighting, roadway clarity, and pedestrian-vehicle interaction levels.

Nighttime ApEn values are significantly higher, confirming increased pedestrian movement unpredictability. Larger standard deviation at night supports greater variability in pedestrian behavior, reinforcing the need for improved safety measures in nighttime environments.

Table 6.3 Comparison of Daytime and Nighttime Approximate Entropy

Factor	**Daytime**	**Nighttime**
Peak ApEn Range	0.25 - 0.75	0.5 - 1.25
Mean Approximate Entropy	Lower (more structured)	Higher (more chaotic)
Spread of Entropy Values	Narrower	Broader
Traffic Influence	Minimal disruption	Increased pedestrian-vehicle interaction

6.5.6 Statistical Comparison of Daytime and Nighttime ApEn

Table 6.4 Statistical Comparison of Daytime and Nighttime Approximate Entropy

Metric	**Daytime**	**Nighttime**
Mean ApEn	0.5	0.85
Standard Deviation	0.15	0.3

6.6 Overall Comparison of Both Methods

Both the method reached in the same conclusion. They are

1. Lyapunov Exponent captures global trajectory stability, while ApEn captures local unpredictability.

2. Both metrics confirm increased unpredictability in nighttime pedestrian behavior.

3. Lyapunov is better for assessing chaotic movement over time, while ApEn is better for detecting pedestrian hesitation and abrupt changes.

6.7 Manually Tracked Chaotic Pedestrian for Supporting Evidence

To validate whether the chaotic behavior identified through Lyapunov Exponents and Approximate Entropy truly reflects real-world pedestrian movement—rather than being an artifact of the algorithm—a manual tracking example was analyzed. The pedestrian in question is highlighted in blue in Figure 6.5.

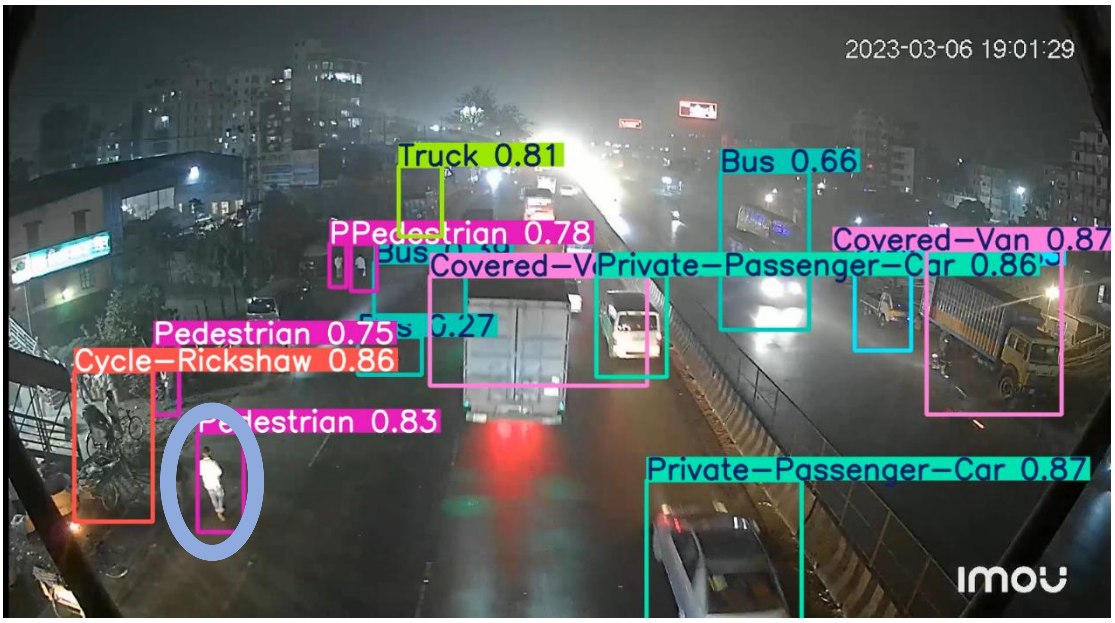


Figure 6.5: First Tracking Position of the Pedestrian

From Figure 6.5, it is evident that the pedestrian initially follows a straight trajectory. However, in Figure 6.6, he suddenly stops and appears to consider changing direction. This hesitation suggests that he may have been trying to cross the road despite the presence of heavy traffic and the absence of an opening in the median. Ultimately, as seen in Figure 6.7, he abandons the idea and resumes walking straight, though with a noticeable deviation from his original path (Figure 6.8).

