Deep Learning Based Crime Detection from Surveillance Videos

Ayele Nugusie Feyissa

A Thesis Submitted to

The Department of Computer Science and Engineering

School Of Electrical Engineering and Computing (SOEEC)

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February, 2025

Adama, Ethiopia

Deep Learning Based Crime Detection from Surveillance Videos

Ayele Nugusie Feyissa

Advisor: Dr. Getinet Yilma

A Thesis Submitted to

The Department of Computer Science and Engineering

School Of Electrical Engineering and Computing (SOEEC)

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February, 2025

Adama, Ethiopia

Declaration

I hereby declare that this Master's Thesis entitled "**Deep Learning Based Crime Detection from Surveillance Videos**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Ayele Nugusie Feyissa	_____	_____
Name of Students	Signature:	Date

Advisor Recommendation

I, the advisor of this thesis, hereby certify that I have closely advised the student while developing this thesis and read the draft thesis entitled **"*Deep Learning Based Crime Detection from Surveillance Videos*"** prepared under my guidance by **Ayele Nugusie Feyissa.** Therefore, I recommend the submission of the thesis to the department for further review and evaluation.

Dr. Getinat Yilma	_____	_____
Main Advisor	Signature	Date

Approval of Board of Examiners

We, the, members, of the board of examiners of the final open defense by Ayele Nugusie, have readout and assessed his/her thesis *entitled "**Deep Learning Based Crime Detection from Surveillance Videos**"* and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial the fulfillment of the requirement of the Degree of Computer Science and Engineering.

(Supervisor/Advisor)	(Signature)	(Date)
(Chairperson)	(Signature)	(Date)
(Internal Examiner)	(Signature)	(Date)
(External Examiner)	(Signature)	(Date)

ACKNOWLEDGMENT

I would like to acknowledge God for his grace, strength and good health as I undertook this research. I would like to thank my esteemed Advisor Dr. Getnet Yilma for his invaluable supervision, support, his friendly approach, readiness and willingness to advise on the research, his critical comments, commitment to guide, and support greatly improved this thesis work. My gratitude extends to the School of Electrical and Computing for the funding opportunity to undertake my studies at the Department of Computer Science and Engineering, Adama Science and Technology University.

Finally, I owe particular gratitude to my relatives for, their unconditional love and sacrifices, which have laid the foundation for all my achievements. Their belief in me has always been an inspiration. To my siblings, for their patience, understanding, and support. They have all been my rock, providing me with the emotional and moral support needed to overcome obstacles and stay focused on my goals. I also give special thanks to my friends, due to their companionship and encouragement have made this journey enjoyable. Their continuous support, whether through late-night discussions, or study sessions has been a source of motivation and comfort. Without these remarkable individual contributions and support, this research would not have been possible. Thank you all for being a part of this work.

TABLE OF CONTENTS

ACKNOWLEDGMENT iv

LIST OF TABLES ix

LIST OF FIGURES x

ACRONYMS xi

ABSTRACT xii

CHAPTER ONE 1

1. INTRODUCTION 1

1.1 Background of Study 1

1.2 Motivation of Study 3

1.3 Statement of the Problem 3

1.4 Research Questions 4

1.5 Objective of the Study 4

1.5.1 General Objective 4

1.5.2 Specific Objectives 4

1.6 The methodology of the Study 5

1.6.1 Modeling Approach 5

1.7 Scope and Limitations of the study 6

1.7.1 The scope of the study 6

1.7.2 The Limitations of the Study 6

1.8 Significance of Study 6

1.9 Organization of the Thesis 7

CHAPTER TWO 8

2. LITERATURE REVIEW AND RELATED WORKS 8

2.1 Overview of the Chapter 8

2.2 Introduction 8
2.2.1 History of Crime detection 9
2.3 Machine Learning and Deep Learning Method for Crime Detection 10
2.3.1 YOLOnV8 11
2.3.2 YOLOnanoV11 14
2.4 Related Works 18
CHAPTER THREE 21
3. METHODOLOGY 21
3.1 Overview of Chapter 21
3.2 Dataset 21
3.2.1 Dataset Preparation 22
3.2.2 Data Preprocessing 24
3.3 Feature Extraction 26
3.4 Model Development 26
3.5 Model Evaluation 27
3.5.1 Evaluation Metrics 27
3.6 Development Tools 28
3.6.1 Design Tools 28
3.6.2 Hardware Tools 29
CHAPTER FOUR 30
4. PROPOSED MODEL ARCHITECTURE 30
4.1 Chapter overview 30
4.2 Problem Formulation 30
4.3 The Proposed Architecture 31
4.4 Proposed Model Architectures 32

4.4.1 YOLOv8n Model Architecture 32
4.4.2 YOLOv11n Model Architecture 33
4.5 Loss Function of the Proposed Model 34
4.5.1 Smooth L1 Loss 34
CHAPTER FIVE 35
5. IMPLEMENTATION OF THE PROPOSED MODEL ARCHITECTURE 35
5.1 Chapter Overview 35
5.2 Working Environment 35
5.3 Environment Setup 35
5.3.1 Tools and Packages 35
5.4 Training Procedure 36
5.5 Dataset Description and Loading of Dataset 36
5.5.1 Dataset Loading 37
5.5.2 Model Hyperparameters 38
5.6 Experiments 40
5.6.1 YOLOv8n Model Implementation 40
5.6.2 Implementation Of Yolov11n Model 41
CHAPTER SIX 43
6. RESULT AND DISCUSSIONS 43
6.1 Chapter Overview 43
6.2 Result Analysis 43
6.3 Results for Deep Learning Models 48
6.3.1 YOLOV8n Model Results 48
6.3.2 YOLOV11n Model Results 49
6.4 Hyper parameter Tuning Result 55

6.5 Discussion of Results 61
CHAPTER SEVEN 62
7. CONCLUSION AND FUTURE WORK 62
7.1 Conclusion 62
7.2 Future works 63
REFERENCES 64
APPENDIXES 67

LIST OF TABLES

Table 2.1. Comparison of different YOLO versions 18
Table 2.2: Summary of Related Work 18
Table 3.1 Dataset Classification 21
Table 3.2 : Hard ware Tools used for proposed 29
Table 5.1 Package Utilized study 36
Table 5.2 Hyperparameter 40
Table 6.1 Results of YOLONanoV8n 49
Table 6.2 Results of YOLOV11n 50
Table 6.3 Hyper parameter (Epochs) tuning result 56
Table 6.4 Model comparison 57
Table 6.5 Related works comparison 60

LIST OF FIGURES

Figure 1.1: Overview of the Research Process 5
Figure 2.1. Yolov8n Architecture(C. Y. Wang & others, 2023). 14
Figure 2.2 Yolov11n Architecture(Li & Wang, 2024). 15
Figure 3.1 Sample video frames from five classes 23
Figure 3.2 Annotation files 24
Figure 4.1 : Proposed Framework for Crime Detection 32
Figure 5.1 Validation Data Distribution 37
Figure 5.2 Training Data Distribution 38
Figure 6.1 Precision-Confidence Curve yolov8n 43
Figure 6.2 Recall-Confidence Curve yolov8n 44
Figure 6.3 Precision-Recall Curve yolov8n 45
Figure 6.4: precision confidence curve yolov11n 46
Figure 6.5 Recall confidence curve yolov11n 47
Figure 6.6: Precision-recall curve yolov11n 48
Figure 6.7 Performance Metrics of YOLOv8n model 50
Figure 6.8 Training and validation losses and Evaluate performance of yolov8n 51
Figure 6.9 Confusion Metrics of Yolov8n 52
Figure 6.10 Performance Metrics of YOLOv11n model 53
Figure 6.11 Training , validation losses and Evaluate performance of yolov11n 54
Figure 6.12 Confusion Metrics of Yolov11 55
Figure 6.13 Train batch bounding box yolov8 57
Figure 6.14 Train batch bounding box yolov11 58
Figure 6.15 Actual and predicted bounding box yolov8 58
Figure 6.16 Actual and predicted bounding box yolov11 59

ACRONYMS

CSPNet - Cross-Stage Partial Network

SPPF -Spatial Pyramid Pooling Fast

OBB - Oriented Bonding Boxes

YOLO - You Only Look Once

PANet - Path Aggregation Network

mAP - Mean Average precision

PGI - Programmable Gradient Information

GELAN - Generalized Efficient Layer Aggregation Network

PANet - Path Aggregation Network

FPN - Feature Pyramid Network

IoU - Intersection over Union

NMS - Non-Maximum Suppression

SOTA - state of the art

CNNs - convolutional neural networks

ABSTRACT

The primary objective of deep learning-based crime detection is enhancing public security through active identification of criminal activities. Traditional surveillance systems rely on human operation of live video streams manually, which is time-consuming and prone to human error. To address this, new algorithms such as YOLOv8n and YOLOv11n have been designed as efficient crime detection automation with enhanced accuracy and speed. YOLOv8n and YOLOv11n, both object detection-capable, are particularly trained to rummage through massive databases of surveillance footage and flag abnormal activity or behavior. With the capability to learn patterns and signatures from labeled crime-related data, the models possess the potential to identify potential threats with high levels of accuracy. The performance of the models is measured by metrics like Mean Average Precision (mAP), Precision, and Recall. YOLOv8n scores 0.993 in, precision, 0.993 mAP. Meanwhile, YOLOv11n is an improvement on its previous model, with a recorded precision 0.995, mAP of 0.995 and better accuracy as well as faster processing for criminal detection activities. Finally, the integration of YOLOv8n and YOLOv11n in crime detection systems will revolutionize public security through the identification and prevention of crime. These deep learning models significantly enhance surveillance and provide security personnel with an effective tool to manage criminal behavior. However, concerns regarding data availability, computation power, and ethics need to be addressed to ensure effective and ethical use of this technology.

Keyword*: *Deep Learning, Crime Detection, Surveillance videos, Yolov8n and yolov11n , UCF dataset

CHAPTER ONE

1. INTRODUCTION

1.1 Background of Study

Crime detection and their prediction is a fundamental process to reduce criminal activities before they actually happen, and therefore, the significance of detection methods is very clear. In the last few years, the computer vision community has pushed on crowd behavior analysis and made a lot of progress in crowd abnormality detection (Del Giorno et al., 2016) (Murino & Puppo, 2015)(Rabiee et al., 2016).

Video surveillance systems play a crucial role in monitoring for security and law enforcement purposes. The area of smart video surveillance has become a prominent topic of research within computer vision applications. These systems generate vast amounts of data that cannot be continuously monitored by human security staff, necessitating a solution capable of detecting unusual or abnormal events in real time (Bhat & S, n.d.). Violence detection aims to timely locate the start and the end of violent events with minimum human resource cost(Wu et al., 2020). Deep learning-based crime detection from surveillance videos is an application of computer vision technology that leverages the power of deep learning algorithms to analyze visual data from surveillance cameras and detect criminal activities. With the growth of CCTV Camera and other surveillance systems, law enforcement agencies are increasingly relying on machine learning algorithms to help prevent and solve crimes.

The basic idea behind surveillance-based crime detection using deep learning is to train a computer vision system to recognize patterns in the visual data that are indicative of criminal activity. By analyzing the data captured by surveillance cameras, the system can detect and classify criminal activity(Anti´c & Ommer, 2011).in the past decade, the number of violent crime rates has reduced across the globe(Piza et al., 2019) suggest that this can majorly be attributed to the explosive rise in the number of surveillance systems being employed.

Deep learning algorithms are particularly well-suited for this task because they are able to learn complex patterns and relationships in visual data. This means that these algorithms can be trained to accurately recognize and classify criminal activity, even in situations where there are many variables at play.

There are currently just a few datasets for anomaly detection accessible, the majority of which are small-scale or confined to particular scenarios, such as UCSD datasets(Rui, 2023). These

datasets are also utilized for semi supervised anomaly detection with normal training samples at first. Only UCF-Crime dataset is presently publicly accessible for the challenge under video level scenario, to our understanding(Sultani et al., n.d.). As a result, we conducted experiments using the UCF-Crime dataset. The primary objective of the proposed method is to create a framework for detecting anomalies in surveillance video feeds. This is achieved by applying the anomaly detection algorithm to individual frames, allowing the system to manage data effectively. This paper presents a unique deep learning-based approach for anomaly detection aimed at enhancing security in campus and airport environments. The model focuses on identifying and classifying irregularities observed in video surveillance, such as: fighting, Road Accidents, shootings, stealing.

YOLO (You Only Look Once) is one of the most advanced object detection frameworks available today, known for its speed and accuracy(Plastiras et al., 2018). YOLOv8, the newest version in this series, builds on earlier iterations by enhancing detection capabilities, especially in difficult conditions. This framework processes images in a single pass, making it ideal for applications that need real-time analysis. Its architecture is designed to balance speed and accuracy, enabling effective object detection even in busy or dynamic settings.

The aim of this research is to come up with an efficient AI model using YOLOv8 for gun and other weapon detection in various environments with high precision, thus scaling up security measures in public places. In this work, we provide the model development methodology, data collection, training procedures, and performance measures. We also discuss the implications of the results and how the system developed can be used in real-world applications

The YOLO (You Only Look Once) model is designed primarily for object detection in images and video frames, and not for explicitly "scene information" extraction in the way one might expect from systems designed for scene understanding, semantic segmentation, or more abstract video analysis. However, YOLO can also provide valuable information about the scene by detecting and localizing various objects present in the video

YOLO, proposed by Redmon et al. (2016), changed the game for object detection by casting it as a single regression problem, thereby enabling the model to predict both the class of objects and their bounding boxes in one pass. This new approach supports real-time operation, which is critical for applications like surveillance and security. Not only is the speed of YOLO impressive, but its ability to recognize a wide number of objects simultaneously is also powerful,

and it is highly suited to dynamic scenes like video monitoring where there are multiple objects and people that might appear in one frame. If used to detect crime, YOLO can identify vital objects and tasks, like firearms, suspicious behavior, or aggressiveness, meaning response can come sooner and supervision can be accomplished more efficiently.

1.2 Motivation of Study

The crime rate increases poses challenges for law enforcement and public safety. The traditional methods of crime detection and prevention rely on manual surveillance and reactive responses, which are inefficient and prone to dely. In the current situation, a crime detection system automated is a must because it is needed by the law enforcers and the citizens too. The process would help in lessening the crime because detection would be simple. Moreover, with the rising trend of smart cities, crime detection systems are being integrated for security purposes.

Overall, crime detection from surveillance using deep learning has the potential to make communities more secure by augmenting the ability of law enforcement agencies to identify and respond to potential threats.

1.3 Statement of the Problem

In today's world crime continues to pose serious threats to the security and well-being of individuals and societies worldwide. Most of the crime detection available in our country are traditional crime detection methods that rely primarily on manual surveillance, small-scale data analysis, and reactive measures, which are time-consuming, prone to human errors, and incapable of detecting and preventing crimes. Moreover, the sheer volumes and complexity of information produced from an array of dissimilar sources such as CCTV cameras, social networks, and sensors are difficult to analyze and pick out crime patterns using the traditional methods. In spite of advancements in technology, there remains a deficiency in tapping into the full potential of artificial intelligence, particularly deep learning, to enhance crime detection. Deep learning algorithms have been proved to excel in dealing with and processing large amounts of data, recognizing complex patterns, and making predictions with high accuracy. However, their application crime detection systems remains in its infancy due to factors such as availability of data, computational burden, and need for domain-specific tuning(Butt et al., 2024). The problem is widespread in developing countries like Ethiopia where there is a high rate of crimes, and there is scarce resource availability for sophisticated surveillance and law enforcement. There is an urgent need for a robust, automated, and scalable crime detection

system that leverages the power of deep learning to analyze diverse streams of data, identify potential threats, and assist law enforcement organizations in preventing and solving crimes effectively.Thus, the research aims to address these challenges through the creation and implementation of a crime detection system based on deep learning capable of identifying suspicious activity and providing actionable intelligence to law enforcement agencies. This solution seeks to bridge the gap between existing crime detection technology and the future potential of deep learning technologies in creating safer communities.

Previous research in visual tasks with deep learning has been predominantly on image classification and not crime detection. While models such as ResNet, DenseNet, and VGG have demonstrated excellent performance in universal object classification, they cannot localize and detect crime-specific events in real-world images. These classification models are good to classify a particular object in an explicitly defined image but cannot look into patterns of crime that are critical in crime detection(Stalidis et al., 2021).

1.4 Research Questions

This research aims to explore the following questions.

RQ1. What are the key limitations of existing image classification models in crime detection, and how can object detection and frame sequence analysis be integrated to improve accuracy and efficiency?

RQ2. What specific lightweight deep learning algorithms or architectures are best suited for identifying suspicious activities and patterns of criminal behavior?

RQ3. To what extent we can improve the performance and effectiveness of the model in crime detection?

1.5 Objective of the Study

1.5.1 General Objective

The general objective of this study is to design and develop deep learning-based crime detection Model from surveillance videos.

1.5.2 Specific Objectives

To achieve the general objective of the study, the following mentioned specific objective has to be addressed:

- Conduct a literature review on deep learning-based crime detection

- Extract and preprocess important frames from existing video crime detection dataset.
- Apply lightweight crime detection deep learning model.
- Evaluate the proposed crime detection approach.