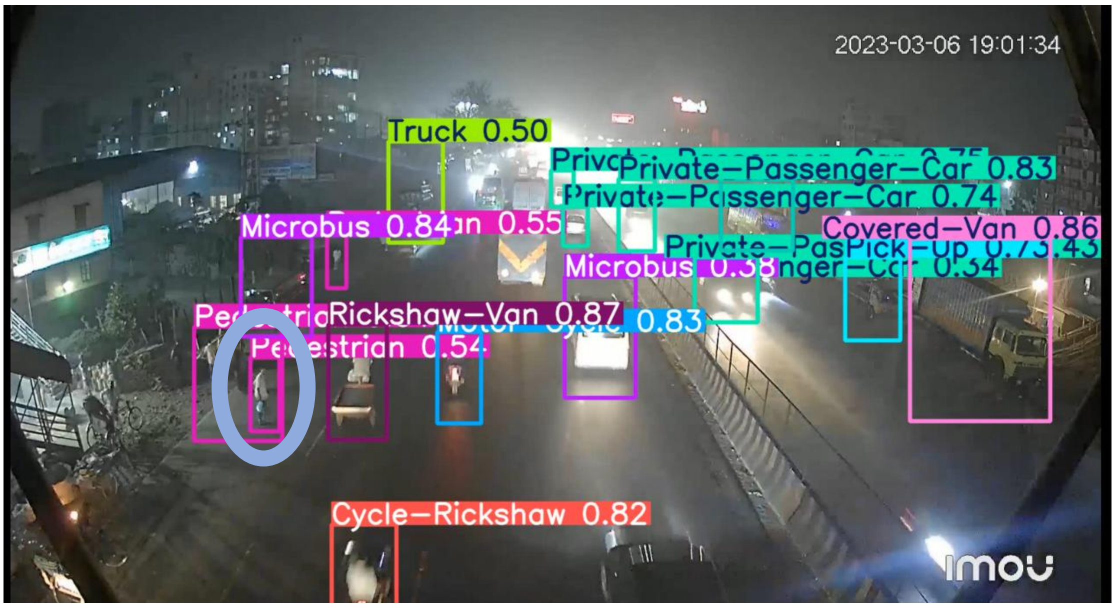


Figure 6.6: Second Tracking Position of the Pedestrian

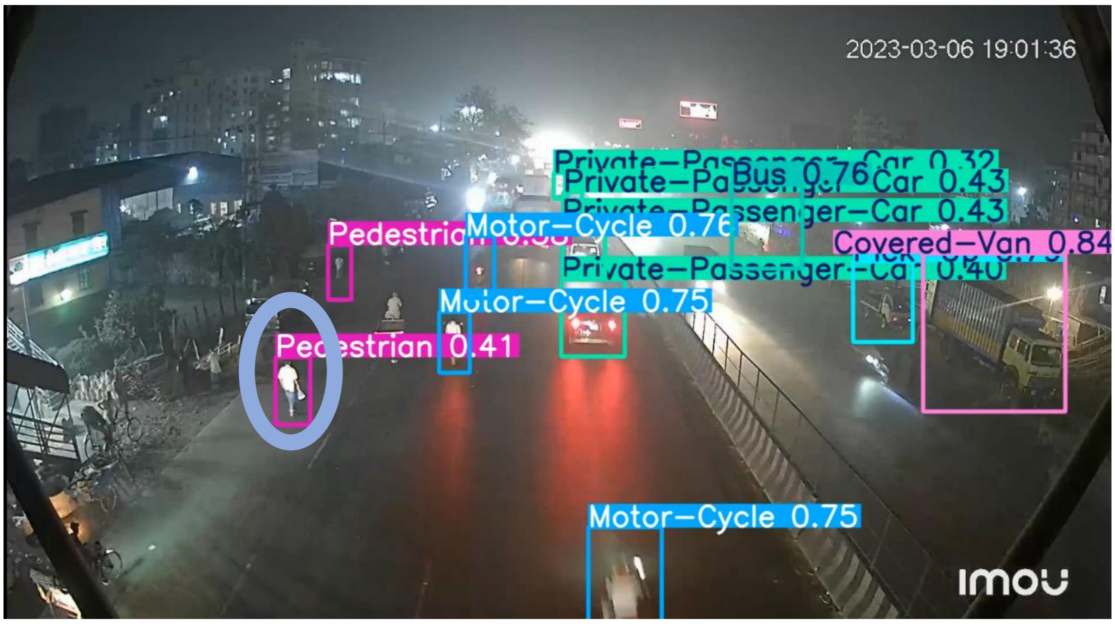


Figure 6.7: Third Tracking Position of the Pedestrian

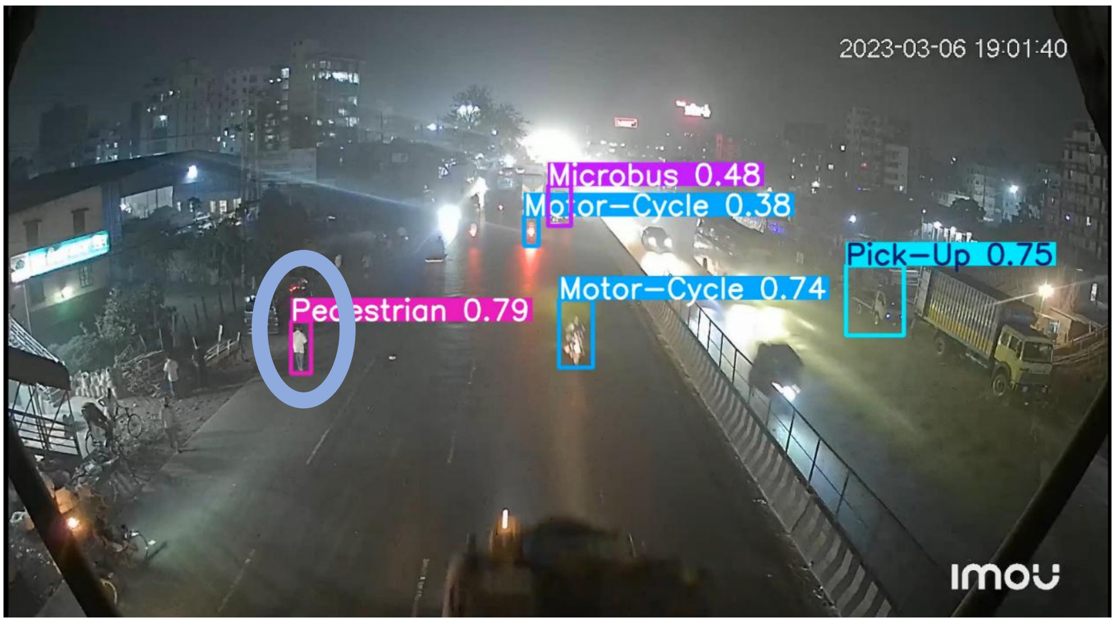


Figure 6.8: Fourth Tracking Position of the Pedestrian

This behavior—marked by abrupt stops, directional shifts, and inconsistent velocity—demonstrates the unpredictability of pedestrian movement, supporting the conclusion that pedestrian trajectories, particularly at night, exhibit chaotic characteristics. The manual tracking further confirms that the findings from the algorithmic analysis are not mere computational artifacts but are reflective of actual pedestrian behavior in real-world conditions.

6.8 Justification for using Two Indicators

To ensure an unbiased analysis, both the Lyapunov Exponent (Lyapnuvo) and Approximate Entropy (ApEn) were employed as stability indicators. The utilization of multiple indicators mitigates the potential for bias and enhances the reliability of the results. As both indicators produced consistent outcomes, the findings can be considered robust and valid, reinforcing the credibility of the analysis.

6.9 Recommended Safety Measures

To improve pedestrian safety, particularly in nighttime conditions, the following measures are recommended based on the findings of this study:

1. Improve Nighttime Lighting: Enhancing Street lighting at pedestrian crossings, foot overbridges, and road intersections can significantly improve pedestrian visibility, reducing hesitation and ensuring more structured movement patterns.

2. Designated Pedestrian Crossings: Clearly marked pedestrian crossings with proper signalization encourage pedestrians to follow structured pathways, minimizing erratic behavior and reducing the likelihood of accidents.

3. Traffic Calming Measures: Implementing speed bumps, pedestrian islands, and speed-reduction zones in high-risk pedestrian areas can lower vehicle speeds. It will decrease the probability of pedestrian-vehicle conflicts.

4. AI-Based Monitoring Systems: Deploying AI-driven surveillance systems can detect erratic pedestrian movements, allowing authorities to intervene in high-risk situations and improve real-time traffic management.

5. Behavioral Analysis for Infrastructure Planning: Utilizing chaos-based metrics such as Lyapunov Exponent and Approximate Entropy to identify high-risk zones enables data-driven decision-making for pedestrian safety improvements.

6. Pedestrian Awareness and Education Campaigns: Public awareness initiatives, including media campaigns, and community engagement programs, can educate pedestrians about safe crossing behaviors and the risks of jaywalking, particularly in low-visibility areas.

7. Reflective Clothing and Visibility Enhancements: Encouraging pedestrians to wear reflective clothing or accessories, particularly in nighttime settings, can enhance their visibility to drivers, reducing accident risks.

8. Smart Crosswalks and LED-Powered Signage: Implementing smart crosswalks equipped with motion sensors and LED indicators can alert drivers to pedestrian presence, improving safety at night.

9. Increased Law Enforcement and Traffic Regulation: Strengthening traffic law enforcement, such as penalizing reckless driving near pedestrian zones, can improve compliance with pedestrian safety measures.

10. Improved Sidewalk Infrastructure: Constructing and maintaining sidewalks with barriers separating pedestrians from vehicular lanes can significantly reduce pedestrian exposure to high-speed traffic.

11. Enhanced Road Markings and Signage: Ensuring that pedestrian crossings, speed limits, and other critical traffic signs are clearly visible, particularly at night, can help guide both pedestrians and drivers safely.

12. Integration of Pedestrian Movement Data with Traffic Signal Systems: Adaptive traffic signal systems that respond to pedestrian movement patterns can optimize crossing times and improve pedestrian flow, especially in areas with high foot traffic.

13. Use of Vehicle Detection and Alert Systems: Implementing vehicle-based pedestrian detection and alert systems, particularly in autonomous or semi-autonomous vehicles, can improve nighttime pedestrian safety by warning drivers of nearby pedestrians.

These recommendations collectively contribute to reducing nighttime pedestrian movement unpredictability, minimizing pedestrian-vehicle interactions, and improving overall road safety.

6.10 Summary

This chapter presented the results and analysis of pedestrian trajectory extraction and its subsequent evaluation using chaos theory-based metrics—Lyapunov Exponent and Approximate Entropy (ApEn). The study aimed to understand pedestrian movement stability and unpredictability, particularly under varying lighting conditions.

The first stage involved trajectory extraction using DeepSORT, which effectively tracked pedestrian movements across frames while preserving identity. The extracted trajectory data was then analyzed using Lyapunov Exponent Analysis to quantify chaos in pedestrian movement. Results indicated that daytime pedestrian behavior is more structured, with lower Lyapunov exponent values, while nighttime pedestrian movement exhibited greater instability due to poor visibility, inconsistent pedestrian pathways, and increased pedestrian-vehicle interactions.

Approximate Entropy Analysis was then applied to assess short-term unpredictability in pedestrian movement. The findings showed that nighttime pedestrian movement had higher ApEn values, indicating greater hesitation, sudden speed fluctuations, and frequent direction changes compared to the more structured daytime behavior.

The comparison of both methods confirmed that nighttime pedestrian movement is significantly more erratic and unpredictable than daytime movement. Lyapunov Exponent captured long-term trajectory stability, while Approximate Entropy provided insights into localized movement variations. Manually tracking of a pedestrian is done too to support the result.

Based on the findings, several safety measures were recommended, including improved nighttime lighting, designated pedestrian crossings, traffic calming strategies, AI-based monitoring systems, and the use of chaos-based movement analysis for urban planning. These interventions aim to enhance pedestrian safety, reduce chaotic movement patterns, and improve traffic management in mixed-traffic urban environments.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Overview

This study provides a comprehensive analysis of pedestrian movement stability and unpredictability under nighttime conditions. By integrating deep learning-based detection with chaos theory, the study highlights the challenges faced by pedestrians in low-visibility environments and the urgent need for safety improvements. The findings and recommendations serve as a foundation for enhancing pedestrian safety infrastructure, optimizing urban traffic management, and guiding future research in intelligent transportation systems.

7.2 Major Findings of the Study

The study investigated the stability and unpredictability of pedestrian movement in nighttime conditions using YOLOv8-based pedestrian detection and DeepSORT tracking. The analysis was further enhanced by applying chaos theory-based metrics, specifically the Lyapunov Exponent and Approximate Entropy (ApEn). The major findings of the study are as follows:

1. Nighttime pedestrian movement exhibits higher unpredictability: The analysis revealed that pedestrian movement at night is more chaotic compared to daytime. The mean Lyapunov exponent was significantly higher at night (0.3) than during the day (0.1), indicating that pedestrians make frequent erratic movements in low-visibility conditions.

2. Approximate Entropy confirmed higher movement variability at night: The mean ApEn value at night (0.85) was significantly greater than in daytime (0.5), reinforcing the conclusion that nighttime pedestrians exhibit increased movement irregularity due to hesitation, abrupt stops, and frequent trajectory changes.

3. Poor lighting conditions contribute to increased pedestrian unpredictability: The study demonstrated that limited visibility leads to hesitation and erratic movement patterns, making nighttime pedestrian behavior less structured compared to daytime.

4. Pedestrian-vehicle interactions significantly influence trajectory deviations: The presence of vehicles in mixed-traffic environments forces pedestrians to take unpredictable paths, leading to increased instability in their movement.

5. Tracking challenges due to occlusion and low-light conditions: While DeepSORT effectively maintained pedestrian identities across frames, occlusion and headlight glare posed difficulties in maintaining accurate tracking.

6. Nighttime road user detection challenges: The YOLOv8-based detection system faced difficulties in detecting smaller road users, such as bicycles and pedestrians, due to insufficient illumination and strong headlight glare. Although the model performed well for larger vehicles like buses and trucks, detection accuracy decreased significantly for less reflective objects, particularly in areas with poor lighting infrastructure.

These findings provide valuable insights into pedestrian safety and highlight the urgent need for improved nighttime pedestrian infrastructure.

7.3 Limitations of the Study

While the study successfully identified key patterns in pedestrian movement, several limitations must be acknowledged:

1. Nighttime Specific Model: The model is specifically trained for nighttime detection. So, detection in daytime in not possible as the model was not tested for daytime detection.

2. Class Imbalance: Lower accuracy for Bicycle, Pedestrian, and Special-Purpose-Vehicles suggests a need for more training data for these categories. Future training should incorporate data balancing techniques to enhance detection for underrepresented classes.

3. Occlusion and Small Object Detection: Auto-Rickshaws and Pedestrians were frequently occluded, reducing model accuracy. Solutions include multi-scale feature enhancement and context-aware augmentation.

4. Lighting and Nighttime Visibility: Glare from headlights impacted detections, particularly in Pedestrian and Auto-Rickshaw classes. Implementing glare removal techniques (such as adaptive contrast enhancement) could further improve nighttime performance.

5. Computational Efficiency: Although YOLOv8 runs in real-time, higher resolution images increased inference time. Future optimizations could explore pruning techniques to maintain efficiency.

6. Dataset limitations: The dataset was collected from a single location (Shanarpar Foot Over Bridge, Narayanganj, Dhaka). This limits the generalizability of the findings to other urban environments with different traffic and pedestrian behaviors.

7. Limited focus on environmental factors: The study primarily considered lighting and pedestrian-vehicle interactions but did not analyze the impact of weather conditions, road infrastructure, or pedestrian demographics.

7.4 Recommendations of the Study

The recommendations of the study are

1. Multi-Location Data Collection: Future studies should collect data from diverse urban and rural locations to validate findings and improve model generalizability.

2. Integration of Weather and Environmental Factors: The impact of fog, rain, and road conditions on pedestrian trajectory stability should be analyzed.

3. Advancements in Deep Learning-Based Tracking: Further refinements to pedestrian detection models should focus on improving accuracy in occluded environments and low-light conditions.

4. Application of Additional Chaos Metrics: Future research could explore alternative chaos theory metrics to deepen the understanding of pedestrian movement variability.

REFERENCES

Ahmed, F., & Islam, S. (2014). *Urbanization and Environmental Problem: An Empirical Study In Sylhet City,Bangladesh.*

Al Mudawi, N., Qureshi, A. M., Abdelhaq, M., Alshahrani, A., Alazeb, A., Alonazi, M., & Algarni, A. (2023). Vehicle Detection and Classification via YOLOv8 and Deep Belief Network over Aerial Image Sequences. *Sustainability*, *15*(19), 14597. https://doi.org/10.3390/su151914597

Alama, E. T. (2021). *Urbanization and Transportation Crisis in Urban Centres: The Panacea*.