1.6 The methodology of the Study

To achieve the aforementioned objectives and address the research questions outlined earlier, the following methods and techniques were utilized

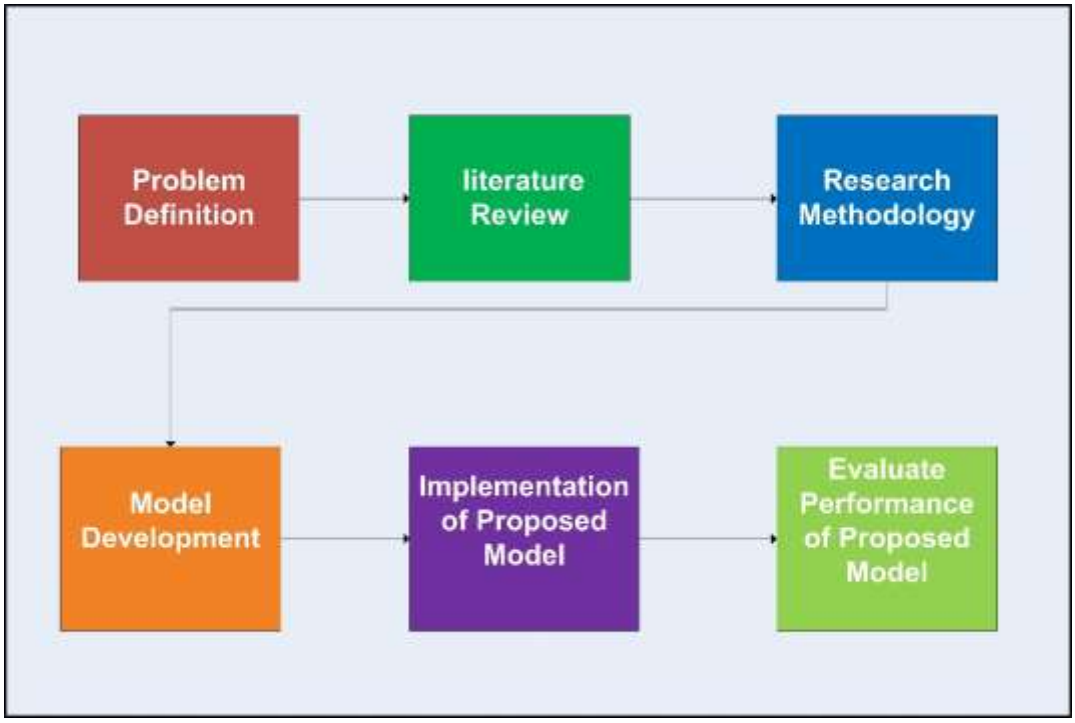


Figure 1.1: Overview of the Research Process

1.6.1 Modeling Approach

Modeling in crime detection involves the development and application of computational and statistical techniques to analyze data, identify patterns, and predict criminal activities. These models can be broadly categorized into statistical, machine learning, and deep learning approaches, each with unique applications and benefits. In this thesis, we intend to employ a deep learning model approach.

1.7 Scope and Limitations of the study

1.7.1 The scope of the study

The study focuses on designing and developing of deep learning-based crime detection system aimed at improving the identification and classification of the criminal activities. The scope includes: On this study the model will be trained video dataset that acquired from University of Central Florida that contains criminal and non-criminal videos. It employee on deep learning approach to detect crime pattern. It detects four specific classes of crimes, along with one normal class, and predicts the class for each instance.

1.7.2 The Limitations of the Study

Implementing crime detection system is more challenging and complex. Because of this fact detecting any type of data is difficult to consider and is beyond the scope of the study. The study focus only four types of crime along with normal class. On the other hand, it doesn't consider other types of crimes. and also, model cannot understand human intention.

1.8 Significance of Study

The significance of the research lies in its potential to improve public safety and more effective law enforcement through advanced deep learning techniques for surveillance video crime detection. By narrowing the attention to specific classes of crime—shooting, fighting, road accidents, and stealing the research aims to address difficult issues in the urban environment where timely and accurate detection can have a significant impact.

Enhanced Response Rates: Detection of these types of crimes in an automated way enables law enforcement to respond faster to incidents, which may help avert escalation and maintain public safety. **Resource Optimization**: With the reduction of reliance on human viewing of surveillance images, this study can lead to more efficient use of police resources so that police officers can spend more time on priority work. **Data-Driven Analysis**: Deep learning models employed encourage the assessment of patterns and trends in crime and provide policymakers and planners with appropriate analysis for the creation of crime prevention frameworks.

Improved Public Awareness: The results can be used in public awareness campaigns so that citizens will be able to grasp the spread of various crime types and why surveillance is critical to ensure security. **Foundation for Future Research**: This paper lays a foundation for future research into the application of deep learning to other aspects of public safety, potentially for

other forms of crime or integration with other technology such as drones or real-time video monitoring systems.

1.9 Organization of the Thesis

This thesis is structured into seven chapters.

Chapter one provides an overview of the central argument. It clarifies the research problems, the hypotheses formulated to address their relevance, and outlines both the general and specific objectives aimed at resolving these issues. Additionally, it presents the general research methodology employed to achieve the research goals. Chapter two serves as a literature review. It examines existing literature and sources related to the research field, supporting the research hypotheses by reviewing various studies on the problem domain. This chapter also identifies gaps in related research through a systematic literature review approach. Chapter three focuses on the research methodology, offering a detailed discussion of the methodologies implemented in the study. Chapter four introduces the proposed system for crime detection using surveillance video, aimed at enhancing crime prevention efforts. It provides a step-by-step explanation of how the proposed solution addresses the identified problems using deep learning techniques, detailing how the models function throughout this chapter. Chapter five elucidates the experiment and application of the deep learning algorithm. It explains how the deep learning model was designed and functions.

Chapter six discusses the results of the proposed solution in relation to previous results from the study. In this chapter, the performance of the model developed is evaluated, and the findings of the study conducted are presented. Chapter seven is the conclusion of the study, giving author reflections and suggestions. It also gives a summary of the contribution of this study in the conclusion section.

CHAPTER TWO

2. LITERATURE REVIEW AND RELATED WORKS

2.1 Overview of the Chapter

This chapter presents the literature review of a deep learning-based crime detection model presents a broad overview of existing research, methods, and technologies utilized for crime detection, deep learning techniques, and surveillance systems. This chapter summarizes the most critical advancements and underscores the literature gaps covered in this research.

2.2 Introduction

Object detection is currently an important field of computer vision, particularly in areas such as surveillance, autonomous vehicles, and robots(Amit et al., 2020). The evolution Object detection algorithms have evolved to be highly precise and efficient, and the YOLO (You Only Look Once) series is among the most popular frameworks to date. Detecting anomalies has never been simple. There have been different solutions developed to detect anomalies, for example, domain-oriented anomaly detection mechanisms such as violence detector and traffic accident detector(Mohammadi et al., 2016)(Sultani et al., n.d.). These solutions are effective for their intended purposes; however, their limitation lies in their inability to be applied to various other types of videos. Numerous efforts have been made to identify anomalies.

Waqas Sultani proposed a deep learning approach to detect real-world anomalies in surveillance videos(Sultani et al., n.d.). They also introduced a new large-scale first of its kind dataset of 128 hours of videos. A system has been created to differentiate between anomalous and normal events by utilizing weakly labeled videos and employing an algorithm known as multiple instance learning (MIL). This approach groups videos into bags that belong to the same category, and segments them to generate individual instances. Furthermore, they introduced scarcity and temporal smoothness constraints in the ranking loss function to better localize anomaly during training. During the training process, an anomaly ranking model is established, which assigns ranks to video instances according to their level of anomaly. Segments identified as anomalous receive higher scores, facilitating the detection of these anomalies.

J. Kim and K. Grauman detects anomalous events by implement space-time Markov random field.(Kamijo et al., 2000) . To learn the typical activity patterns at each local node, the system captures the distribution of standard optical flow using a Mixture of Probabilistic Principal

Component Analyzers. A Markov Random Field (MRF) model is created, which quantizes image features into discrete states and illustrates how these states transition over time. Unusual events are detected by comparing them against a set of normal models; if an event does not conform, it is classified as anomalous.

This method would be effective if normal activities were clear-cut and consistently defined, meaning all normal events could be labeled. However, in practice, what constitutes "normal" activity is often ambiguous and not fixed. As a result, modeling based on normal activity proves challenging, since the term "normal" is broad and encompasses a wide range of behaviors that are difficult to categorize comprehensively.

Abnormal Event Detection in Video via Motion Information Entropy The computation of frame-level motion entropy is utilized to characterize the complexity of a scene(Z. Wang et al., 2018). Based on this motion entropy, the second component determines the Gaussian distribution of the normal scene. It then applies truncated probability to identify whether a frame is abnormal. Both the direction and magnitude of information are combined to represent the scene effectively. The entropy value serves as an indicator of the scene's disorder; typically, a higher entropy value suggests a normal scene, while a lower value indicates an abnormal one.

In summary, surveillance systems have shown promising results in crime detection. However, there is still a need for research to address the challenges of robustness to environmental variations and the integration of multiple sensors for more comprehensive crime detection.

2.2.1 History of Crime detection

The discovery of crime also has an interesting history that reflects the progress of society, science, and technology based on how to discover different types of crimes at different stages. From the ancient methods that are based on intuition and superstition to the modern methods that incorporate sophisticated forensic techniques, the pursuit of the truth and justice has continued to evolve with evolving challenges and advancements.

1. **Early Developments (1980s-2010s)**
    - **Introduction of Neural Networks**: The foundation for deep learning was laid in the 1980s with the creation of artificial neural networks. Nevertheless, due to the constraint of limited computation and data at hand, applying them to the detection of crimes was not viable.

- **Machine Learning for Crime Detectio**n: In the early 2000s, machine learning methods began to be applied to crime analysis with the purpose of predictive policing and crime mapping. Techniques like decision trees, support vector machines, and clustering were being extensively applied.

2. **Emergence of Deep Learning (2010s-Present)**

The surge in deep learning's popularity began around 2012 with the advent of more powerful GPUs, large datasets, and improved algorithms. This allowed for significant advancements in image recognition, natural language processing, and anomaly detection.

2.3 Machine Learning and Deep Learning Method for Crime Detection

A variety of research studies have introduced numerous methodologies that incorporate both conventional machine learning algorithms and deep learning techniques for various applications. for crime detection. The use of machine learning and deep learning in crime detection can improve the accuracy and speed at which reported incidents with bias elements are identified. Deep learning techniques have demonstrated remarkable potential in analyzing various types of data sources, such as surveillance footage, sensor data, and social media feeds, to identify criminal activities, predict incidents, and aid law enforcement agencies in proactive responses. Previous methods for object detection, like R-CNN and its variations, used a pipeline to perform this task in multiple steps(J. Wang et al., 2024). This can be slow to run and also hard to optimize, because each individual component must be trained separately and in case of YOLO the name itself explains a lot about it, that it just goes through the entire image just once. YOLO is designed to handle a variety of vision AI tasks, including detection, segmentation, pose estimation, tracking, and classification. Its cutting-edge architecture provides exceptional speed and accuracy, making it ideal for a wide range of applications, from edge devices to cloud APIs.Developed by Joseph Redmon and Ali Farhadi at the University of Washington, YOLO (You Only Look Once) is a widely used model for object detection and image segmentation that was introduced in 2015. It quickly became popular due to its impressive speed and precision. The release of YOLOv2 in the year 2016 enhanced the original model through the incorporation of features such as batch YOLOv3, released in 2018 and with increased enhancements to the model using an efficient backbone network, more than one anchor, and spatial pyramid pooling. In 2020, YOLOv4 was released along with further improvements such as Mosaic data augmentation, new non-anchor-based detection head, and enhanced loss function. YOLOv5

continued the trend of advancement by boosting performance and introducing features like hyperparameter tuning, experiment tracking out of the box, and native export to numerous popular formats. Meituan open-sourced YOLOv6 in 2022, which is currently utilized in the majority of autonomous delivery robots within the company. YOLOv7 expanded its reach by adding tasks such as pose estimation on the COCO keypoints benchmark. Ultralytics' 2023 release of YOLOv8 introduced new functionality and features aimed at improving performance, versatility, and efficiency in a broad spectrum of vision AI tasks. YOLOv9 introduced revolutionary techniques such as Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN).

Finally, YOLOv10 was deployed by Tsinghua University researchers using the Ultralytics Python package. This release delivers real-time object detection enhancement through the utilization of an End-to-End head that eliminates Non-Maximum Suppression (NMS) needs.

YOLO11 NEW: Ultralytics' latest YOLO models performing state of the art (SOTA) levels in a broad set of tasks including detection, segmentation, pose estimation, tracking, and classification, leverage potential in a broad spectrum of AI applications and industries.

How YOLO works

YOLOv1 streamlined the object detection process by simultaneously identifying all bounding boxes. To achieve this, YOLO partitions the input image into an **S×S** grid and predicts **B** bounding boxes of the same class, along with their confidence scores for C different classes at each grid cell. Each bounding box prediction includes five components: P_c, b_x, b_y, b_h, and b_w, here, Pc represents the confidence score indicating the model's certainty that the box contains an object and the accuracy of its dimensions. The coordinates b_x and b_y specify the center of the box relative to the grid cell, while b_h and b_w denote the height and width of the box in relation to the entire image. The output of YOLO is a tensor of S×S×(B×5+C) which may be processed with non-maximum suppression (NMS) to eliminate overlapping detections. In their original paper, the authors employed the PASCAL VOC dataset, which includes 20 classes (C=20), a 7×7 grid (S=7), and a maximum of 2 classes per grid cell (B=2), resulting in an output prediction of 7×7×30 output prediction(Kuang, 2023).

2.3.1 YOLOnV8

One popular type of deep learning model for crime detection is the YOLOv8. Deep learning architectures referred to as YOLOv8 (You Only Look once, version 8) represent the latest

version in the family of object detection models of YOLO. It is better in terms of speed, accuracy, and adaptability compared to earlier versions. YOLOnv8 is designed to run real-time object detection, classification, and segmentation and is appropriately optimized for implementation on various platforms and environments such as edge devices(Huang et al., 2024). YOLOv8 is a robust deep learning model based on convolutional neural networks (CNNs).YOLOv8 is the latest in the YOLO series of real-time object detectors with state-of-the-art accuracy and speed performance. Building on the advancements of the previous versions of YOLO, YOLOv8 introduces new features and enhancements that make it the best choice for various object detection tasks in various applications(Huang et al., 2024).

The YOLOv8 model family provides a variety of models, each appropriate for specific tasks in computer vision. The models are designed to meet different needs, ranging from object detection to more complex operations such as instance segmentation, pose and keypoint detection, oriented object detection, and classification.

Each model in the YOLOv8 family is optimized for its specific task so that performance and accuracy are optimal. The models also support multiple modes of operations, including Inference, Validation, Training, and Export, which enhance their use across the various stages of deployment and development. We can train a YOLOv8 model using Python or the CLI. The model operates by passing the input image initially to the YOLOv8 architecture for feature extraction and prediction. It uses a backbone network to extract hierarchical features from the image, which is essential for accurate detection. Then, the detection head predicts the bounding boxes, object classes, and confidence of each object detected. To improve these estimates, YOLOv8 employs Non-Maximum Suppression (NMS) to suppress the redundant bounding boxes with overlap, retaining the most confident ones. Last but not least, the output is the coordinates of the detected objects, their respective class labels, and the confidence scores, giving a comprehensive representation of the scene.

2.3.1.1 Architecture of YOLOv8

It builds on the advancements in earlier models such as YOLOv5, introducing significant enhancements in its architecture and training methods. This makes it more adaptable and effective for real-time applications such as autonomous cars, surveillance, and robotics.

2.3.1.2 Core Components of YOLOv8 Architecture

1. Backbone: CSPDarknet53

The core of YOLOv8 is derived from CSPDarknet53, which is a better version of the CSPNet (Cross-Stage Partial Network). The backbone extracts fine features from input images. CSPDarknet53 divides feature maps into two halves and then merges them, giving better gradient flow and saving computational cost. This architecture excels at extracting intricate features, even for tiny objects, hence enhancing overall detection accuracy

2. Neck: Path Aggregation Network (PANet)

YOLOv8 utilizes a new neck architecture, using a C2f module to substitute the standard Feature Pyramid Network (FPN) to enhance multi-scale feature fusion. The PANet architecture allows the model to merge feature maps of various layers, enhancing its object detection capability across different sizes. This results in more precise localization and enhanced small object detection through favoring information from shallow layers.

3. Head: Multi-Detection Heads

The detection head of YOLOv8 employs Dynamic Anchor Boxes and has to predict class probabilities and bounding boxes. It uses IoU loss (Intersection over Union) for enhancing the accuracy of bounding box regression. The model predicts three different scales to ensure that the larger objects as well as the small objects are detected with equal accuracy. In addition, the YOLOv8 head uses anchor-free prediction techniques, which improve its performance in complex object detection scenarios involving occlusions or overlapping objects.

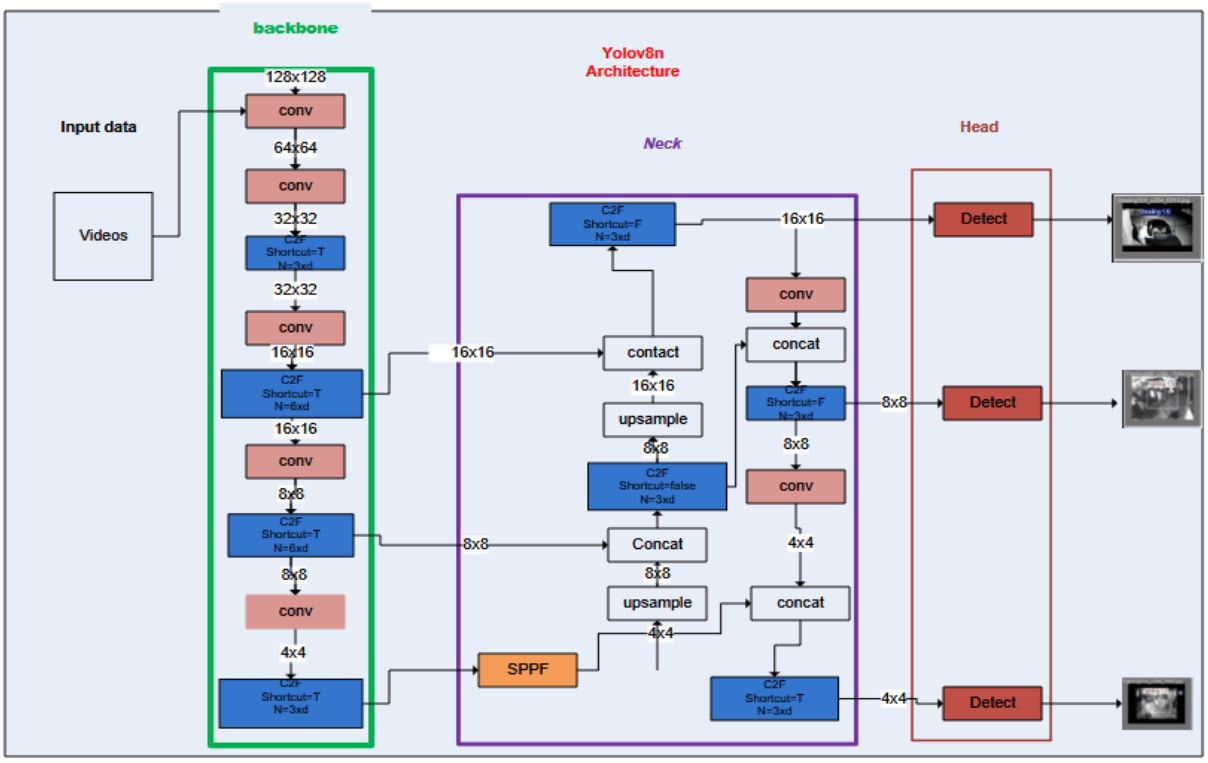


Figure 2.1. Yolov8n Architecture(C. Y. Wang & others, 2023).

2.3.2 YOLOnanoV11

YOLO11 represents the newest member of the Ultralytics YOLO series of real-time object detectors, pioneering new heights for accuracy, velocity, and resource efficiency. Benefiting from the outstanding improvements over earlier YOLO releases, YOLO11 brings powerful innovations in terms of architecture and training approaches and is a flexible option for all types of computer vision applications. YOLOv11 is an advanced model with the focus on improving accuracy, speed, and efficiency in object detection, segmentation, classification, and related tasks. YOLOv11 builds upon the success of previous versions of YOLO and features architectural improvements for better performance in various computer vision tasks(Li & Wang, 2024). Ultralytics YOLO11 boasts some important enhancements compared to its predecessors, including. **Improved Feature Extraction**: YOLO11 utilizes an improved backbone and neck model, which further improves its ability to extract features for improved object detection. **Improved Speed and Performance**: The more advanced architectural approaches and optimized training processes result in improved processing time with a nice balance between performance and accuracy. **Greater Accuracy with Smaller Parameters**: YOLO11m gains greater mean Average Precision (mAP) from other datasets despite having fewer parameters

than YOLOv8n, hence becoming computationally more efficient while not sacrificing precision. **Usage Across Different Environments**: YOLO11 may be deployed over a variety of environments ranging from edge devices to cloud infrastructures and NVIDIA GPU-supporting machines. **Broad Range of Supported Tasks**: YOLO11 supports a range of computer vision tasks including object detection, instance segmentation, image classification, pose estimation, and oriented object detection (OBB).

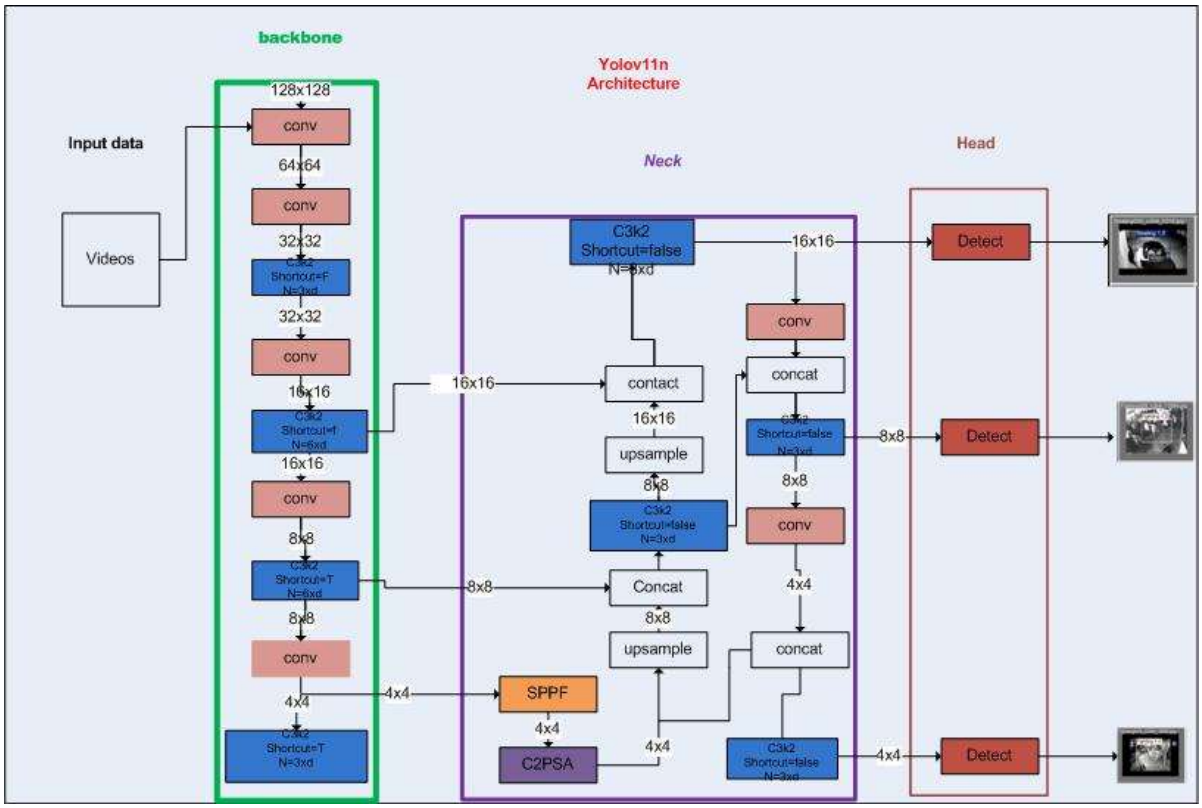


Figure 2.2 Yolov11n Architecture(Li & Wang, 2024).

The main architectural improvements in YOLOv11 are to the C3K2 block, the SPFF module, and the C2PSA block, which collectively enhance its ability to process spatial information at the cost of trading little speed for inference.

YOLOv11 Backbone Convolutional Block: The component is referred to as the Conv Block, and it processes the provided dimensions (c, h, w) through a 2D convolutional layer, followed by a 2D Batch Normalization layer, and finally a SiLU Activation Function. **Bottleneck:** This is a series of convolutional blocks that have a shortcut parameter, and the parameter will determine whether to include the residual component or not. It behaves similar to the ResNet

Block; when the shortcut is set to False, the residual won't be included. **C2F (YOLOv8):** The C2F block, or Cross Stage Partial Focus (CSP-Focus), is in accordance with the CSP network and emphasizes efficiency while trying to retain feature maps. It starts with a Conv Block, followed by splitting the output into two halves (channel division). The output of each half is passed through a series of 'n' Bottleneck layers, and their outputs are concatenated and passed through a last Conv Block. This improves the connectivity of feature maps and eliminates redundant information. **C3K2:** In YOLOv11, blocks C3K2 are utilized for feature extraction at various levels of the backbone. Smaller 3x3 kernel usage optimizes computational cost without compromising the capability of the model to extract relevant image features. The fulcrum of YOLOv11's backbone, the C3K2 block, is the better alternative of the CSP (Cross Stage Partial) bottleneck in earlier versions. This block optimizes information flow throughout the network by dividing the feature map and using a series of small kernel convolutions (3x3), both more efficient and faster than big kernels. Processing small, individual feature maps and then combining them after several convolutions, the C3K2 block enhances feature representation with fewer parameters compared to the C2F blocks in YOLOv8(Sohan et al., 2024).

C3K block and the C2F block both share the same architecture but don't have splitting involved. The input is routed through a convolutional block first, followed by 'n' consecutive bottleneck layers with concatenation, and one last convolutional block. Information is processed using the C3K block within the C3K2 block, which starts and ends in two convolutional blocks with middle portions consisting of multiple C3K blocks placed in between them. Finally, it adds the convolutional blocks output and the last C3K block before stopping at a terminal convolutional block. This pattern attempts to reach an equilibrium in speed and in precision by using the CSP pattern to its advantages

Neck: Spatial Pyramid Pooling Fast (SPPF) and Up sampling: YOLOv11 adds the SPPF module (Spatial Pyramid Pooling Fast) to pool features from various regions of an image at multiple scales. This significantly enhances the network's capability to detect objects of varying sizes, with a particular improvement in detecting small objects, which has been challenging in previous versions of YOLO. The SPPF module utilizes several max-pooling operations with different kernel sizes to combine multi-scale context information. In this manner, it is ensured that small objects are also detected well by the model as it aggregates information from different

resolutions. With the addition of the SPPF, YOLOv11 achieves a balance between maintaining real-time processing rates and improving its object detection on diverse scales.

Attention Mechanisms: C2PSA Block Among the major upgrades of YOLOv11 is the incorporation of the C2PSA block (Cross Stage Partial with Spatial Attention). The block incorporates attention mechanisms to enable the model to pay greater attention to meaningful areas of an image, particularly small or occluded objects, by highlighting spatial importance in the feature maps. **Position-Sensitive Attention:** This class provides the functionality for using position-sensitive attention and feed-forward networks on input tensors, enhancing feature extraction and processing. The layer passes the input through an attention layer and concatenates it with the input. This combined output is fed through a Feed Forward Neural Network, followed by a convolutional block, and lastly a convolutional block without an activation function. Finally, the output of the convolutional block is concatenated with the output of the first concatenation layer. **C2PSA:** The C2PSA block consists of two Partial Spatial Attention (PSA) modules, each operating on a different branch of the feature map before concatenation, as in the C2F block. This is carried out to allow the model to pay attention to spatial information while balancing the detection accuracy and computational cost. By using spatial attention over the learned features, the C2PSA block enhances the model's capability of selectively concentrating on areas of interest. As a result, YOLOv11 can demonstrate superior performance to earlier models, e.g., YOLOv8, particularly in applications that require the detection of fine-grained object details with high accuracy. **Head: Detection and Multi-Scale Predictions:** Like its predecessors, YOLOv11 employs a multi-scale prediction head to identify objects of varying sizes. The head generates detection boxes for three scales: low, medium, and high, depending on the feature maps produced by the backbone and neck.

The detection head leverages the output of three feature maps, typically referred to as P3, P4, and P5, corresponding to different scales of image description. The above mechanism enables detection at lower scales for small objects with a greater accuracy in the P3 feature map and in upper scales using the high-order features contained in the P5 feature map.

This study would be conducted on the following deep learning models. As summarized in the Table 2.1, the main reason we used Yolo8 and Yolo 11 in our work are mainly computation requirement and the task application areas. Whereas Yolo8 and Yolo11 are lightweight and suited for detection tasks, the yolo9 Nano version even if lightweight it requires original image,

ground truth image, and bounding box coordinates in which we have no ground truth image for UCF crime dataset. Moreover, Yolo10 Nano is more trainable parameter compared to Yolo11. Therefore, we selected Yolo8 and Yolo11 based on the nature of the dataset(C. Y. Wang & others, 2023)(Chen & others, 2023)(Zhang & Liu, 2024)(Li & Wang, 2024).

Table 2.1. Comparison of different YOLO versions

Feature	YOLO 8 Nano	YOLO 9 Nano	YOLO 10 Nano	YOLO 11 Nano
No of Params	~3 million	~4 million	~5.5 million	~5.4m
mAP	~40% on COCO dataset	~42% on COCO dataset	~44% on COCO dataset	~46% on COCO dataset
Model complexity	Very Lightweight	Lightweight	Moderate	Lightweight
Cases	Ideal for edge devices	Fast detection in constrained environments	Efficient for real-time applications	Fast and efficient for small devices

2.4 Related Works

To achieve the goals of this research, conducting a systematic literature review on related works is essential. Numerous studies have been carried out focusing on deep learning techniques for crime detection in surveillance footage. This section highlights research that is particularly relevant to the current study:

Table 2.2: Summary of Related Work

Author	**Research Title**	**Detection Method**	**Evaluation Metrics**	**Research Gaps**
Ayush Thakur (2024)	Real-Time Weapon Detection Using YOLOv8 for	YOLOv8	Precision, Recall, F1-score mAP	✓ Limited dataset size, generalizability to different types of crimes ✓ Overfitting is noted

	Enhanced Safety			
Waqas Sultani1,(2018)	Real-world Anomaly Detection in Surveillance Videos	Deep Multiple Instance Learning (MIL)	Operating Characteristic curve (AUC-ROC)	✓ Method extract feature is C3D it contains external kernel dimension it causes of increase Parameter which make more difficult training Data. ✓ Its heavy weight model requires high computing resource
Shreyash Chole (2023)	"Deep Learning for Suspicious Activity Detection"	3D CNN	Precision, Sensitivity	✓ Difficulty in handling complex and overlapping activities
Muhammad Tahir Bhatti (2021)	"Weapon Detection in Real-Time CCTV Videos"	SSD-MobileNetV1, YOLOv3, Faster RCNN	mAP (mean Average Precision) Recall F1 score	✓ Higher computational cost ✓ Lack of performance
Abdirahman Osman (2023)	Deep Learning Models for Crime Intention Detection	YOLOv6, Faster R-CNN, VGG-19, ResNet50, Google Net	Accuracy: - mAP@0.5:	✓ Higher computational cost. ✓ Lack efficiently and accuracy of object detection

	Using Object Detection			

Research Gap Summary

Previous works primarily deal with crime classification rather than localization and object detection at crime scenes. Classification was achieved primarily through models such as ResNet, DenseNet, and VGG with no ability to detect particular objects or activities within a video or an image(Piza et al., 2019). Some of the detection models such as SSD(Sohan et al., 2024) and F-CNN are bulky frameworks with a tremendous requirement for computational resources, and this increases the training time, decreases efficiency, and decreases accuracy in crime detection systems that operate in real-time(Piza et al., 2019).

Our research enhances these limitations by proposing an optimized YOLO-based system of detection, which classifies and detects objects simultaneously. The system enhances detection, minimizes computational complexity, and ensures efficient performance and is therefore more appropriate for crime monitoring and surveillance.

I am employing the YOLO Nano model, a deep learning lightweight architecture, to address the gigantic shortage of high compute resource needs in crime detection systems. YOLO Nano is specifically designed to deliver efficient performance with reduced memory and processing capacity, suitable for deployment in resource-constrained environments, such as characteristically provided in developing countries(Butt et al., 2024) (Stalidis et al., 2021). This model provides object detection without sacrificing accuracy, hence allowing for timely response to potential threats without the need for expensive hardware upgrades. Through the inclusion of YOLO Nano, I can seamlessly implement a robust crime detection solution that relies on existing infrastructure, ultimately enhancing public security and law enforcement capabilities without the need for extensive computational burdens.

CHAPTER THREE

3. METHODOLOGY

3.1 Overview of Chapter

This chapter describes the methodology used in the study consisting of the methods, tools, and techniques employed to accomplish the research objectives. It refers to some phases like data collection, data preprocessing, feature extraction, model training, and model evaluation.

3.2 Dataset

Information plays a pivotal role in training most machine learning models, particularly the ones employed in crime detection across all regions. In detecting crime through video recorded by surveillance cameras, acquiring the necessary video information is a central process of data preparation for conducting the research on a good dataset. One of the primary challenges in crime detection and classification is the lack of large video data sets. For this reason, researchers tend to perform targeted data collection operations aligned with their requirements for crime detection. This is achieved by obtaining data from a number of sources, including sensors, other organizations, and other applicable data providers. For this study, we employed a dataset that is committed to the detection of crimes using deep learning models on video surveillance. The dataset is made up of various video recordings capturing criminal and non-criminal acts. The classes of videos consist of activities such as shooting, fighting, road accidents, Stealing, and everyday activities. The video samples were collected from publicly available sources of UCF and comprises various contents of videos representing 13 class of crimes(Sultani et al., n.d.).

Table 3.1 Dataset Classification

Class Name	No of Videos	Description
Fighting	50	A physical altercation between two individuals, possibly involving aggression or violence.
Shooting	50	The act of discharging a firearm at a person or target, usually within the context of violence.
Normal videos	150	Everyday activities or actions that are not of a violent or criminal nature, showing normal life scenarios.

Road Accident	150	A unexpected event on the road involving motor vehicles, resulting in damage, injury, or death.
Stealing	100	The unlawful act of taking another's property without their approval, with an aim to deprive them of it forever.
Total	500 videos	

3.2.1 Dataset Preparation

Data preparations are fundamental in deep learning-based crime detection systems to feed the model with clean, relevant, and structured data. The initial step involves data cleaning to remove irrelevant or redundant data from the video dataset. As the overall dataset of crime can be large and resource-demanding, our first task is to conduct benchmarking for choosing a representative subset to train and test the model. This step entails selecting a subset of the data that retains the most significant features and patterns of the larger dataset but is computationally manageable for model development and testing.

By drawing a sample from the whole dataset, we can ensure that the chosen subset is representative of the complexity and diversity of the entire dataset. The subset is a representative sample that can effectively be used to develop, train, and test the crime detection model.

3.2.1.1 Frame Extraction

This process enables the frames to be read from the input video file, effectively decomposing the video content into an exhaustive sequence of individual images. The process begins by reading the class ID of the input action and determining whether the class name exists in a predefined map. To avoid any redundant processing, the method initially looks for the existence in a log file to verify that the video has not already been processed. OpenCV is then used to read the video, looping over its frames one by one. On each nth frame—calculated based on a frame rate of 24 frames per second—the image is written as a JPEG file to the specified output directory, with the file name built up from the video title and frame number. This systematic procedure assists in creating a properly formatted dataset from raw video data, which can then be used for further analysis or training.