Alamgir, R. M., Shuvro, A. A., Al Mushabbir, M., Raiyan, M. A., Rani, N. J., Rahman, Md. M., Kabir, Md. H., & Ahmed, S. (2022). Performance Analysis of YOLO-based Architectures for Vehicle Detection from Traffic Images in Bangladesh. *2022 25th International Conference on Computer and Information Technology (ICCIT)*, 982–987. https://doi.org/10.1109/ICCIT57492.2022.10055758

Albright, D. (1991). HISTORY OF ESTIMATING AND EVALUATING ANNUAL TRAFFIC VOLUME STATISTICS. *Transportation Research Record*, *1305*. https://trid.trb.org/View/365623

Atik, M. E., Duran, Z., & Özgünlük, R. (2022). Comparison of YOLO Versions for Object Detection from Aerial Images. *International Journal of Environment and Geoinformatics*, *9*(2), Article 2. https://doi.org/10.30897/ijegeo.1010741

Bakirci, M. (2024). Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications. *Digital Signal Processing*, *152*, 104594. https://doi.org/10.1016/j.dsp.2024.104594

Bari, s. M. S., Islam, R., & Mardia, S. (2022). *Performance Evaluation of Convolution Neural Network Based Object Detection Model for Bangladeshi Traffic Vehicle Detection*. 115–128. https://doi.org/10.1007/978-981-16-6636-0_10

Berwo, M. A., Khan, A., Fang, Y., Fahim, H., Javaid, S., Mahmood, J., Abideen, Z. U., & M.S., S. (2023). Deep Learning Techniques for Vehicle Detection and Classification from Images/Videos: A Survey. *Sensors*, *23*(10), 4832. https://doi.org/10.3390/s23104832

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection* (arXiv:2004.10934). arXiv. https://doi.org/10.48550/arXiv.2004.10934

Bretzke, W.-R. (2013). Global urbanization: A major challenge for logistics. *Logistics Research*, *6*(2), 57–62. https://doi.org/10.1007/s12159-013-0101-9

Chen, M., Jin, L., Jiang, Y., Gao, L., Wang, F., & Xie, X. (2016). Study on Leading Vehicle Detection at Night Based on Multisensor and Image Enhancement Method. *Mathematical Problems in Engineering*, *2016*, 1–13. https://doi.org/10.1155/2016/5810910

Chen, S., & Lin, W. (2019). Embedded System Real-Time Vehicle Detection based on Improved YOLO Network. *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 1400–1403. https://doi.org/10.1109/IMCEC46724.2019.8984055

Chen, X. (2024). Vehicle Object Detection Algorithm Based on Improved YOLOv8. *2024 5th International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, 1513–1516. https://doi.org/10.1109/CVIDL62147.2024.10603727

Chen, X., Jia, Y., Tong, X., & Li, Z. (2022). Research on Pedestrian Detection and DeepSort Tracking in Front of Intelligent Vehicle Based on Deep Learning. *Sustainability*, *14*(15), Article 15. https://doi.org/10.3390/su14159281

Cheng, X., Zhou, J., Song, J., & Zhao, X. (2023). A Highway Traffic Image Enhancement Algorithm Based on Improved GAN in Complex Weather Conditions. *IEEE Transactions on Intelligent Transportation Systems*, *24*(8), 8716–8726. IEEE Transactions on Intelligent Transportation Systems. https://doi.org/10.1109/TITS.2023.3258063

Datta, A., Islam Meghla, T., Khatun, T., Hasan Bhuiya, M., Rahman Shuvo, S., & Mahfujur Rahman, Md. (2020). Road Object Detection in Bangladesh using Faster R-CNN: A Deep Learning Approach. *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, 348–351. https://doi.org/10.1109/WIECON-ECE52138.2020.9397954

de Souza, A. M., Brennand, C. A., Yokoyama, R. S., Donato, E. A., Madeira, E. R., & Villas, L. A. (2017). Traffic management systems: A classification, review, challenges, and future perspectives. *International Journal of Distributed Sensor Networks*, *13*(4), 1550147716683612. https://doi.org/10.1177/1550147716683612

Diwan, T., Anirudh, G., & Tembhurne, J. V. (2023). Object detection using YOLO: Challenges, architectural successors, datasets and applications. *Multimedia Tools and Applications*, *82*(6), 9243–9275. https://doi.org/10.1007/s11042-022-13644-y

Du, P., Wang, X., Zheng, Q., Wang, X., Li, W., & Xu, X. (2024). Glare countering and exploiting via dual stream network for nighttime vehicle detection. *The Visual Computer*. https://doi.org/10.1007/s00371-024-03433-z

Fahim, S. H., Rasel, A. A. S., Sarker, A. R., & Chowdhury, T. (2023). *Bangla License Plate Detection Using YOLO v8 Model*. *3*(1).

Fei, Z., Dudu, G. U. O., Yang, W., Qingqing, W., Yin, Q. I. N., Zhuomin, Y., & Haijun, H. E. (2024). *Vehicle Detection Algorithm Based on Improved YOLOv8 in Traffic Surveillance. | Journal of Computer Engineering &amp; Applications | EBSCOhost*. *60*(6), 110. https://doi.org/10.3778/j.issn.1002-8331.2310-0101

Gothane, Dr. S. (2021). A Practice for Object Detection Using YOLO Algorithm. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 268–272. https://doi.org/10.32628/CSEIT217249

Guo, J., Shao, M., Chen, X., Yang, Y., & Sun, E. (2024). *Research on night-time vehicle target detection based on improved KSC-YOLO V5*. https://doi.org/10.21203/rs.3.rs-4244561/v1

Gupta, S. (2020). *YOLOv2 based Real Time Object Detection*. *8*(3).

Haupt, J., & Nowak, R. (2006). Compressive sampling vs. conventional imaging: 2006 IEEE International Conference on Image Processing, ICIP 2006. *2006 IEEE International Conference on Image Processing, ICIP 2006 - Proceedings*, 1269–1272. https://doi.org/10.1109/ICIP.2006.312576

Ho, N., Pham, M., Vo, N. D., & Nguyen, K. (2020). Vehicle Detection at Night Time. *2020 7th NAFOSTED Conference on Information and Computer Science (NICS)*, 250–255. https://doi.org/10.1109/NICS51282.2020.9335870

Huang, S., He, Y., & Chen, X. (2021). M-YOLO: A Nighttime Vehicle Detection Method Combining Mobilenet v2 and YOLO v3. *Journal of Physics: Conference Series*, *1883*(1), 012094. https://doi.org/10.1088/1742-6596/1883/1/012094

Hussain, M. (2023). YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines*, *11*(7), 677. https://doi.org/10.3390/machines11070677

Hussain, M. (2024). YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO. *IEEE Access*, *12*, 42816–42833. IEEE Access. https://doi.org/10.1109/ACCESS.2024.3378568

Inoue, H. (2018). *Data Augmentation by Pairing Samples for Images Classification* (arXiv:1801.02929). arXiv. https://doi.org/10.48550/arXiv.1801.02929

Islam, M., Haque, N., & Hadiuzzaman, M. (2024). *DEEGITS: Deep Learning based Framework for Measuring Heterogenous Traffic State in Challenging Traffic Scenarios* (arXiv:2411.08335). arXiv. https://doi.org/10.48550/arXiv.2411.08335