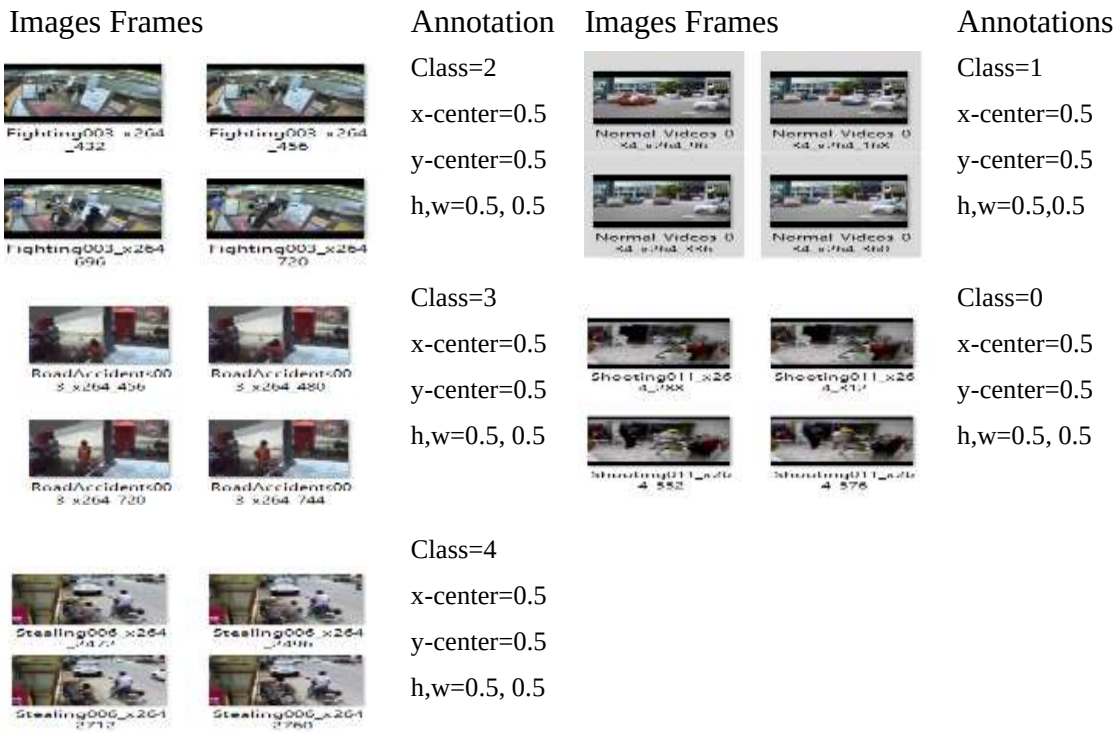


Figure 3.1 Sample video frames from five classes

3.2.1.2 Annotation Preparation

Annotation refers to the process of labeling images with metadata, specifically marking regions of interest (e.g., objects) in an image and associating them with their respective categories. This is a critical step in preparing datasets for training YOLO or any object detection model.

In our suggested approach, the preparation of annotation is necessary in order to generate files that recognize actions from videos. For every frame kept in the specified label output directory, a text file is created with the filename derived according to the video title and frame number. Every annotation file contains the class ID, which defines the action that is depicted in the frame, and placeholder values of 0.5 for fixed bounding box coordinate dimensions. The placeholders are merely simple annotations of the occurrence of the defined action class, which is vital for machine learning model training. By continuously creating these annotation files alongside the frames extracted, we create an organized system that enables video content to be more understandable in terms of detected actions, and therefore action recognition tasks more effective.

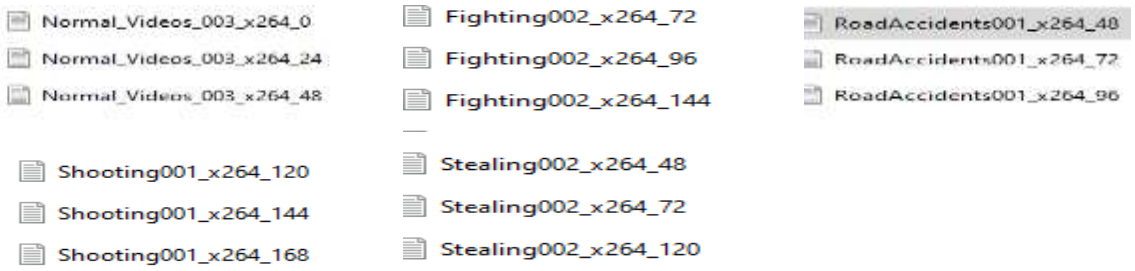


Figure 3.2 Annotation files

3.2.1.3 Split Data

Split the dataset into training, and validation sets, ensuring that each set has a balanced distribution of classes to avoid bias. Our proposed methodology for splitting the data into training and validation sets is to achieve optimum model evaluation efficiency. It is usually an 80-20 division, where 80% is used for training and 20% for validation. We begin by gathering all the JPEG images and their equivalent label files in order to match them precisely. These images and labels are then put into tuples and shuffled to randomize the distribution of the data, hence reducing the possibility of bias in the training process. A split index is determined from the validation ratio given, resulting in the formation of separate directories for validation images, validation labels, training images, and training labels. If these folders don't already exist, they are created and the target files are moved from temporary storage to their permanent locations. The operation ends in a cleanup step that deletes any temporary directories, leaving a tidy dataset prepared for training and validation, and ultimately model evaluation and performance monitoring.

3.2.2 Data Preprocessing

After preparing the dataset for crime detection, the second most important step is data preprocessing. Data preprocessing consists of several tasks that are performed to prepare the dataset in the correct format for model training.

processing video and its labels is a fundamental process in preparing data for machine learning tasks, particularly video classification tasks. The purpose of the process is to convert raw video data into an organized format which can be efficiently utilized to train a deep learning model.

Resize

The Resize transformation is the initial operation in the preprocessing pipeline, wherein the images are resized to a standard size of 128x128 pixels. This makes all input images the same

size, which is extremely important for batch processing in neural networks since they need fixed input sizes. Resizing reduces the computational expense and memory and thereby makes the training process faster. Additionally, normalizing the image size allows the model to focus on learning features rather than adjusting to various sizes, which allows it to generalize better on new data.

Random Horizontal Flip

Random Horizontal Flip is a data augmentation strategy in which images are horizontally flipped randomly during training. Randomness adds variety to the training dataset, and the model becomes capable of identifying objects irrespective of their orientation. Exposing the model to both flipped and original versions of the same image makes the model invariant to changes in real-world environments, thereby enhancing its performance and preventing overfitting. This method is particularly helpful in such tasks in which the orientation of the object will not affect its classification, such as in identifying crimes.

Color Jitter

Color Jitter transformation introduces random variations in images' brightness, contrast, saturation, and color with given range values for each parameter. The below example utilizes brightness and contrast of 0.2, saturation of 0.2, and hue of 0.1. This data augmentation simulates various lighting conditions and color variations the model may face in real environments. By training from images with altered color features, the model learns to be invariant to such transformations, enabling it to better recognize objects regardless of environmental conditions. This is particularly important in crime scene investigation, where light and color can be significantly different under different scenarios.

To Tensor

The To Tensor () transformation is used to convert the image data from PIL image or NumPy array format to PyTorch tensor. This is necessary since PyTorch models need the input data in tensor format for batched and optimized processing. The transformation also normalizes the pixel values, rescaling them from the range [0, 255] to [0, 1]. This scaling is necessary to enable the model to handle the data appropriately so that there are no vanishing or exploding gradients during training. Consequently, the model learns better from the input data.

Normalize

The Normalize transformation rescales the image pixel values of the tensors based on given mean and standard deviation values ([0.485, 0.456, 0.406] for mean and [0.229, 0.224, 0.225] for standard deviation). The normalization here is to make the input data have a zero mean and unit variance, which helps to stabilize the training process. Scaling the data normalizes it, which helps the model converge more quickly and can enhance overall performance by placing all input features on the same scale. It is especially helpful when training deep learning models since it reduces the effect of outliers and allows the model to learn the patterns in the data more effectively.

3.3 Feature Extraction

We used spatial feature extraction methods used the YOLO backbone model, that are specifically intended for detecting and analyzing visual patterns, objects, and structures in individual frames of a video. It is an initial step towards understanding the objects and context of surveillance videos. These features form the foundation for several computer vision tasks such as anomaly detection, scene understanding, and object detection.

3.4 Model Development

Once the crime detection dataset is ready and preprocessed, the next step is to train the detection models. In this research, we have tested two different models:

1. **Yolov8n (You Look Only Once):**

Yolov8n is a deep learning framework often applied in object detection. To conduct the task of crime detection, the model can be tuned to process video data for varied purposes at varying locations. The model obtains relevant features from the inputted video and performs pooling layers for reducing the data. The Yolov8n model has also reported good performance in detecting crimes across all classes.

2. **Yolov11n**

YOLOv11n brings with it a range of powerful capabilities and optimizations to allow it to be faster, more precise, and highly flexible. YOLOv11n is used in many computer vision tasks from real-time object detection to classification, taking developers and researchers to an extent previously unforeseen. Among the most notable improvements include enhanced feature extraction to effectively capture more detailed information, increased accuracy when using fewer parameters, and speed of computation improved with much to give in enhancing real-time ability. In this thesis, we'll take a closer look at the features that make YOLOv11n stand

out and how it can transform your computer vision applications. YOLOv11n attains improved accuracy with fewer parameters due to the development in model architecture and optimization methods by comparing these two models, the study will compare their effectiveness in crime detection. The two models both possess a distinct architecture and apply distinct techniques in the detection of patterns and relationships within the data.

3.5 Model Evaluation

Model evaluation is an important process to ascertain the performance and efficiency of crime detection systems. It allows us to calculate the model quality and ensure its correct functioning. We used the k-fold cross-validation technique in this study for model evaluation.

3.5.1 Evaluation Metrics

Evaluation measures are applied to check the quality of the model, and they also help us identify whether the model is performing correctly and efficiently. The evaluation was done by training the YOLOv8n and YOLOv11n model on the preprocessed dataset. The performance was checked using the following benchmark metrics

1. **Precision**

Precision is a metric that measures how often a machine learning model correctly predicts the positive class. Measures how many detected crimes were correct.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

2. **Recall**

Recall is a **performance metric** used in classification tasks to measure the ability of a model to identify relevant instances among all actual positive instances. Measures the model's ability to identify all crimes.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

3. **Mean Average Precision (mAP)**

Assesses the balance between precision and recall for all identified objects across various Intersection over Union (IoU) thresholds.

   a. **mAP@50 and mAP@[0.5:0.95]**

IoU threshold = 0.5 (used in early YOLO versions). Measures how well predicted bounding boxes overlap with ground truth. Evaluates across multiple IoU thresholds (0.5 to 0.95, in steps of 0.05).

Stricter metric; used in modern models like YOLOv8.

$$\text{mAP} = \frac{1}{C}\sum_{c=1}^{C} Apc \quad (3)$$

where C is the total number of classes, and A Pc is the Average Precision for class c.

4. **Confusion matrix**

Confusion matrix can be a valuable tool for evaluating YOLO models like YOLOv8n and YOLOv11n, particularly for analyzing classification performance and understanding errors at a granular level. However, it should be used alongside other object detection-specific metrics like mAP, IoU, and precision-recall curves for comprehensive evaluation. A confusion matrix is a performance measurement tool that summarizes the results of a classification task by showing:

True Positives (TP): Correctly identified instances. **True Negatives (TN):** Correctly rejected non-instances. **False Positives (FP):** Incorrectly identified instances. **False Negatives (FN):** Missed instances

3.6 Development Tools

3.6.1 Design Tools

Design concepts are created, presented, and interpreted using design tools. Enterprise Architect is the main design tool that is used to design the system. It is used to create flowcharts, network diagrams, and other design concepts which are professional-looking and it is lightweight, and powerful.

3.6.2 Hardware Tools

The hardware tools listed below will be utilized to carry out this research.

Table 3.2 : Hard ware Tools used for proposed

N0.	Tools	Specifications	Use
1	GPU	Nvidia TX	To enhance computational efficiency and accelerate the training process.
2	Solid state Disk	NV Me SSD 256 GB	Serves as a repository for large datasets, enabling quicker computations during training.
3	RAM	16 GB Kingstone	To improve computational efficiency and speed up the training process.

CHAPTER FOUR

4. PROPOSED MODEL ARCHITECTURE

4.1 Chapter overview

This chapter presents the architecture of the proposed solution for crime detection using deep learning, along with relevant mathematical equations and diagrams. It is organized into four sections: the first section addresses the problem formulation, the second outlines the model architecture, the third discusses the dataset, the fourth covers various preprocessing techniques and their implementation, and the final section details the implementation of the proposed model.

For this study we have selected two models after a careful consideration of various parameters for crime detection and classification.

4.2 Problem Formulation

The mathematical model detailed here describes a crime detection system based on YOLO, aimed at identifying and classifying four specific criminal activities: stealing, fighting, road accidents, and shooting. The architecture of the Model includes a Backbone Network, Neck Network, and Head Network, reflecting the YOLO model's design to facilitate effective feature extraction and potential crime detection. The model's evaluation relies on performance metrics such as Mean Average Precision (mAP), precision, and recall, highlighting its efficacy and applicability in real-world situations.

The mathematical model is outlined as follows:

C = {stealing, fighting, road accident, shooting, Normal}

- **Model Parameters:**
  - **Backbone Network (x):**
    - Extracts high-level features from input images and video frames.
  - **Neck Network (x):**
    - Creates feature pyramids at various scales.
  - **Head Network (x), (x):**
    - Predicts bounding boxes ($b(x)$) and class probabilities in five dimensions ($c(x)$).

The prediction equation is as follows:

$$b(x), c(x) = fhead\big(P(x)\big) - (4)$$

Where:

- b(x): Represents predicted bounding boxes for input.
- c(x): Indicates predicted bounding boxes for input, with each element corresponding to the probability of a specific crime class $C1, 2\ldots, C5$.

The function $fhead$ processes the features generated by the Neck Network $P(x)$ to provide predictions for bounding boxes and class probabilities.

Loss Function:

- **Smooth L1 Loss:**
  - Aims to minimize the difference between the predicted and actual bounding boxes.
  - Promotes precise object localization within video frames.

Mathematically, this is expressed as:

$$(b, c, cgt) = \sum 4i = 1\ BCE(ci, cigt) + SmothL1(b, bgt) - - - - - - - - - - - - - - - - (5)$$

Equation 1 function loss

Where:

- b: Predicted bounding boxes
- c: Predicted class probabilities (5-dimensional vector)
- bgt: Ground truth bounding boxes
- cgt: Ground truth bounding boxes (4-dimensional vector)
- ($cgtcigt$): Cross-Entropy loss for class ci

Relationships and Decision Rule:

- The Backbone Network extracts high-level features from input video frames, which are subsequently forwarded to the Neck Network for hierarchical feature extraction. This procedure establishes a hierarchy of features to capture significant representations

4.3 The Proposed Architecture

The main goal of this research is to create a deep learning model for crime detection. As mentioned earlier, we will test two deep learning models utilizing the benchmark subset dataset. The figure below illustrates the high-level architecture of the proposed solution.

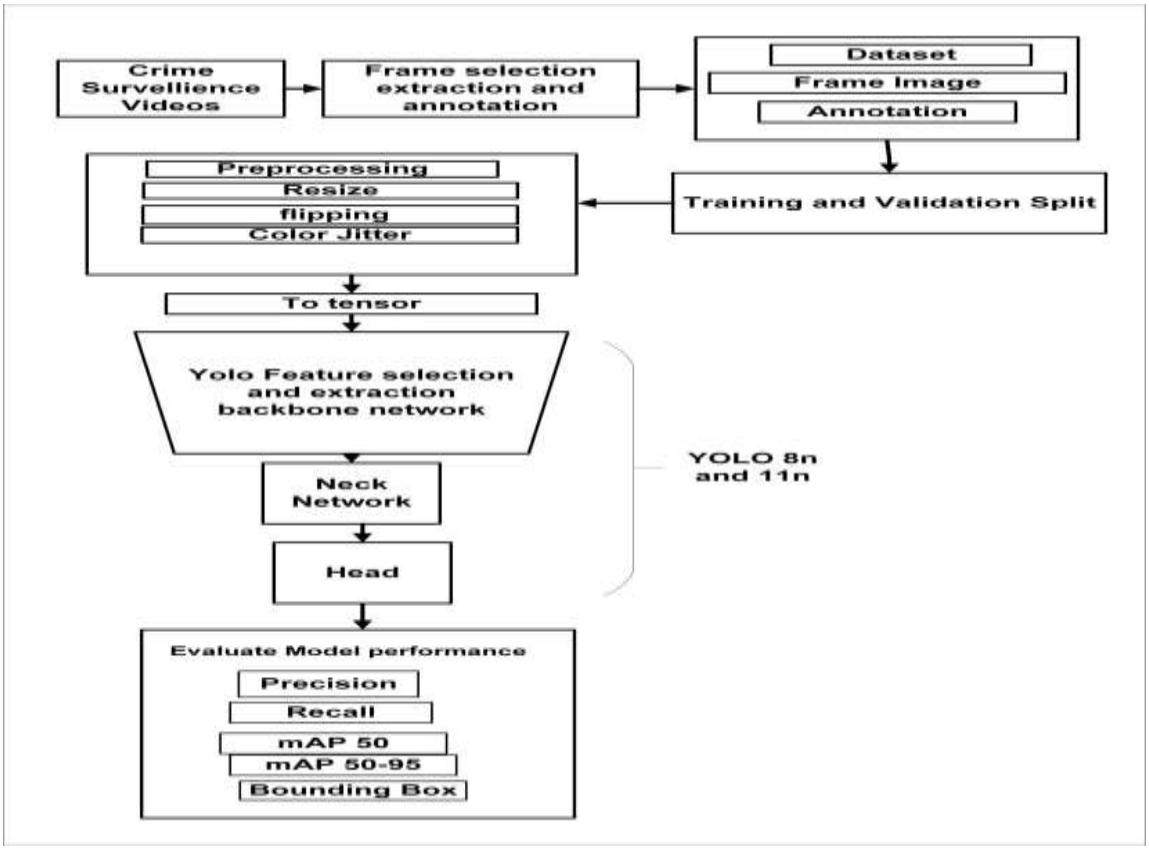


Figure 4.1 : Proposed Framework for Crime Detection

4.4 Proposed Model Architectures

In this study, we have chosen two models after thoroughly evaluating a range of options for crime detection and classification. The research will be carried out using the following deep learning models.

We proposed YOLOv11n as our preferred choice among the other model which we experimented on for various reasons. We selected YOLOv11n as our preferred model from the other options we tested for several reasons. Yolov11n address the limitations of YOLOv8n, and it is more suited for crime detection. These enhancements could include better feature extraction, improved handling of imbalanced datasets, and robust performance under diverse environmental conditions all of which are critical for detecting crimes in real-world surveillance scenarios.

4.4.1 YOLOv8n Model Architecture

YOLO (You Only Look Once) is a state-of-the-art object detection model that achieves real-time performance with high accuracy. The architecture follows a one-stage detection pipeline,

combining feature extraction and object localization into a single neural network. This document details the YOLO architecture, including its **Backbone, Neck, and Head** components.

Backbone Structure:

1. Input Image (e.g., 128x128 RGB)
2. Initial Convolution: Conv2d(3, 16, kernel size=3, stride=2)
3. Series of Convolution and C2f Blocks:
    - Conv (16->32), C2f (32->32), Conv (32->64), C2f (64->64)
    - Conv (64->128), C2f (128->128), Conv (128->256), C2f (256->256)

Neck (Feature Merging & Up sampling)

1. Up sample: Scales 256x16x16 → 128x32x32
2. Concat with earlier feature maps
3. C2f for feature refinement
4. Up sample again to 64x64x64

Detection Head

- Processes refined feature maps through convolution layers.
- Outputs bounding boxes and class predictions at three different scales.

4.4.2 YOLOv11n Model Architecture

YOLO (You Only Look Once) is an object detection framework that utilizes deep learning techniques, specifically created for object detection performance.

. The architecture described in this document builds upon previous YOLO versions with enhanced backbone and feature extraction mechanisms, incorporating **C3k2, C3k, SPPF, and C2PSA** modules for optimized object detection.

Backbone (Feature Extraction Layer)

1. Conv Layer 1:

Input: 3x128x128 (RGB image), Output: 16x64x64 (Feature maps reduced by factor of 2)

2. Conv Layer 2: Output: 32x32x32
3. C3k2 (Feature Aggregation) Output: 64x32x32
4. Conv Layer 3: Output: 64x16x16
5. C3k2 (Deeper Feature Learning) Output: **128x16x16**
6. SPPF (Multi-scale Feature Pooling) Output: 256x16x16

7. C2PSA (Attention Mechanism)
    - Output: 256x16x16

Neck (Feature Merging & Up sampling)

1. Up sample: Scales 256x16x16 → 128x32x32
2. Concat with earlier feature maps
3. C3k2 for feature refinement
4. Up sample again to 64x64x64

Detection Head

- Processes refined feature maps through convolution layers.
- Outputs bounding boxes and class predictions at three different scales.

4.5 Loss Function of the Proposed Model

4.5.1 Smooth L1 Loss

In our proposed model, we employed Smooth L1 Loss as the loss function that widely used loss function for object detection tasks, for bounding box regression. It combines the advantages of L1 loss (mean absolute error) and L2 loss (mean squared error) by being less sensitive to outliers while still penalizing large errors.

CHAPTER FIVE

5. IMPLEMENTATION OF THE PROPOSED MODEL ARCHITECTURE

5.1 Chapter Overview

This chapter focuses on the implementation of the proposed model architecture. It covers the working environment, the implementation of the YOLOV8n and YOLOv11n models, and the experiments conducted

5.2 Working Environment

This section outlines the hardware setup used for implementing the modeling process, including the specifications of the components:

- **Laptop**: Used for dataset preprocessing and model design
    - Operating System: Windows 11
    - Processor: Intel® Core™ i5-6300HQ CPU @ 2.50GHz
    - RAM: 8GB
    - System Type: 64-bit operating system, x64-based processor
    - Graphics Card: Nvidia GTX 4GB

5.3 Environment Setup

For our study, we utilized specific programming environments and IDEs:

- **JupyterLab**: An open-source, web-based interactive development environment designed for data science, machine learning, and scientific computing. It offers a flexible and extensible platform with a user-friendly interface for working with Jupyter notebooks, code editors, terminals, and other interactive tools.
- **Colab**: A cloud-based notebook from Google that enables users to write Python code in any browser, providing access to powerful computing resources (including TPU and 12GB RAM) for training with TPU and GPU capabilities.

5.3.1 Tools and Packages

This section outlines the tools and Python packages utilized in the modeling process, including their specifications. The libraries are listed along with their versions and descriptions.

Table 5.1 Package Utilized study

Libraries	**Description**
Numpy	A robust library in Python for numerical computing, Numpy supports large, multi-dimensional arrays and matrices, offering a variety of mathematical functions for efficient operations on these arrays.
Pytorch	PyTorch is an open-source machine learning library for Python, designed for flexible and efficient experimentation with deep learning models.
Sklearn	Sklearn facilitates the development and deployment of machine learning solutions, making it a popular choice in academia, research, and industry.
Pandas	Pandas provides powerful data structures, such as DataFrames and Series, enabling efficient handling and manipulation of structured data.
Seaborn	Seaborn streamlines the creation of various plots, including scatter plots, bar plots, box plots, and heatmaps.
Matplotlib	Matplotlib is a widely-used library in Python for generating visualizations and plots.

5.4 Training Procedure

The training procedure encompassed several steps to meet the objectives of this research:

- Data Preparation
- Data Preprocessing
- Designing the Proposed Solution
- Training the Model

Each phase is described in detail to offer additional insight into the actions taken.

5.5 Dataset Description and Loading of Dataset

Our preprocessing method utilizes a dataset comprising 500 videos, which feature four categories of crimes alongside normal videos.

5.5.1 Dataset Loading

To train our model, we must transfer the dataset from our local machines to Jupiter Notebook.500 videos extracted to 74552 videos frames and split into train 59452 videos frame and validation 15100 videos frame. This dataset is classified based on different types of crimes.

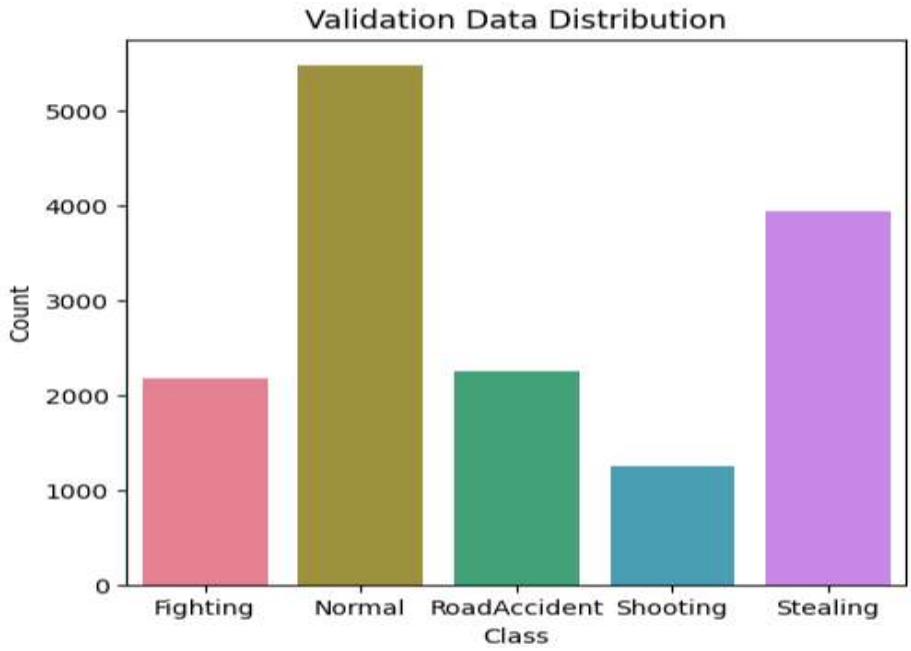


Figure 5.1 Validation Data Distribution

From figure 5.1 the y-axis "Count" in the graph is the number of instances for each class in the validation set, normally between 0 and 5000. The height of each bar is the number of validation instances of the class.The x-axis, which is titled "Class," represents the various classes being validated, such as "Fighting," "Normal," "RoadAccident," "Shooting," and "Stealing." Each class is shown as a bar, and x-axis positions are utilized to denote the particular category.

Overall, the graph illustrates the distribution of validation data among different classes, showing the makeup of the validation set and revealing any potential imbalances or biases the model will face while testing.

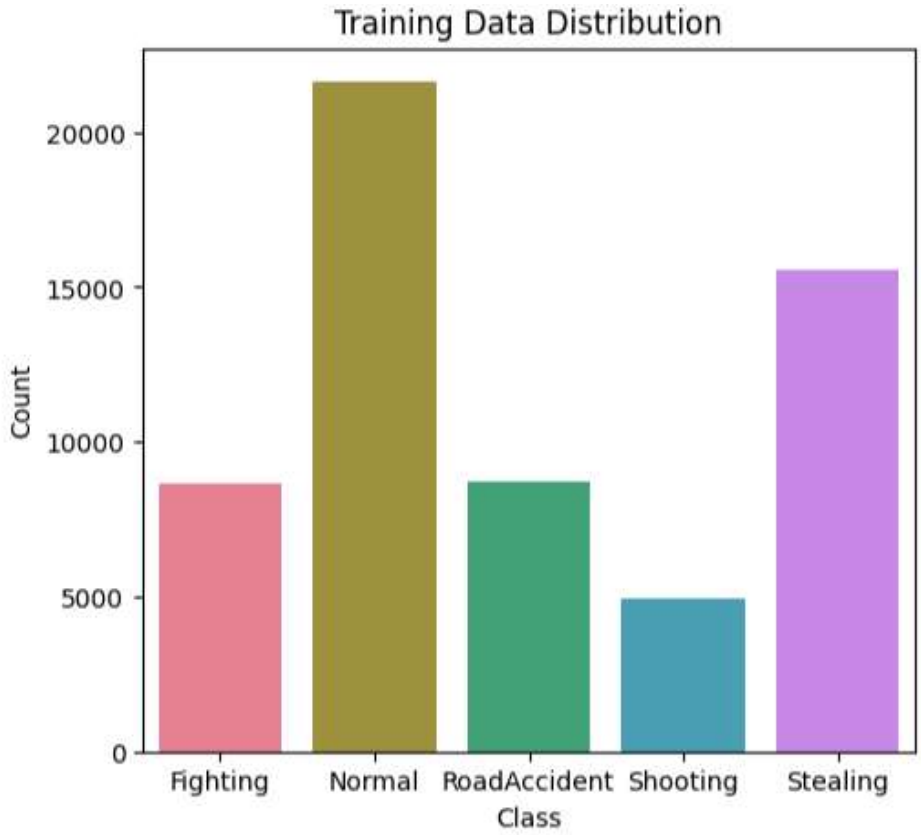


Figure 5.2 Training Data Distribution

The y-axis in Figure 5.2 is labeled as "Count," representing the quantity of samples or instances of each class available in the training data. The axis shows the availability of samples for each class, and the prevalence of different classes of crime can be compared visually. The y-axis is from 0 to 20000, and the height of each bar is the quantity of samples within that particular class.The x-axis is labeled as "Class," and the various categories are represented as being under analysis, such as "Fighting," "Normal," "Road Accident," "Shooting," and "Stealing." There is a bar for each class, and the x-axis position denotes the particular crime or activity being represented. The spread of the bars gives a visual cue regarding the number of training samples per class, and it is easy to spot any skews in the data that would affect model training and performance.

5.5.2 Model Hyperparameters

Hyperparameter tuning is an important process in developing machine learning models since it affects their performance and ability to generalize to a great extent. Each hyperparameter plays a particular role in making the overall behavior and performance of the model improved.

AdamW Optimizer

AdamW optimizer is utilized in the YOLO architecture to improve object detection training. AdamW modifies the original Adam optimizer by decoupling weight decay from the gradient update so that regularization is improved. The architecture initializes the learning rate (lr0) as 0.01, with weight decay as 0.0005, which reduces overfitting and offers good generalization ability. The optimizer also employs momentum, set at 0.937, to further accelerate convergence by dampening the updates. A warm-up approach is also employed, where warmup_epochs is set at 3.0, where the learning rate is ramped up slowly from a lower initial value to stabilize training. All these components in combination allow for effective optimization of the model, with good learning and effective performance in detecting objects of various classes.

SiLU Activation

The SiLU (Sigmoid Linear Unit) activation function, also known as the Swish function, is utilized in the YOLO architecture to add non-linearity to the model. SiLU is mathematically defined as

$f(x)=x\cdot\text{sigmoid}(x)$, incorporating the properties of both linear and non-linear functions. The activation function has proved superior to conventional activations such as ReLU in the majority of deep learning tasks, especially convolutional neural networks. Its smooth slope assists in reducing problems such as the vanishing gradient problem, which enables the model to learn better during training. SiLU implementation in the YOLO model improves the model capacity in learning intricate patterns within data, which leads to better object detection performance.

BatchNorm2d

BatchNorm2d is a layer normalization utilized in the YOLO architecture for enhancing the stability and effectiveness of training. It scales and shifts the output of the last layer by normalizing it, hence opposing complications such as internal covariate shift. Within this architecture, BatchNorm2d has an epsilon value of 0.001 and a momentum of 0.03. The epsilon value provides numerical stability for the division operation, and the momentum value regulates the running averages of the mean and variance. BatchNorm2d makes higher learning rates possible and makes the training process less sensitive to weight initialization by normalizing the input to each layer, leading to an efficient training process and better object detection performance.

The following table summarizes the significance of each hyperparameter used in our models and its possible impacts on training and performance

Table 5.2 Hyperparameter

Name	Values	Description
Batch Size	128	This hyperparameter controls the number of samples which are handled by the model per training iteration. Having a batch size of 128 implies that the model updates its parameters after handling 128 samples.
EPOCHS	30	This hyperparameter defines how many times the whole dataset will be exposed to the model while training. In 30 epochs, the model will expose the whole dataset 30 times..
Input Shape	128	This hyperparameter specifies the size of blocks in our model. It is provided as (3, 128, 128), which indicates that a block consists of 3 channels with height 128 and width 128.

5.6 Experiments

This section outlines how the two deep learning models designed to identify crime have been implemented. The principal goal of implementing the two deep learning models was to determine how crime can best be identified.

Comparing the performance of both models allowed us to explore their accuracy, efficiency, and overall applicability in actual crime detection procedures. The findings we derived from the experiments serve as the foundation of the selection for the most superior model that is capable of enhancing the accuracy and efficiency of crime detection and classification systems, hence making them usable in real scenarios.

5.6.1 YOLOv8n Model Implementation

The YOLOv8n crime detection model is utilized with the PyTorch deep learning framework. A function named create_yolo_model() is created to build the YOLOv8n model architecture. The process starts with an input layer processing video data of a certain shape.

The model consists of multiple block layers with different batch sizes, kernel sizes, strides, and activation functions (SiLU). The layers are responsible for extracting essential patterns and features from the input video. The features are then passed through the backbone, two C2f and SPPF modules, to enhance feature extraction. The processed data is then passed to the next layer, which aggregates features from different layers to promote multi-scale detection. Finally, the detection head concatenates feature maps from multiple convolutional layers and generates predictions through three detection layers at different scales. The model is optimized using an **AdamW optimizer**, and its performance is assessed using evaluation metrics such as **Precision, Recall, mAP0.5, and mAP0.5-0.95**. The dataset is divided using a **k-fold cross-validation** strategy, ensuring a robust evaluation process. The model is trained using the **fit function**, and during training, checkpoints are saved based on validation loss to enhance performance tracking and model selection.

```
def create_yolo_model():
    # Define path to data.yaml
    data_yaml_path = "C:/Users/Ayele/data.yaml"
    print("Path", data_yaml_path)

    # Check if data.yaml exists
    if not os.path.exists(data_yaml_path):
        print(f"Skipping training: '{data_yaml_path}' not found.")
        return None
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = YOLO('yolov8n.yaml')
    model.to(device)
    return model
```

5.6.2 Implementation Of Yolov11n Model

The implementation of the **YOLOv11n** model for crime detection follows a similar approach to the **YOLOv8n** model. A function named **create_yolo_model ()** is defined to construct the **YOLOv11n architecture**, where the input shape of the video data is specified, and multiple layers are incorporated to process the data effectively. All the layers are pre-defined in terms of a filter size, kernel size, and an activation function, which all contribute to capture the temporal dependencies and recognize patterns at different scales.

Similar to the YOLOv8n model, a global average pooling layer is added, reducing spatial dimensions, followed by a dense output layer with SiLU activation function for categorical predictions. The model is compiled with the AdamW optimizer, and key evaluation metrics such as Precision, Recall, mAP 0.5, and mAP 0.5-0.95 are used for performance evaluation. For ensuring proper evaluation, k-fold cross-validation is used. Training and validation sets are allotted to each fold from the folds. The model then trained using the fit function and model checkpoints are saved based on validation loss for optimization while training.

```
def create_yolo_model():
    # Define path to data.yaml
    data_yaml_path = "C:/Users/Ayele/data.yaml"
    print("Path", data_yaml_path)

    # Check if data.yaml exists
    if not os.path.exists(data_yaml_path):
        print(f"Skipping training: '{data_yaml_path}' not found.")
        return None
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
    model = YOLO("yolo11n.pt")
    model.to(device)
    return model
```

CHAPTER SIX

6. RESULT AND DISCUSSIONS

6.1 Chapter Overview

This chapter provides results achieved from the proposed crime detection model and comparative results via graphical representations in addition to tabular form. The explanation is based on implementation concepts previously discussed and methodological details.

6.2 Result Analysis

For performance testing of the proposed models, we have employed a preprocessed data and compared the performance of the proposed models with respect to principal assessment parameters like precision, recall, mAP 0.5, and mAP 0.5-0.95. Our study has taken into account developing two deep learning models, which were assessed through traditional benchmarking practices.To verify consistency of performance assessment, we conducted multiple time-series experiments using a 5-fold cross-validation method. We also generated precision-confidence curves, Recall-confidence curves and precision-recall curves to have a better idea about the accuracy and reliability of the models.