Jhong, S.-Y., Chen, Y.-Y., Hsia, C.-H., Lin, S.-C., Hsu, K.-H., & Lai, C.-F. (2021). Nighttime object detection system with lightweight deep network for internet of vehicles. *Journal of Real-Time Image Processing*, *18*(4), 1141–1155. https://doi.org/10.1007/s11554-021-01110-1

Jiang, P., Ergu, D., Liu, F., Ying, C., & Ma, B. (2022). A Review of Yolo Algorithm Developments. *Procedia Computer Science*, *199*, 1066–1073. https://doi.org/10.1016/j.procs.2022.01.135

Khrisne, D. C., Sudarma, M., Dwi Giriantari, I. A., & Made Wiharta, D. (2023). Nighttime Traffic Surveillance Using Glare Reduction and Zero-DCE-Based Low-Light Image Enhancement. *2023 International Conference on Smart-Green Technology in Electrical and Information Systems (ICSGTEIS)*, 161–165. https://doi.org/10.1109/ICSGTEIS60500.2023.10424144

Kuang, H., Chen, L., Gu, F., Chen, J., Chan, L., & Yan, H. (2016). Combining Region-of-Interest Extraction and Image Enhancement for Nighttime Vehicle Detection. *IEEE Intelligent Systems*, *31*(3), 57–65. IEEE Intelligent Systems. https://doi.org/10.1109/MIS.2016.17

Kuang, H., Zhang, X., Li, Y.-J., Chan, L. L. H., & Yan, H. (2017). Nighttime Vehicle Detection Based on Bio-Inspired Image Enhancement and Weighted Score-Level Feature Fusion. *IEEE Transactions on Intelligent Transportation Systems*, *18*(4), 927–936. https://doi.org/10.1109/TITS.2016.2598192

Kumar, S., Singh, S. K., Varshney, S., Singh, S., Kumar, P., Kim, B.-G., & Ra, I.-H. (2023). Fusion of Deep Sort and Yolov5 for Effective Vehicle Detection and Tracking Scheme in Real-Time Traffic Management Sustainable System. *Sustainability*, *15*(24), Article 24. https://doi.org/10.3390/su152416869

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications* (arXiv:2209.02976). arXiv. https://doi.org/10.48550/arXiv.2209.02976

Li, G., Yang, Y., Qu, X., Cao, D., & Li, K. (2021). A deep learning based image enhancement approach for autonomous driving at night. *Knowledge-Based Systems*, *213*, 106617. https://doi.org/10.1016/j.knosys.2020.106617

Li, X., Wang, W., Wu, L., Chen, S., Hu, X., Li, J., Tang, J., & Yang, J. (2020). *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection* (arXiv:2006.04388). arXiv. https://doi.org/10.48550/arXiv.2006.04388

Lima, A. A., Kabir, Md. M., Das, S. C., Hasan, Md. N., & Mridha, M. F. (2022). Road Sign Detection Using Variants of YOLO and R-CNN: An Analysis from the Perspective of Bangladesh. In M. S. Arefin, M. S. Kaiser, A. Bandyopadhyay, Md. A. R. Ahad, & K. Ray (Eds.), *Proceedings of the International Conference on Big Data, IoT, and Machine Learning* (pp. 555–565). Springer. https://doi.org/10.1007/978-981-16-6636-0_42

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). *Path Aggregation Network for Instance Segmentation* (arXiv:1803.01534). arXiv. https://doi.org/10.48550/arXiv.1803.01534

Mandal, G., Bhattacharya, D., & De, P. (2021). Real-time automotive night-vision system for drivers to inhibit headlight glare of the oncoming vehicles and enhance road visibility. *Journal of Real-Time Image Processing*, *18*(6), 2193–2209. https://doi.org/10.1007/s11554-021-01104-z

Miao, Y., Liu, F., Hou, T., Liu, L., & Liu, Y. (2020). A Nighttime Vehicle Detection Method Based on YOLO v3. *2020 Chinese Automation Congress (CAC)*, 6617–6621. https://doi.org/10.1109/CAC51589.2020.9326819

Mittal, D., Reddy, A., Ramadurai, G., Mitra, K., & Ravindran, B. (2018). Training a deep learning architecture for vehicle detection using limited heterogeneous traffic data. *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, 589–294. https://doi.org/10.1109/COMSNETS.2018.8328279

Onim, Md. S. H., Akash, M. I., Haque, M., & Hafiz, R. I. (2020). Traffic Surveillance using Vehicle License Plate Detection and Recognition in Bangladesh. *2020 11th International Conference on Electrical and Computer Engineering (ICECE)*, 121–124. https://doi.org/10.1109/ICECE51571.2020.9393109

Padilla Carrasco, D., Rashwan, H. A., García, M. Á., & Puig, D. (2023). T-YOLO: Tiny Vehicle Detection Based on YOLO and Multi-Scale Convolutional Neural Networks. *IEEE Access*, *11*, 22430–22440. https://doi.org/10.1109/ACCESS.2021.3137638

Parico, A. I. B., & Ahamed, T. (2021). Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT. *Sensors*, *21*(14), Article 14. https://doi.org/10.3390/s21144803

Rafi, M. M., Chakma, S., Mahmud, A., Rozario, R. X., Munna, R. U., Wohra, Md. A. A., Joy, R. H., Mahmud, K. R., & Paul, B. (2022). Performance Analysis of Deep Learning YOLO models for South Asian Regional Vehicle Recognition. *International Journal of Advanced Computer Science and Applications*, *13*(9). https://doi.org/10.14569/IJACSA.2022.01309100

Rahman, Z., Ami, A. M., & Ullah, M. A. (2020). A Real-Time Wrong-Way Vehicle Detection Based on YOLO and Centroid Tracking. *2020 IEEE Region 10 Symposium (TENSYMP)*, 916–920. https://doi.org/10.1109/TENSYMP50017.2020.9230463

Razzok, M., Badri, A., El Mourabit, I., Ruichek, Y., & Sahel, A. (2023). Pedestrian Detection and Tracking System Based on Deep-SORT, YOLOv5, and New Data Association Metrics. *Information*, *14*(4), Article 4. https://doi.org/10.3390/info14040218

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement* (arXiv:1804.02767). arXiv. https://doi.org/10.48550/arXiv.1804.02767

::RMMS:: (n.d.). Retrieved November 16, 2024, from https://www.rhd.gov.bd/RoadDatabase/roaddetail.asp?RoadID=1585

Rodríguez-Rangel, H., Morales-Rosales, L. A., Imperial-Rojo, R., Roman-Garay, M. A., Peralta-Peñuñuri, G. E., & Lobato-Báez, M. (2022). Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO. *Applied Sciences*, *12*(6), Article 6. https://doi.org/10.3390/app12062907

Rukmana, D. (2018). Rapid urbanization and the need for sustainable transportation policies in Jakarta. *IOP Conference Series: Earth and Environmental Science*, *124*(1), 012017. https://doi.org/10.1088/1755-1315/124/1/012017

Saha, B., Islam, M. J., Mostaque, S. K., Bhowmik, A., Taton, T. K., Chowdhury, M. N. H., & Reaz, M. B. I. (2024). *Bangladeshi Native Vehicle Detection in Wild* (arXiv:2405.12150). arXiv. https://doi.org/10.48550/arXiv.2405.12150