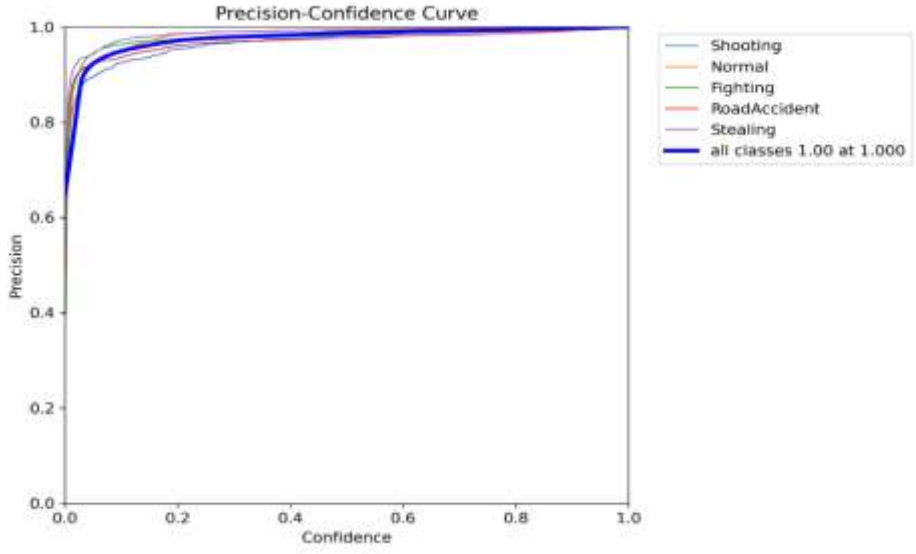


Figure 6.1 Precision-Confidence Curve yolov8n

Figure 6.1 illustrates Precision-Confidence Curve is a graphical illustration of the precision versus various confidence thresholds for individual classes within a model. Colored lines are for specific classes—Shooting, Normal, Fighting, Road Accident, and Stealing—and the blue

line for the overall performance on all classes. Precision generally rises with an increase in confidence threshold, indicating higher confidence predictions have more accurate classification. Notably, the "Shooting" class curve is consistently steep, with its peak precision of 1.00 occurring at a confidence of approximately 0.989, indicating that this class is handled best by the model. Precision in other classes is lower and at roughly comparable confidence levels, and these can be considered possible areas for optimization of the model for classification effectiveness. On balance, the curves demonstrate a strong performance across classes, particularly in critical situations, but also that more refinement is needed for lower-precision classes.

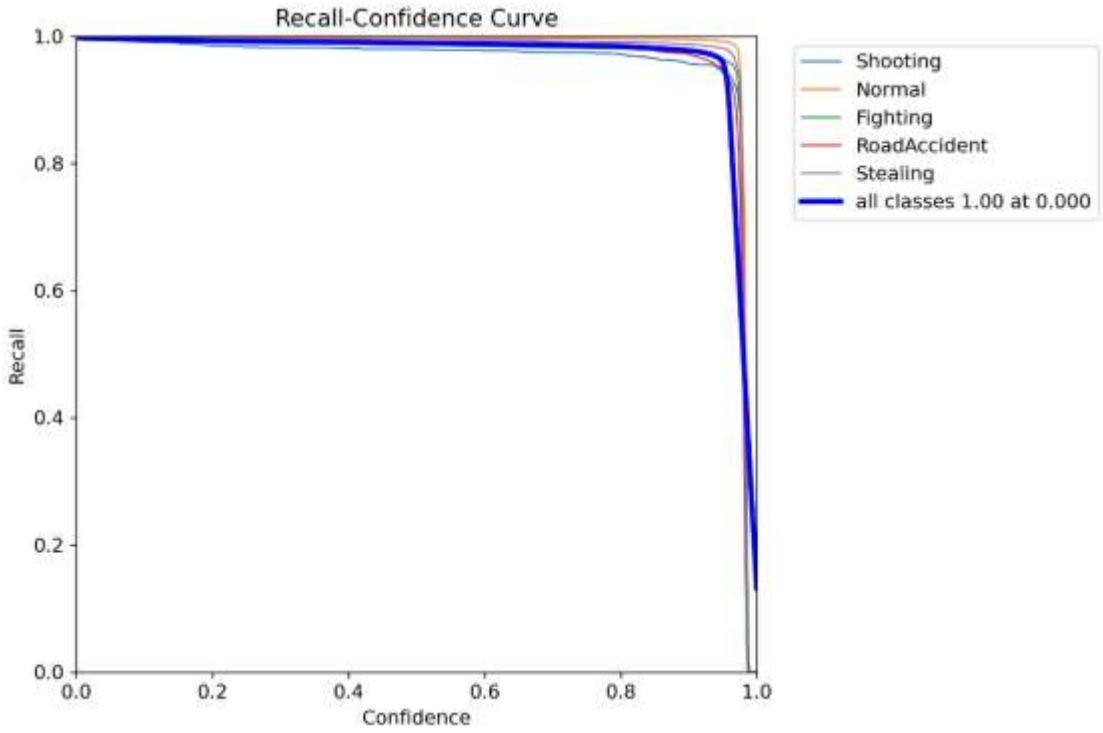


Figure 6.2 Recall-Confidence Curve yolov8n

Figure 6.2 shows the Recall-Confidence Curve illustrates the relationship between recall and confidence levels for different classes in the model. Every colored line is a line for one specific class—Shooting, Normal, Fighting, Road Accident, and Stealing—with the blue line representing the overall performance across all classes. Unlike the Precision-Confidence Curve, recall values are always high and high across a large number of confidence levels, indicating the strength of the model in finding true positive instances in all classes even at lower confidence thresholds. Curves illustrate how the model is able to identify much relevant

instance though on some classes performance falls very prominently, particularly if confidence level falls. The "Shooting" class has comparatively weaker recall with stronger confidence than the other classes, which means that precision is very high, but the model becomes more conservative in order to pick this class. Overall, the curves suggest the model has very strong recall across all classes, which shows that it can very well detect notable instances, but may be able to enhance performance for some of the categories if the model was tweaked.

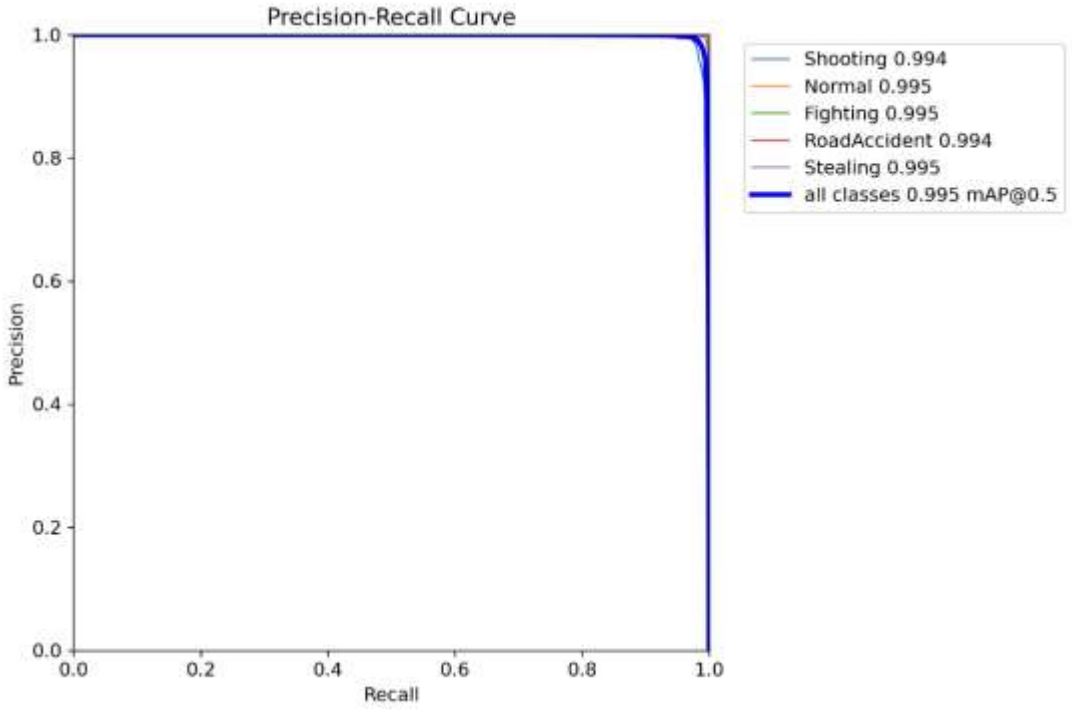


Figure 6.3 Precision-Recall Curve yolov8n

Figure 6.3 illustrates the Precision-Recall Curve plots precision-recall trade-off of different classes of the model with each colored line representing the performance of a specific class—Shooting, Normal, Fighting, Road Accident, and Stealing. The blue line represents overall performance of all classes. The graphs demonstrate that the accuracy is extremely high, close to 1.0, at various recall rates, demonstrating that the model performs extremely well in predicting positively, particularly for the classes under consideration. The near-perfect accuracy demonstrates that whenever the model predicts a positive instance, it is almost always correct. However, the graph indicates a steep drop in recall as it approaches 1.0, which indicates that even though the model is good in accuracy, it may lose a significant percentage of true positives at lower levels of recall. Overall, the high precision scores indicate a calibrated model, but the

steep drop in recall indicates a potential shortcoming in identifying all event instances, meaning more optimization is needed to improve detection, especially for infrequent events.

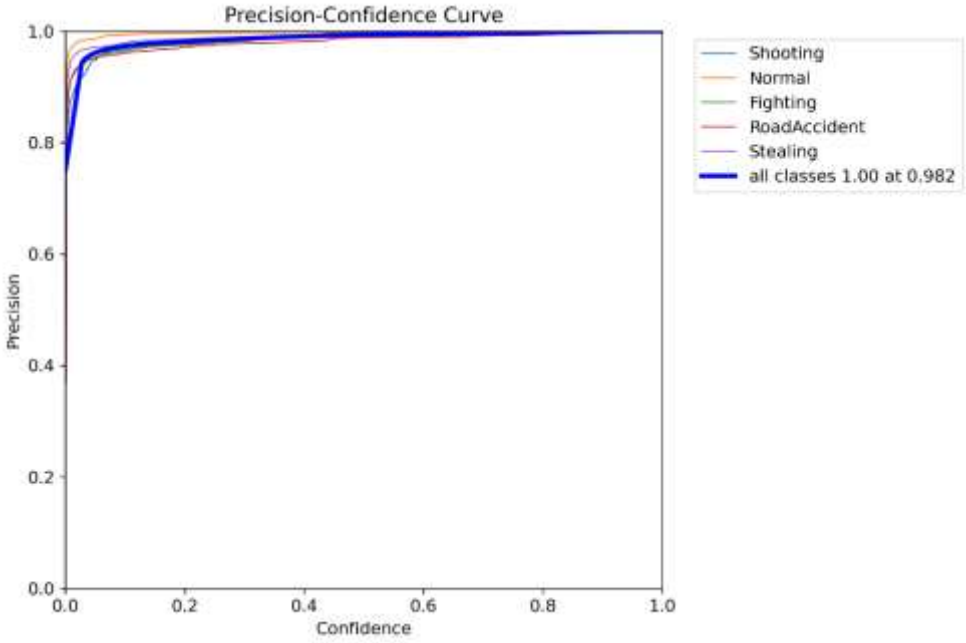


Figure 6.4: precision confidence curve yolov11n

As can be seen from Figure 6.4, Precision-Confidence Curve for the detection model, high performance across all the classes, i.e., Shooting, Normal, Fighting, Road Accident, and Stealing, can be observed. With growing confidence, precision values tend towards 1.00, which signifies that with mounting confidence in predictions by the model, more correct prediction takes place along with decreasing false positives. The horizontal line at high confidence levels indicates that the model maintains high precision in predicting with high confidence. With overall precision of 1.00, this is indicative of no misclassifications at high confidence levels, which supports the model's capacity to correctly and confidently detect incidents. This finding validates the effectiveness and reliability of the model for real-world detection issues, where confident and accurate predictions are crucial.

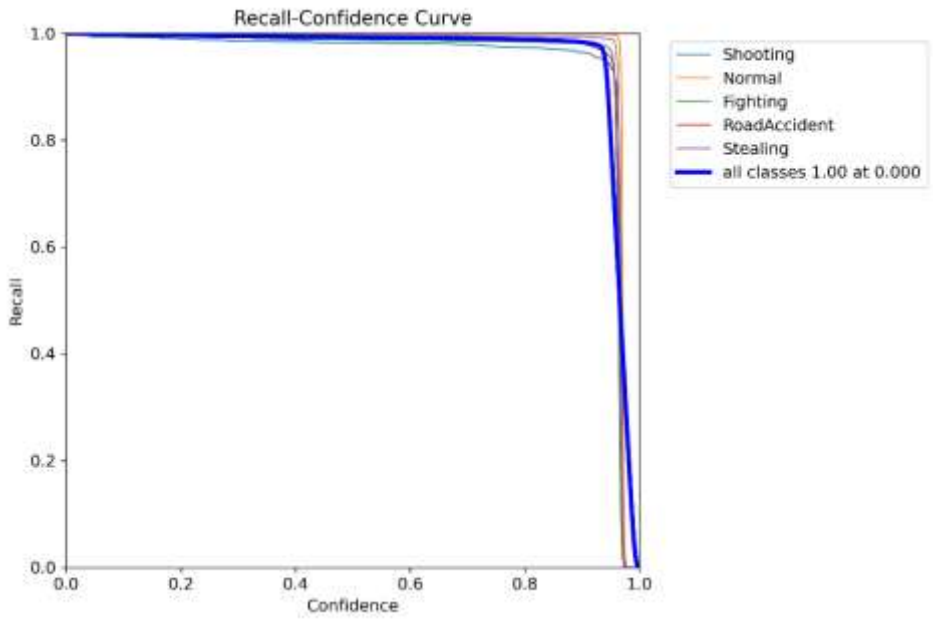


Figure 6.5 Recall confidence curve yolov11n

Figure 6.5 illustrates the Recall-Confidence Curve that provides the detection model's performance in five classes: Shooting, Normal, Fighting, Road Accident, and Stealing. Confidence values ranging from 0 to 1 are on the x-axis, while recall values within the same range are plotted on the y-axis. It is apparent that with growing confidence, all classes tend towards a recall of 1.00, demonstrating the model's ability to detect almost all true positives at high confidence. There is a specific color for each class—Shooting with blue, Normal with orange, Fighting with green, Road Accident with red, and Stealing with purple—demonstrating that all the curves converge closely, particularly at low confidence thresholds. Such proximity of convergence here illustrates the model's general capacity to provide high recall for all the classes, testifying to its consistency in adequately detecting events at greater confidence.

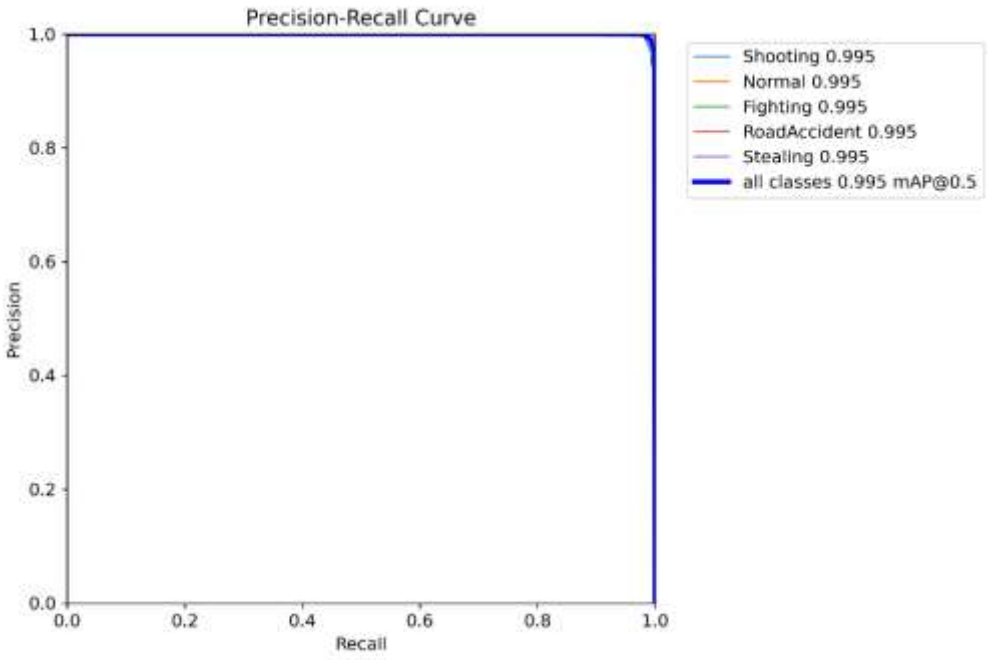


Figure 6.6: Precision-recall curve yolov11n

The above Figure 6.6 illustrates the Precision-Recall Curve of the detection model with excellent performance on all the classes, namely Shooting, Normal, Fighting, Road Accident, and Stealing, with precision as high as 0.995. The extremely high precision here indicates that the model is extremely successful in rightfully identifying positive instances with very few false positives.The high precision throughout the range of recalls on the curve shows the model's strength in terms of detection accuracy even when trying to detect more true positives. The overall model exhibits reliability in distinguishing between the various classes, as represented by a high mean Average Precision (mAP) of 0.995. The performance demonstrates its usability in real-life applications with high accuracy. In conclusion, the results show that the model performs well in balancing precision and recall, which makes it suitable for many detection purposes.

6.3 Results for Deep Learning Models

6.3.1 YOLOV8n Model Results

Our research started with the deployment of the YOLOv8n model, a deep learning model that is widely utilized for crime detection. For accurate assessment, we used a 5-fold cross-validation approach and split the dataset into five partitions. We trained the model using each subset, while the remaining were reserved for validation and testing.

The learning process lasted 30 epochs in each fold and had 465 training steps to enhance the model performance per epoch.Then, we considered the YOLOv11n model, which enhances previous YOLO models through the addition of further architectural innovation. The YOLOv11n model applies the use of the neck layer to maximize the feature extraction process and accuracy of crime detection.Our findings showed that YOLOv11n outperformed YOLOv8n, having improved detection accuracy and performance generally in crime detection operations.

Table 6.1 Results of YOLONanoV8n

Crime class	YOLOv8n			
	Precision	Recall	mAP50	mAP50-95
All	0.993	0.985	0.995	0.99
shooting	0.991	0.974	0.994	0.994
Normal	0.998	0.996	0.995	0.995
Fighting	0.996	0.981	0.995	0.995
Road Accident	0.986	0.983	0.994	0.994
stealing	0.996	0.991	0.995	0.995

6.3.2 YOLOV11n Model Results

At this phase of research, experiments were conducted on utilizing the YOLOv11n model for crime detection. The model utilizes dilated convolutions, which enhance the receptive field and boost the ability to learn long-term dependencies over time. Such an architecture allows YOLOv11n to function well with video data and identify intricate temporal patterns, which makes it greatly appropriate for utilization in crime detection. To compare its efficiency, the YOLOv11n model was trained and evaluated on our dataset. As major evaluating metrics, Precision, Recall, mAP 50, and mAP 50-95 were used to measure its efficiency. These metrics determined the model's prediction accuracy and overall efficiency so that it could be contrasted with other models, i.e., YOLOv8n and previous models of YOLOv11n. The experiment results are given in the table 6.1.

Table 6.2 Results of YOLOV11n

Crime class	YOLOv11n			
	Precision	Recall	mAP50	mAP50-95
All	0.995	0.992	0.995	0.995
shooting	0.997	0.982	0.995	0.995
Fighting	0.996	0.987	0.995	0.995
stealing	0.995	0.996	0.995	0.995
Road Accident	0.99	0.996	0.995	0.995
Normal	0.999	0.999	0.995	0.995

Figures 6.7 to 6.12 present graphical plots of the performance of YOLOv8n and YOLOv11n models on various crime categories. The plots present useful information about the relative performance of the two models, highlighting their capacity to efficiently detect crimes of various classifications.

Performance Metrics of YOLOv8n model with UCF dataset

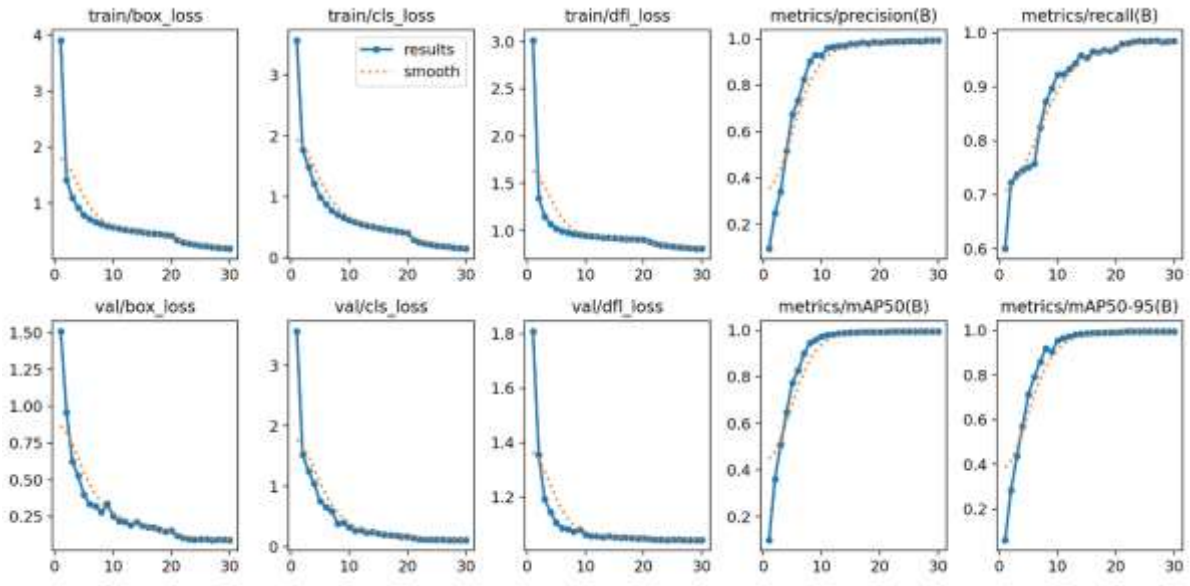


Figure 6.7 Performance Metrics of YOLOv8n model

Figure 15 presents the plot offers a close view of the training and validation loss and the critical performance metrics of a model through multiple epochs. In the first row, training box loss as well as classification loss both sharply decline, indicating successful learning, and the DFL loss becomes stable after dropping initially, indicating stability in model predictions. The corresponding precision and recall measures have a consistent rise, indicating the model's

increased accuracy in selecting pertinent instances. The validation losses are aligned with training losses in the bottom row, with box loss and classification loss also drastically falling, verifying the model's generalization to unseen data. The DFL loss is low, indicating good management of uncertainty by the model. The mAP scores in the bottom columns represent good performance with different thresholds, which validate the capacity of the model to balance recall and precision. Trends usually indicate good training progress and quality model performance, suggesting readiness for deployment into real-world applications.

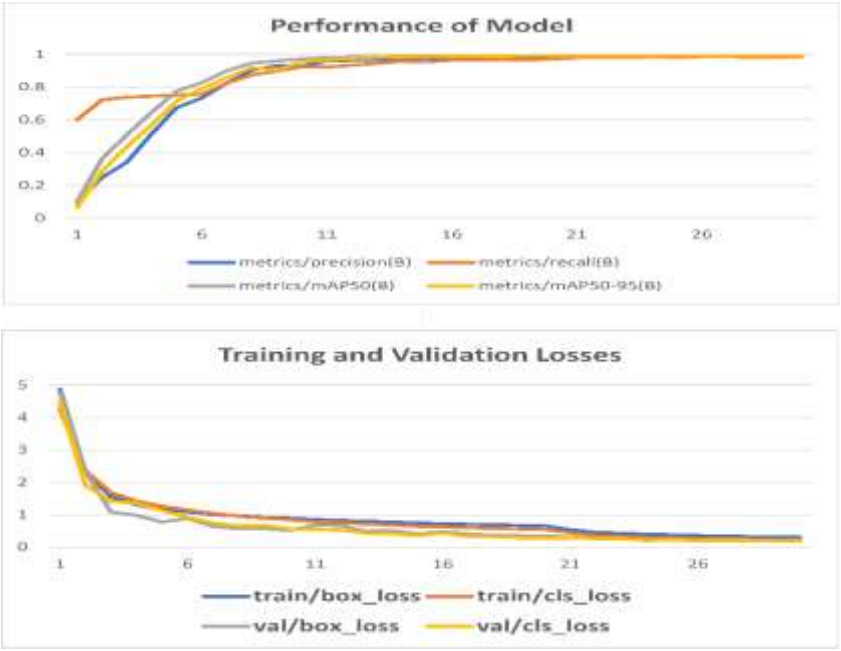


Figure 6.8 Training and validation losses and Evaluate performance of yolov8n

Figure 6.8 illustrate the plot presents a general overview of the model's performance metrics and training/validation losses over a range of epochs. At the top, the performance metrics—precision, recall, and mean Average Precision (mAP)—all have a clear upward trend, indicating that the predictive ability of the model significantly improves as training progresses. Observably, precision and recall approach 1.0, reflecting that the model is very precise in its prediction, with the mAP measures being consistently high, showing good performance across thresholds. At the bottom, training and validation losses of both box and classification losses drop drastically, indicating successful learning during training. The validation loss curves are extremely close to the training loss curves, which is a sign of good generalization to new data. However, the slight divergence towards the end might be a sign of potential overfitting, where

the model is fitting the training data extremely well but may not be able to maintain that level of performance on validation data. Overall, the graph conveys a positive trend in model performance, yet also shows room for further optimization in order to reach consistent accuracy across all data sets

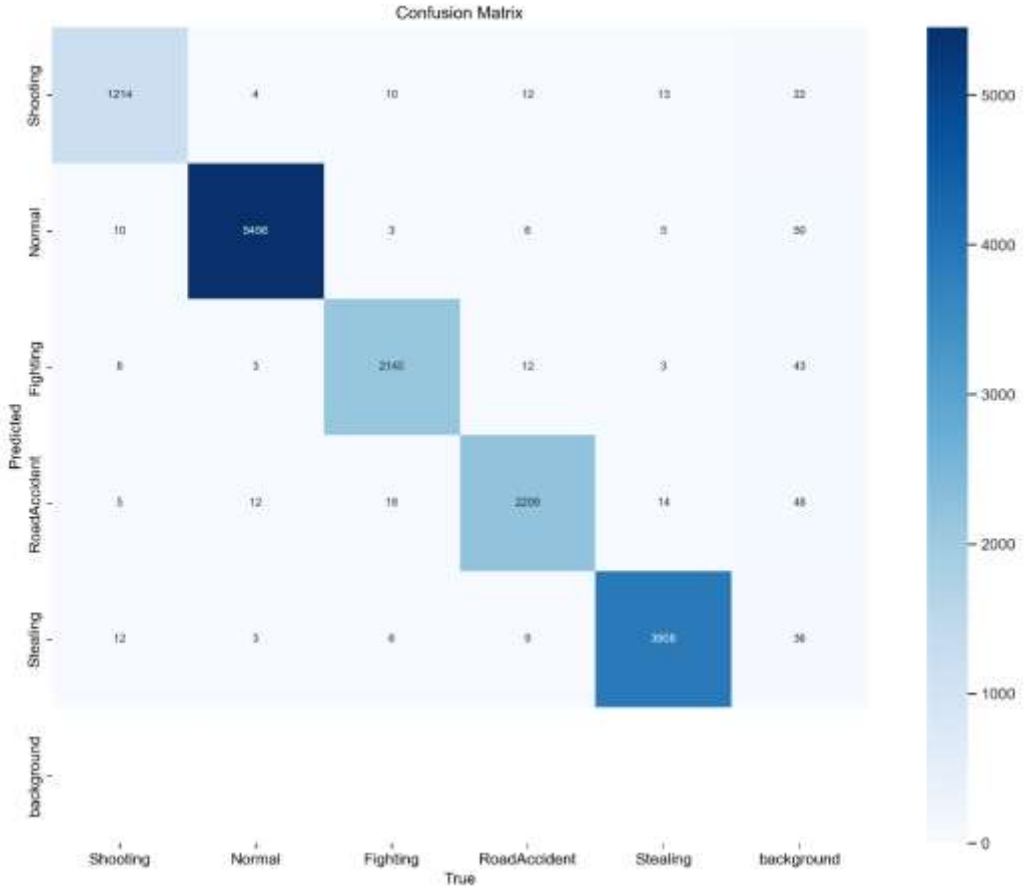


Figure 6.9 Confusion Metrics of Yolov8n

Figure 6.9 is the confusion matrix provides a breakdown of how the model performs across various classes, indicating how accurately the model can distinguish between the categories: Shooting, Normal, Fighting, Road Accident, Stealing, and background. The correct classifications are represented by the diagonal elements, of which Normal has the highest count (5,486), and it indicates that the model performs best for this class. But the matrix also shows some mis predict; e.g., 2145 examples of Road Accident were classified under Fighting, implying that the model is confusing these two classes. Likewise, there are significant instance mis predicted of background instances as other classes, most notably Stealing (48 examples). The low numbers for classes such as Shooting and Stealing imply that these could be more

difficult for the model to accurately Predict. In general, although the model works well in most categories, the confusion matrix shows certain categories that need to be addressed and fine-tuned for better overall accuracy and lower mis prediction rates.

Performance Metrics of YOLOv11n model with UCF dataset

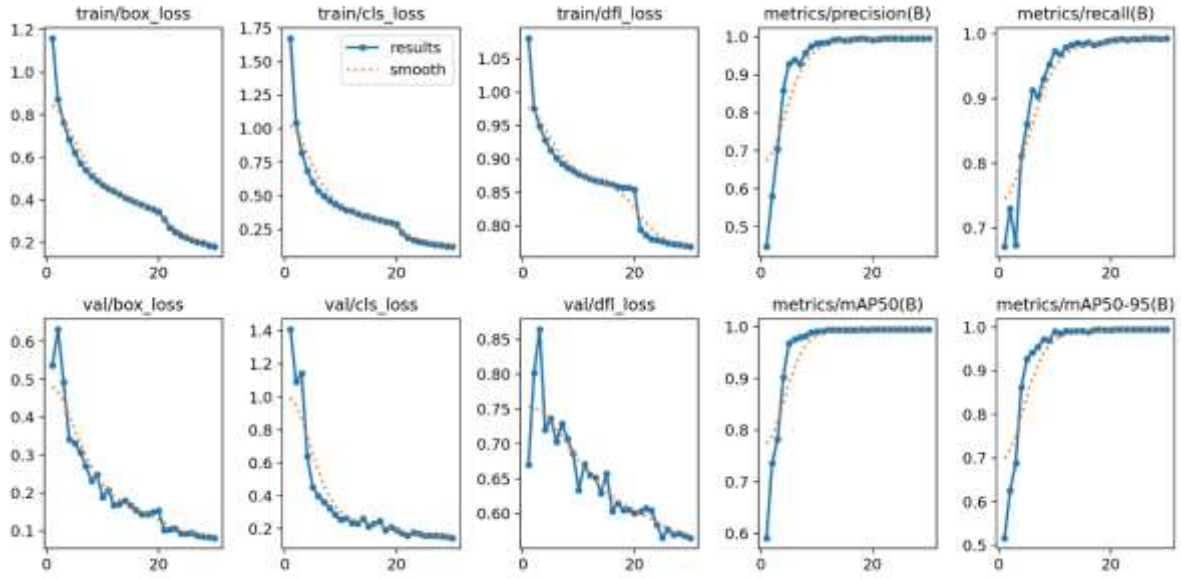
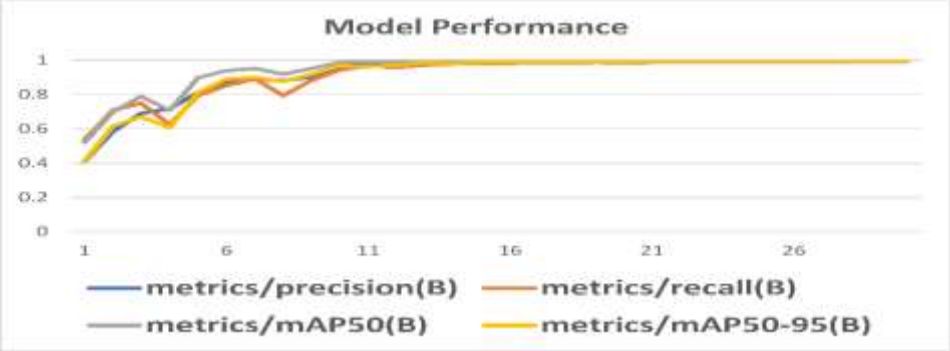


Figure 6.10 Performance Metrics of YOLOv11n model

Figure 6.10 offers a summary of the training of the detection model, plotting important metrics like train and validation box loss, classification loss, precision, recall, and mAP versus epochs. Both training and validation losses follow a downward trend, indicating good learning and error reduction. Validation losses also decline, indicating good generalization to new data. High precision and recall on both datasets point towards accurate detection of positive samples. The mAP curves, particularly at thresholds 0.5 and 0.5-0.95, demonstrate the model's very good capability in class distinction. In general, these results validate that the detection model is well-trained and stable, and can be applied in practical use.



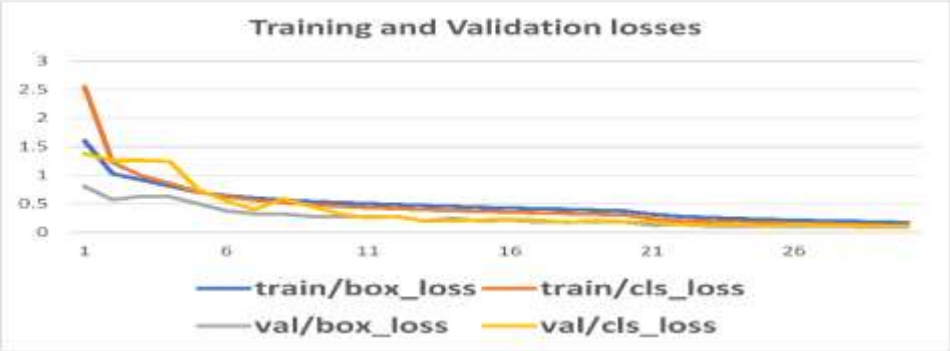


Figure 6.11 Training , validation losses and Evaluate performance of yolov11n

Figure 6.11 illustrate the plots provide an indication of the performance of a Deep learning model over training iterations. In the graph above, the validation metric for Precision, Recall, and mean Average Precision (mAP) consistently indicate high values, which is an indication of the model's good ability to classify instances correctly without significant oscillation, particularly in the metrics at thresholds of 0.5 and across the range of 0.5 to 0.95. This is an indication of good detection abilities. The lower curve is training and validation losses as a function of epochs, wherein the two losses decrease sharply, the training loss dropping sharply over the validation loss. This is a trend indicative of the model learning well from the training and the model works well and there are good learning dynamics.