Sang, J., Wu, Z., Guo, P., Hu, H., Xiang, H., Zhang, Q., & Cai, B. (2018). An Improved YOLOv2 for Vehicle Detection. *Sensors*, *18*(12), 4272. https://doi.org/10.3390/s18124272

Shah Junayed, M., Baharul Islam, M., Sadeghzadeh, A., & Aydin, T. (2021). Real-Time YOLO-based Heterogeneous Front Vehicles Detection. *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, 1–7. https://doi.org/10.1109/INISTA52262.2021.9548650

Shao, X., Wei, C., Shen, Y., & Wang, Z. (2021). Feature Enhancement Based on CycleGAN for Nighttime Vehicle Detection. *IEEE Access*, *9*, 849–859. https://doi.org/10.1109/ACCESS.2020.3046498

Shen, Q. (1997). Urban transportation in Shanghai, China: Problems and planning implications. *International Journal of Urban and Regional Research*, *21*(4), 589–606. https://doi.org/10.1111/1468-2427.00103

Sheng, W., Shen, J., Huang, Q., Liu, Z., Ding, Z., Sheng, W., Shen, J., Huang, Q., Liu, Z., & Ding, Z. (2024). Multi-objective pedestrian tracking method based on YOLOv8 and improved DeepSORT. *Mathematical Biosciences and Engineering*, *21*(2), Article mbe-21-02-077. https://doi.org/10.3934/mbe.2024077

Shi, C., Lei, M., You, W., Ye, H., & Sun, H. (2024). Enhanced floating debris detection algorithm based on CDW-YOLOv8. *Physica Scripta*, *99*(7), 076019. https://doi.org/10.1088/1402-4896/ad5657

Shomee, H. H., & Sams, A. (2021). License Plate Detection and Recognition System for All Types of Bangladeshi Vehicles Using Multi-step Deep Learning Model. *2021 Digital Image Computing: Techniques and Applications (DICTA)*, 01–07. https://doi.org/10.1109/DICTA52665.2021.9647284

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Singh, A., & Banga, D. V. K. (2013). *NIGHT IMAGE ENHANCEMENT USING FUSION TECHNIQUE*. *2*(5).

Sohan, M., Ram, T., & Ch, V. (2024). *A Review on YOLOv8 and Its Advancements* (pp. 529–545). https://doi.org/10.1007/978-981-99-7962-2_39

Soylu, E., & Soylu, T. (2023). A performance comparison of YOLOv8 models for traffic sign detection in the Robotaxi-full scale autonomous vehicle competition. *Multimedia Tools and Applications*, *83*(8), 25005–25035. https://doi.org/10.1007/s11042-023-16451-1

Tabassum, S., Ullah, Md. S., Al-Nur, N. H., & Shatabda, S. (2020). Native Vehicles Classification on Bangladeshi Roads Using CNN with Transfer Learning. *2020 IEEE Region 10 Symposium (TENSYMP)*, 40–43. https://doi.org/10.1109/TENSYMP50017.2020.9230991

Telaumbanua, A., Larosa, T., Pratama, P., Fauza, R., & Husein, A. (2023). Vehicle Detection and Identification Using Computer Vision Technology with the Utilization of the YOLOv8 Deep Learning Method. *Sinkron*, *8*, 2150–2157. https://doi.org/10.33395/sinkron.v8i4.12787

Terven, J., Córdova-Esparza, D.-M., & Romero-González, J.-A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Machine Learning and Knowledge Extraction*, *5*(4), Article 4. https://doi.org/10.3390/make5040083

Wang, B., Li, Y.-Y., Xu, W., Wang, H., & Hu, L. (2024). Vehicle–Pedestrian Detection Method Based on Improved YOLOv8. *Electronics*, *13*(11), 2149. https://doi.org/10.3390/electronics13112149

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). *YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors*. 7464–7475. https://openaccess.thecvf.com/content/CVPR2023/html/Wang_YOLOv7_Trainable_Bag-of-Freebies_Sets_New_State-of-the-Art_for_Real-Time_Object_Detectors_CVPR_2023_paper.html

Wang, H., Liu, C., Cai, Y., Chen, L., & Li, Y. (2024). YOLOv8-QSD: An Improved Small Object Detection Algorithm for Autonomous Vehicles Based on YOLOv8. *IEEE Transactions on Instrumentation and Measurement*, *73*, 1–16.

IEEE Transactions on Instrumentation and Measurement. https://doi.org/10.1109/TIM.2024.3379090

Wang, M., & Ren, L. (2023). *SSB-YOLO: A vehicle object detection algorithm based on improved YOLOv8*. https://doi.org/10.21203/rs.3.rs-3743453/v1

Wei, S., & Yu, S. (2024). Improvement of Nighttime Vehicle Detection Algorithm Based on YOLOv8n. *Proceedings of the 2024 International Conference on Computer and Multimedia Technology*, 430–436. https://doi.org/10.1145/3675249.3675323

Wen, H., Dai, F., & Yuan, Y. (2021). *A Study of YOLO Algorithm for Target Detection*.

Wojke, N., Bewley, A., & Paulus, D. (2017). *Simple Online and Realtime Tracking with a Deep Association Metric* (arXiv:1703.07402). arXiv. https://doi.org/10.48550/arXiv.1703.07402

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016). *Understanding data augmentation for classification: When to warp?* (arXiv:1609.08764). arXiv. https://doi.org/10.48550/arXiv.1609.08764

Wu, T., & Dong, Y. (2023). YOLO-SE: Improved YOLOv8 for Remote Sensing Object Detection and Recognition. *Applied Sciences*, *13*, 12977. https://doi.org/10.3390/app132412977

Wu, Y., Tang, Y., & Yang, T. (2023). An improved nighttime people and vehicle detection algorithm based on YOLO v7. *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, 266–270. https://doi.org/10.1109/NNICE58320.2023.10105722

Xu, B., Wang, B., & Gu, Y. (2019). Vehicle Detection in Aerial Images Using Modified YOLO. *2019 IEEE 19th International Conference on Communication Technology (ICCT)*, 1669–1672. https://doi.org/10.1109/ICCT46805.2019.8947049

Xu, K., Gong, H., & Liu, F. (2020). Vehicle detection based on improved multitask cascaded convolutional neural network and mixed image enhancement. *IET Image Processing*, *14*(17), 4621–4632. https://doi.org/10.1049/iet-ipr.2020.1005

Ye, S., Yin, J.-L., Chen, B.-H., Chen, D., & Wu, Y. (2020). Single Image Glare Removal Using Deep Convolutional Networks. *2020 IEEE International Conference on Image Processing (ICIP)*, 201–205. https://doi.org/10.1109/ICIP40778.2020.9190712

Yin, T., Chen, W., Liu, B., Li, C., & Du, L. (2023). Light “You Only Look Once”: An Improved Lightweight Vehicle-Detection Model for Intelligent Vehicles under Dark Conditions. *Mathematics*, *12*(1), 124. https://doi.org/10.3390/math12010124

Yoon, S., & Cho, J. (2023). Low-Light Image Contrast Enhancement with Adaptive Noise Attenuator for Augmented Vehicle Detection. *Electronics*, *12*(16), 3517. https://doi.org/10.3390/electronics12163517

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., & Yoo, Y. (2019). *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features* (arXiv:1905.04899). arXiv. https://doi.org/10.48550/arXiv.1905.04899

Zhang, H., Yang, K.-F., Li, Y.-J., & Chan, L. L.-H. (2024). Night-Time Vehicle Detection Based on Hierarchical Contextual Information. *IEEE Transactions on Intelligent Transportation Systems*, 1–14. IEEE Transactions on Intelligent Transportation Systems. https://doi.org/10.1109/TITS.2024.3395666

Zhang, Y., Guo, Z., Wu, J., Tian, Y., Tang, H., & Guo, X. (2022). Real-Time Vehicle Detection Based on Improved YOLO v5. *Sustainability*, *14*(19), 12274. https://doi.org/10.3390/su141912274

Zhao, K., Liu, J., & Xiao, X. (2021). Low-illumination environment vehicle detection based on optimized adaptive image enhancement. In Y. Zhang & D. Zhang (Eds.), *2021 International Conference on Image, Video Processing, and Artificial Intelligence* (p. 16). SPIE. https://doi.org/10.1117/12.2608155

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), Article 07. https://doi.org/10.1609/aaai.v34i07.6999

Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T.-Y., Shlens, J., & Le, Q. V. (2019). *Learning Data Augmentation Strategies for Object Detection* (arXiv:1906.11172). arXiv. https://doi.org/10.48550/arXiv.1906.11172

Zubaer, K. H., Alam, Q. M., Toha, T. R., Salim, S. I., & Islam, A. B. M. A. A. (2020). Towards simulating non-lane based heterogeneous road traffic of less developed countries using authoritative polygonal GIS map. *Simulation Modelling Practice and Theory*, *105*, 102156. https://doi.org/10.1016/j.simpat.2020.102156

APPENDIX

Pseudocodes for the used algorithms in this thesis will be provided here.

Algorithm 1: Extract Frames from Video

Step	Description
Input:	- video_path (Path to the video file) - output_directory (Directory to save extracted frames)
Output:	- Extracted frames saved as images in the specified output directory
1	BEGIN
2	// Open the video file
3	vidcap ← OPEN_VIDEO(video_path)
4	// Get frames per second (fps) of the video
5	fps ← GET_VIDEO_FPS(vidcap)
6	// Initialize frame counter
7	count ← 0
8	// Calculate the interval for capturing one frame per second
9	interval ← CONVERT_TO_INTEGER(fps)
10	// Loop through the video frames
11	WHILE vidcap IS_OPEN DO
12	ret, image ← READ_FRAME(vidcap)
13	IF ret THEN
14	IF count MOD interval = 0 THEN
15	frame_number ← INTEGER_DIVISION(count, interval)
16	SAVE_IMAGE(output_directory + frame_number + ".jpg", image)

17	END IF
18	count ← count + 1
19	ELSE
20	BREAK
21	END IF
22	END WHILE
23	// Release the video capture object
24	RELEASE_VIDEO(vidcap)
25	DESTROY_ALL_WINDOWS()
26	END

Algorithm 2: Convert Polygon Annotations to Bounding Box Format

Step	Description
Input:	- input_directory (Path to annotation files)
 - output_directory (Path to save converted annotations)
 - img_directory (Path to images)
Output:	- Converted annotation files with bounding box coordinates
1	BEGIN
2	FUNCTION convert_polygon_to_bbox(components):
3	xs ← extract x-coordinates from components
4	ys ← extract y-coordinates from components
5	x_min ← MIN(xs), x_max ← MAX(xs)
6	y_min ← MIN(ys), y_max ← MAX(ys)
7	x_center ← (x_min + x_max) / 2, y_center ← (y_min + y_max) / 2

```
    width ← x_max - x_min, height ← y_max - y_min
    RETURN [x_center, y_center, width, height]
END FUNCTION
FUNCTION is_polygon(components):
    RETURN LENGTH(components) > 5
END FUNCTION
FUNCTION convert_and_save_annotations(input_directory, output_directory, img_directory):
    IF NOT EXISTS(output_directory) THEN CREATE_DIRECTORY(output_directory)
    FOR filename IN LIST_FILES(input_directory) DO
        IF filename ENDS WITH ".txt" THEN
            input_file_path ← CONCAT(input_directory, filename)
            output_file_path ← CONCAT(output_directory, filename)
            img_file_path ← CONCAT(img_directory, REPLACE(filename, ".txt", ".jpg"))
            TRY: img ← OPEN_IMAGE(img_file_path)
            img_width, img_height ← GET_IMAGE_SIZE(img)
            CATCH FileNotFoundError: PRINT "Image file not found for", filename, "skipping..."
            CONTINUE
        OPEN input_file_path AS infile FOR READING
        OPEN output_file_path AS outfile FOR WRITING
        FOR each line IN infile DO
```

28	components ← SPLIT(line)
29	class_id ← components[0]
30	coordinates ← CONVERT_TO_FLOAT(components[1:])
31	IF is_polygon(coordinates) THEN bbox ← convert_polygon_to_bbox(coordinates)
32	ELSE bbox ← coordinates
33	bbox_line ← CONCAT(class_id, " ", bbox)
34	WRITE outfile, bbox_line
35	END FOR
36	CLOSE infile
37	CLOSE outfile
38	END FOR
39	END FUNCTION
40	// Call the conversion function with parameters
41	CALL convert_and_save_annotations(input_directory, output_directory, img_directory)
42	PRINT "Converted annotations have been saved to:", output_directory
43	END

Algorithm 3: Object Tracking using YOLO and DeepSORT

Step	Description
Input:	- video_path (Path to the video file) - model_path (Path to the trained YOLO model) - output_csv_path (Path to save trajectory data)
Output:	- CSV file containing tracked object trajectories

```
BEGIN
Load YOLO model
model ← LOAD_YOLO(model_path)
Initialize DeepSORT tracker
deepsort ← INITIALIZE_DeepSORT()
Open video file
cap ← OPEN_VIDEO(video_path)
Initialize empty list for trajectory data
trajectory_data ← EMPTY_LIST()
Initialize frame index
frame_idx ← 0
Loop through the video frames
WHILE True DO
    ret, frame ← READ_FRAME(cap)
    IF NOT ret THEN BREAK
    frame_idx ← frame_idx + 1
    Run YOLO detection on the frame
    results ← RUN_YOLO(model, frame)
    boxes ← EXTRACT_BOUNDING_BOXES(results)
    confs ← EXTRACT_CONFIDENCE_SCORES(results)
    class_ids ← EXTRACT_CLASS_IDS(results)
    Filter detections for class ID 11
    class_11_boxes ← EMPTY_LIST()
    FOR i IN RANGE(LENGTH(class_ids)) DO
        IF class_ids[i] = 11 THEN
            APPEND class_11_boxes ← [boxes[i], confs[i], class_ids[i]]
        END IF
    END FOR
    IF class_11_boxes IS NOT EMPTY THEN
        detections ← EMPTY_LIST()
        FOR box, conf, * IN class_11_boxes DO
            x1, y1, w, h ← box
            detection ← [x1 - w / 2, y1 - h / 2, x1 + w / 2, y1 + h / 2, conf]
            APPEND detections ← detection
        END FOR
        tracker ← deepsort.UPDATE_TRACKS(detections, frame)
        FOR track IN tracker DO
            track_id ← track[1]
            x, y, w, h ← track[2]
            APPEND trajectory_data ← [frame_idx, track_id, x, y, w, h]
        END FOR
    END IF
END WHILE
Save the trajectory data to CSV
```

45	df ← CREATE_DATAFRAME(trajectory_data, ['Frame', 'ID', 'X', 'Y', 'Width', 'Height'])
46	SAVE_CSV(df, output_csv_path)
47	Release video capture and close windows
48	RELEASE_VIDEO(cap)
49	DESTROY_ALL_WINDOWS()
50	PRINT "Trajectory data saved to", output_csv_path
51	END

Algorithm 4: Compute Lyapunov Exponents for Pedestrian Trajectories

Step	Description
Input:	- trajectory_csv_path (Path to CSV file containing pedestrian trajectories)
Output:	- Lyapunov exponents for each pedestrian trajectory and histogram plot
1	BEGIN
2	Load the CSV file
3	data ← READ_CSV(trajectory_csv_path, PRESERVE_VARIABLE_NAMES)
4	Extract relevant columns
5	frame ← EXTRACT_COLUMN(data, "Frame")
6	objectID ← EXTRACT_COLUMN(data, "ObjectID")
7	x_center ← EXTRACT_COLUMN(data, "x1")
8	y_center ← EXTRACT_COLUMN(data, "y1")
9	Group data by unique object IDs
10	uniqueIDs ← FIND_UNIQUE_VALUES(objectID)

11	trajectories ← INITIALIZE_EMPTY_LIST(LENGTH(uniqueIDs))
12	FOR i FROM 1 TO LENGTH(uniqueIDs) DO
13	id ← uniqueIDs[i]
14	idx ← FIND_INDICES(objectID = id)
15	trajectories[i] ← CONCATENATE_COLUMNS(x_center[idx], y_center[idx])
16	END FOR
17	Compute Lyapunov exponents for each trajectory
18	lyapunovExponents ← INITIALIZE_ARRAY(LENGTH(uniqueIDs), NaN)
19	FOR i FROM 1 TO LENGTH(trajectories) DO
20	traj ← trajectories[i]
21	TRY
22	lyapunovExponents[i] ← COMPUTE_ROSENSTEIN_LYAPUNOV(traj, 3, 10)
23	CATCH ERROR
24	lyapunovExponents[i] ← NaN
25	END TRY
26	END FOR
27	Display results
28	PRINT_TABLE(uniqueIDs, lyapunovExponents)
29	Plot histogram of Lyapunov exponents
30	CREATE_HISTOGRAM(lyapunovExponents, BIN_WIDTH=0.1, COLOR="blue")

```
LABEL_X_AXIS("Lyapunov Exponent")
LABEL_Y_AXIS("Frequency")
SET_TITLE("Histogram of Lyapunov Exponents")
Function: Rosenstein's Lyapunov Exponent Calculation
FUNCTION COMPUTE_ROSENSTEIN_LYAPUNOV(traj, dim, tau)
    embedded ← TIME_DELAY_EMBEDDING(traj, dim, tau)
    N ← GET_NUMBER_OF_ROWS(embedded)
    d0 ← INITIALIZE_ARRAY(N, 0)
    neighborIdx ← INITIALIZE_ARRAY(N, 0)
    FOR i FROM 1 TO N DO
        distances ← COMPUTE_DISTANCES(embedded, embedded[i])
        EXCLUDE_TEMPORAL_NEIGHBORS(distances, i, tau)
        d0[i], neighborIdx[i] ← FIND_NEAREST_NEIGHBOR(distances)
    END FOR
    maxIter ← MIN(100, N - MAX(neighborIdx))
    divergence ← INITIALIZE_ARRAY(maxIter, 0)
    FOR j FROM 1 TO maxIter DO
        validIdx ← FIND_VALID_INDICES(N, j)
        divergence[j] ← COMPUTE_MEAN_LOG_DISTANCE(embedded,
neighborIdx, validIdx, j)
    END FOR
    timeVec ← CREATE_SEQUENCE(1, maxIter)
    coeff ← LINEAR_FIT(timeVec, divergence)
```

53	RETURN coeff[1]
54	END FUNCTION
55	Function: Time-Delay Embedding
56	FUNCTION TIME_DELAY_EMBEDDING(traj, dim, tau)
57	N ← GET_NUMBER_OF_ROWS(traj)
58	M ← N - (dim - 1) * tau
59	IF M ≤ 0 THEN THROW ERROR
60	embedded ← INITIALIZE_MATRIX(M, dim * GET_NUMBER_OF_COLUMNS(traj))
61	FOR i FROM 1 TO dim DO
62	embedded[:, (i-1)*COLUMNS(traj) + 1 : i*COLUMNS(traj)] ← traj[(1:M) + (i-1)*tau, :]
63	END FOR
64	RETURN embedded
65	END FUNCTION
66	END

Algorithm 5: Compute Approximate Entropy and Analyze Pedestrian Behavior

Step	Pseudocode
1	Input: Time series data with identifier, timestamp, and positional coordinates
2	Output: Approximate Entropy values for derived metrics and visualization
3	BEGIN
4	Load the input data file
5	Validate required columns exist (ID, timestamp, coordinates)

6	Sort data by ID and timestamp
7	Initialize derived metric columns
8	FOR each unique identifier in dataset DO
9	Extract data subset for current identifier
10	Calculate first derived metric (e.g., velocity)
11	Calculate second derived metric (e.g., direction change)
12	Update dataset with calculated metrics
13	END FOR
14	Set ApEn parameters (embedding dimension m, tolerance multiplier r)
15	Initialize results collection
16	FOR each unique identifier in dataset DO
17	Extract data subset for current identifier
18	IF length of subset < minimum required samples THEN CONTINUE
19	Extract first metric values
20	Extract second metric values
21	Calculate tolerance for first metric
22	Calculate tolerance for second metric
23	Compute ApEn for first metric
24	Compute ApEn for second metric
25	IF both ApEn values are valid THEN
26	Store results with identifier and ApEn values
27	END IF
28	END FOR
29	Export results to output file
30	Generate visualization of ApEn distributions
31	END
32	FUNCTION ComputeApproximateEntropy(U, m, r)
33	FUNCTION MaxDistance(x_i, x_j)
34	return maximum absolute difference between corresponding elements

35	END FUNCTION
36	FUNCTION Phi(m)
37	N ← length(U) - m + 1
38	Initialize count array C of size N
39	FOR i = 1 to N DO
40	x_i ← subsequence of U starting at i with length m
41	count ← 0
42	FOR j = 1 to N DO
43	x_j ← subsequence of U starting at j with length m
44	IF MaxDistance(x_i, x_j) ≤ r THEN
45	count ← count + 1
46	END IF
47	END FOR
48	C[i] ← count/N
49	END FOR
50	return natural logarithm of the sum of C divided by N
51	END FUNCTION
52	IF length(U) < m + 1 THEN
53	return undefined
54	END IF
55	phi_m ← Phi(m)
56	phi_m_plus_1 ← Phi(m + 1)
57	return phi_m - phi_m_plus_1
58	END FUNCTION