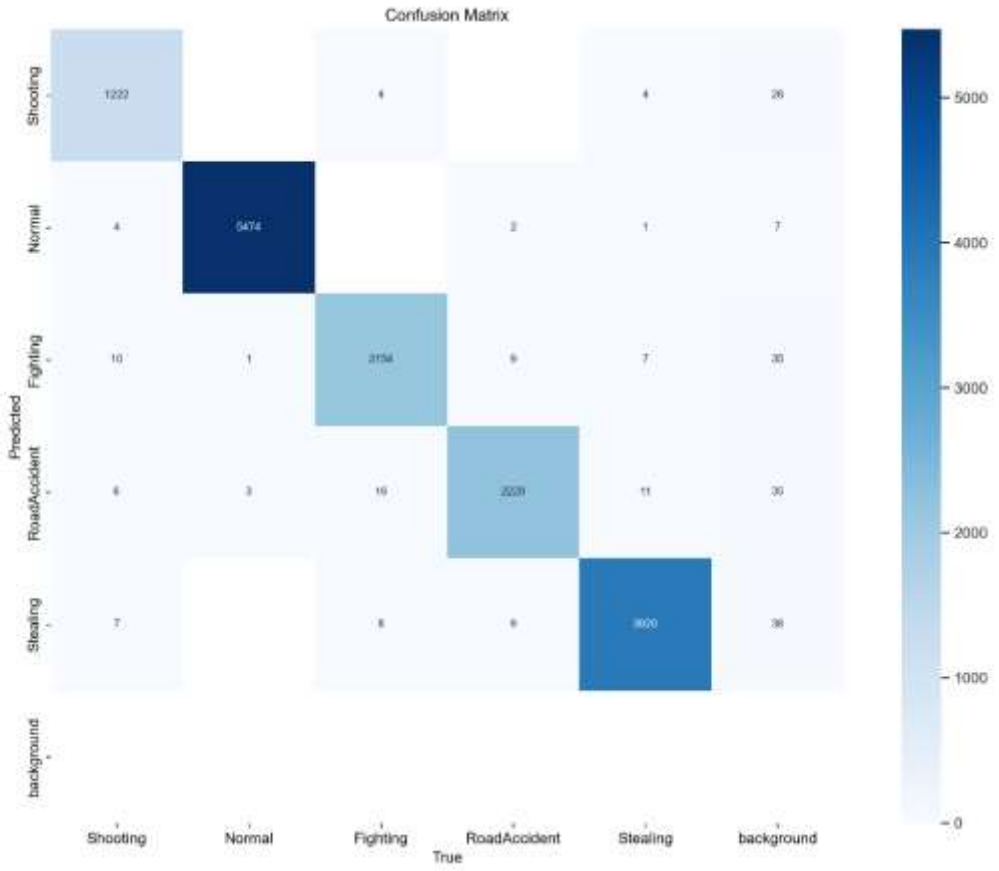


Figure 6.12 Confusion Metrics of Yolov11

Figure 6.12 depicts the confusion matrix, wherein the model's strengths and weaknesses are clear. The "Normal" class has very high accuracy with only 4 mislabeled as "Fighting." and Road Accident The "Shooting" class contains 27 mis predict. The "Fighting" class also contains 28 mis predict. the "Road/Accident" class contains 20 misprediction. The "Stealing" class has 23 instance miss labeled. Good non-crime instance detection is reflected in high true negatives for the "background" category. Overall, while the model detects "Normal" events and "Road/Accident" well, refinement for "Fighting" and "Shooting" is needed.

6.4 Hyper parameter Tuning Result

For the purpose of performance improvement of our models (YOLOv8n, YOLOv11n), we conducted large-scale hyper parameter tuning experiments. The primary objective of these experiments was to test the impact of varying the number of training epochs, batch size and activation function. By varying these hyper parameters, we wanted to discover the optimal combinations which would assist in the performance improvement of the models.

For hyper parameter tuning experiments, we trained all the models with 20 numbers of epochs. This allowed us to check their performance on different training time and determine the effect of training. We also experimented with the models on different performance metrics like Precision, recall mAP50 and mAP50-95 The result of the hyper parameter tuning experiments on crime detection is shown in Table 8. We observed that the performance of the models varied based on the number of epochs. For instance, in the YOLOv8n model, the epochs were raised from 20 to 30, and there was a slight decrease in mAP50-95 and, while Recall, precision and mAP 50 also improve slightly.in the YOLOv8n model, the number of epochs was increased from 20 to 30, and this caused a slight rise in Recall.

Table 6.3 Hyper parameter (Epochs) tuning result

Model	Epoch	Precision	Recall	mAP50	mAP50-95
YOLOv8n	20	0.983	0.977	0.993	0.944
YOLOv8n	30	0.993	0.985	0.995	0.993
YOLOv11n	20	0.994	0.99	0.993	0.993
YOLOv11n	30	0.995	0.992	0.995	0.995

Our Contribution to Crime Detection

My job involves processing crime data for effective detection by doing the following:

Selection and Extraction of Video Frames: I selectively picked and extracted relevant frames from video footage to make sure that the most informative moments were captured for examination.

Annotation Generation: I created annotations from the frames extracted that facilitated the classification of various crime-related activities. This involved tagging the objects and actions in the frames to create a comprehensive dataset for training.

Performance Improvement: To enhance the detection accuracy, I used the YOLOv11 object detection model, which is both accurate and fast for real-time processing. Through the support of this enhanced model, I could significantly improve the performance of crime detection operations.

Table 6.4 Model comparison

Model	Precision	Recall	mAP50	mAP50-95
Yolov8n	0.993	0.985	0.995	0.993
Yolov11n	0.995	0.992	0.995	0.995

Table 6.4 illustration the comparison of the detection models Yolov8n and Yolov11n reveals contrasting performance on several key metrics. Yolov11n outperforms Yolov8n on both precision (0.995 vs 0.993) and recall (0.992 vs 0.9835), indicating that it is not only better at predicting correctly but also at identifying more true positives. Both models share the same mean Average Precision at 50 (mAP@50) at 0.995, indicating that they are tied at this threshold. However, Yolov11n takes the lead in mean Average Precision from 0.5 to 0.95 (mAP@0.5:0.95) with 0.995 compared to Yolov8n's 0.993, showing that Yolov11n is more stable and consistent at different confidence levels. Overall, Yolov11n is the superior model with higher accuracy and reliability for detection tasks.

Train batch bounding box

Figure 6.13 Train batch bounding box yolov8

Figure 6.14 Train batch bounding box yolov11

Predicted and Actual bounding box

Figure 6.15 Actual and predicted bounding box yolov8

As illustrate Figure 6.15 the two images on the left are actual scenes from crime detection cases, and they represent real-world scenarios that must be examined. The two images on the right are the predicted bounding boxes generated by a deep learning model, and they indicate the regions of interest found within the actual scenes. These bounding boxes indicate where the model finds relevant objects or activities, providing insights into potential criminal activity or events. The

comparison between the actual images and the estimated bounding boxes emphasizes the model's ability to interpret and process visual data in crime detection.

Figure 6.16 Actual and predicted bounding box yolov11

As illustrate Figure 6.16 the two images on the left are actual scenes from crime detection cases, and they represent real-world scenarios that must be examined. The two images on the right are the predicted bounding boxes generated by a deep learning model, and they indicate the regions of interest found within the actual scenes. These bounding boxes indicate where the model finds relevant objects or activities, providing insights into potential criminal activity or events. The comparison between the actual images and the estimated bounding boxes emphasizes the model's ability to interpret and process visual data in crime detection.

Table 6.5 Related works comparison

Study	**Algorithm**	**Recall**	**Precision**	mAP
M. Tahir Bhatti (2021)(Bhatti et al., 2021)	YoloV4	0.88	0.93	N/A
	SSD-MobileNet-v1	0.602	0.628	N/A
Thakur(Thakur et al., 2024)	yolov8	0.85	0.80	0.78
Abdirahman Osman(Hashi et al., 2023)	yolov6	N/A	N/A	0.631
	Faste R_CNN	N/A	N/A	0.618
Ours proposed	yolov8n	0.993	0.985	0.993
	yolov11n	0.995	0.992	0.995

Table 6.5 illustrate YOLOv11n model proposed here clearly outperforms all the existing object detection models. Its high precision and recall rates demonstrate a remarkable ability to identify and classify objects precisely, surpassing all the existing models, including YOLOv8 and YOLOv4. Even though these models are well performing, they fail to match the accuracy and dependability of YOLOv11n. The result indicates that YOLOv11n is a significant upgrade from its earlier versions, and it is even a better choice for object detection. Its enhanced performance makes it viable for real-world applications and reinforces its position as our best algorithm in the category.

Reasons for Improved Accuracy:

Advanced Algorithm Selection: We used YOLOv8n and YOLOv11n, which are state-of-the-art models known to perform better in object detection tasks. These models possess architectural improvements that extract features better and improve processing speed.

Optimized Training with Labeled Datasets: Our solution was to build a highly qualitative and complete annotated dataset for the task of crime detection. The dataset allows the models to learn more, as it encompasses a variety of scenarios and objects that are related to the domain of crime.

Improved Data Handling Methods: We employed data augmentation methods that increased the size of the dataset and made it more variable, allowing the models to generalize better to real-world situations. This method avoids overfitting and improves recall and precision.

6.5 Discussion of Results

The aim of the research before was to determine the optimal deep learning architecture to enhance the effectiveness of crime detection systems. Through a detailed study, it was determined that YOLOv11n performed better than the rest of the deep learning architectures consistently, and crime detection was enhanced significantly. Through strict testing with crime data sets, YOLOv8n and YOLOv11n were determined to be better-performing models than the other models. They significantly improved accuracy in crime detection, efficiency in crime identification, and decision-making through their high-level features.

CHAPTER SEVEN

7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In summary, this study centered on crime detection and sought to to accurately detect the crime and identify the class of the crime detected. By conducting various experiments and evaluations, it has been determined that the YOLOv8n and YOLOv11n performs well in this crime detection tasks.

This research demonstrates the effective application of deep learning algorithms, specifically YOLOv8n and YOLOv11n, in crime detection tasks. By leveraging their object detection capabilities, the study achieved significant accuracy in identifying suspicious activities and objects associated with criminal activities. YOLOv8n portrays a good trade-off between computational efficiency and detection precision, which is ideal for systems with limited resources. However, YOLOv11n provides higher precision and flexibility, and it is a very good option for high-performance crime detection systems.

The findings show the potential of advanced deep learning model, YOLO architectures to enhance public safety by enabling automated and proactive crime detection measures. Future research can focus on improving model robustness in diverse real-world scenarios, optimizing algorithms for edge devices, and integrating multimodal data to further augment the efficacy of crime detection systems. With continued advancements in deep learning, these methodologies can significantly contribute to building safer and more secure communities in different areas.

The findings of this research have significant implications for the to make a meaningful societal impact, enabling informed decisions regarding crime detection by improving the efficiency, accuracy, and responsiveness of surveillance systems. By accurately detecting the crime, the burden on the investigators can be reduced, human biases can be reduced in crime identification., Safety can be enhanced, and identification expenses can be streamlined. In summary, this research adds to the field of crime detection by demonstrating strong performance with the proposed YOLOv8n and YOLOv11n models. Their effective implementation underscores their potential to improve crime detection efforts in various regions, resulting in more efficient and dependable detection activities.

7.2 Future works

There are some promising areas in the domain of crime detection and classification that need further research focus and development. The following are some areas for further improvement that will add to the effectiveness of the systems for crime detection:

- Testing the models across diverse environmental conditions, such as variable lighting, occlusion, and weather, for improved adaptability in real-world scenarios;
- Optimization of YOLOv8n and YOLOv11n for deployment on low-power and edge devices, ensuring scalability and widespread applicability.
- Merging video, audio, and other sensor data to create a better abstraction in the area of activities related to crime detection.
- Incorporating mechanisms for online learning by which the models adapt dynamically in an ever-changing pattern of criminal behavior.
- Addressing possible ethical concerns regarding privacy invasion, and researching anonymization techniques for compliance with data protection regulations.

REFERENCES

Amit, Y., Felzenszwalb, P., & Girshick, R. (2020). Object Detection. *Computer Vision, January 2020*, 1–9. https://doi.org/10.1007/978-3-030-03243-2_660-1

Anti´c, B. A., & Ommer, B. (2011). *Video Parsing for Abnormality Detection*. IEEE.

Bhat, S. P., & S, K. T. (n.d.). Detection of Abnormal Events. *International Journal of Innovations in Engineering and Technology*. https://doi.org/10.21172/ijiet.124.11

Bhatti, M. T., Khan, M. G., Aslam, M., & Fiaz, M. J. (2021). Weapon Detection in Real-Time CCTV Videos Using Deep Learning. *IEEE Access*, *9*, 34366–34382. https://doi.org/10.1109/ACCESS.2021.3059170

Butt, U. M., Letchmunan, S., Hassan, F. H., & Koh, T. W. (2024). Leveraging transfer learning with deep learning for crime prediction. *PLoS ONE*, *19*(4 April), 1–20. https://doi.org/10.1371/journal.pone.0296486

Chen, X., & others. (2023). YOLOv10: A Comprehensive Review and Implementation. *ArXiv Preprint*.

Del Giorno, A., Bagnell, J. A., & Hebert, M. (2016). *A Discriminative Framework for Anomaly Detection in Large Videos*.

Hashi, A. O., Abdirahman, A. A., Elmi, M. A., & Rodriguez, O. E. R. (2023). Deep Learning Models for Crime Intention Detection Using Object Detection. *International Journal of Advanced Computer Science and Applications*, *14*(4), 300–306. https://doi.org/10.14569/IJACSA.2023.0140434

Huang, H., Wang, B., Xiao, J., & Zhu, T. (2024). Improved small-object detection using YOLOv8: A comparative study. *Applied and Computational Engineering*, *41*(1), 80–88. https://doi.org/10.54254/2755-2721/41/20230714

Kamijo, S., Matsushita, Y., Ikeuchi, K., & Sakauchi, M. (2000). Traffic Monitoring and Accident Detection at Intersections. *IEEE Transactions on Intelligent Transportation Systems*, *1*(2), 108–117. https://doi.org/10.1109/6979.880968

Kuang, W. (2023). Object Detection-YOLO. *Fundamentals of Deep Learning: A Step-by-Step Guide*.

Li, H., & Wang, L. (2024). YOLOv11: The Future of Object Detection. *Computer Vision and Image Understanding*. https://doi.org/10.1016/j.cviu.2024.104567

Mohammadi, S., Perina, A., Kiani, H., & Murino, V. (2016). *Angry Crowds: Detecting Violent*

Events in Videos (pp. 3–18). https://doi.org/10.1007/978-3-319-46478-7_1

Murino, V., & Puppo, E. (Eds.). (2015). *Image Analysis and Processing — ICIAP 2015* (Vol. 9280). Springer International Publishing. https://doi.org/10.1007/978-3-319-23234-8

Piza, E. L., Welsh, B. C., Farrington, D. P., & Thomas, A. L. (2019). CCTV surveillance for crime prevention: A 40-year systematic review with meta-analysis. *Criminology and Public Policy*, *18*(1), 135–159. https://doi.org/10.1111/1745-9133.12419

Plastiras, G., Kyrkou, C., & Theocharides, T. (2018). Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3243394.3243692

Rabiee, H., Haddadnia, J., Mousavi, H., Nabi, M., Murino, V., & Sebe, N. (2016). *Emotion-Based Crowd Representation for Abnormality Detection*.

Rui, B. (2023). *ADL : Anomaly Detection and Localization in Crowded Scenes Using Hybrid Methods*. https://doi.org/10.20944/preprints202305.1617

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

Stalidis, P., Semertzidis, T., & Daras, P. (2021). Examining Deep Learning Architectures for Crime Classification and Prediction. *Forecasting*, *3*(4), 741–762. https://doi.org/10.3390/forecast3040046

Sultani, W., Chen, C., & Shah, M. (n.d.). *Real-world Anomaly Detection in Surveillance Videos*. http://crcv.ucf.edu/projects/real-world/

Thakur, A., Shrivastav, A., Sharma, R., Kumar, T., & Puri, K. (2024). *Real-Time Weapon Detection Using YOLOv8 for Enhanced Safety*. 1–21. http://arxiv.org/abs/2410.19862

Wang, C. Y., & others. (2023). YOLOv8: A New Era of Real-Time Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2023.1234567

Wang, J., Zhang, Q., Xie, H., Chen, Y., & Sun, R. (2024). Enhanced Dual-Channel Model-Based with Improved Unet++ Network for Landslide Monitoring and Region Extraction in Remote Sensing Images. *Remote Sensing*, *16*(16). https://doi.org/10.3390/rs16162990

Wang, Z., Hou, C., Li, B., Chen, T., Yao, L., & Song, M. (2018). Global Abnormal Event Detection in Video via Motion Information Entropy. *2018 2nd URSI Atlantic Radio Science Meeting (AT-RASC)*, 1–4. https://doi.org/10.23919/URSI-AT-

RASC.2018.8471516

Wu, P., Liu, J., Shi, Y., Sun, Y., Shao, F., Wu, Z., & Yang, Z. (2020). *Not only Look, but also Listen: Learning Multimodal Violence Detection under Weak Supervision*. http://arxiv.org/abs/2007.04687

Zhang, Y., & Liu, J. (2024). YOLOv9: Enhancements in Speed and Accuracy for Object Detection. *Journal of Computer Vision and Image Understanding*. https://doi.org/10.1016/j.jcviu.2024.103456

APPENDIXES

Appendix 1

sample code of library

[1]:
```
import os
import glob
import re
import cv2
import shutil
import torch
import numpy as np
import sys
import random
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, confusion_matrix, ConfusionMatrixDisplay
from ultralytics import YOLO
import seaborn as sns
import pandas as pd
import yaml  # For creating the data.yaml file
```

Preprocessing sample code

```
# Define transformations
train_transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])

val_transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

Appendix 2

Frame extraction and Generating Annotation sample Code

```
def video_to_frames_and_labels(video_path, image_output_folder, label_output_folder, frame_rate=30, class_name="Fighting", log_file="processed_videos.log
    class_id = class_id_map.get(class_name)
    if class_id is None:
        raise ValueError(f"Class '{class_name}' not in class_id_map.")

    video_name = os.path.splitext(os.path.basename(video_path))[0]
    if is_video_processed(video_name, log_file):
        print(f"Skipping {video_name}: Already processed.")
        return

    cap = cv2.VideoCapture(video_path)
    for frame_count in range(int(cap.get(cv2.CAP_PROP_FRAME_COUNT))):
        success, image = cap.read()
        if not success: break
        if frame_count % frame_rate == 0:
            frame_file = os.path.join(image_output_folder, f"{video_name}_{frame_count}.jpg")
            label_file = os.path.join(label_output_folder, f"{video_name}_{frame_count}.txt")
            if not os.path.exists(frame_file):
                cv2.imwrite(frame_file, image)
                with open(label_file, 'w') as label:
                    label.write(f"{class_id} 0.5 0.5 0.5 0.5\n")
    cap.release()
    mark_video_as_processed(video_name, log_file)

def is_video_processed(video_name, log_file):
    if not os.path.exists(log_file):
        return False
    with open(log_file, 'r') as file:
        return video_name in file.read().splitlines()

def mark_video_as_processed(video_name, log_file):
    with open(log_file, 'a') as file:
        file.write(f"{video_name}\n")
```

Appendix 3

Train Model sample code

```python
def train_model(model, data_yaml_path, epochs=30):
    results = model.train(data=data_yaml_path,
        epochs=30,
        batch=128,
        imgsz=128,
        workers=3
    )
    log_data = {
        "mean_results": results.mean_results,
        "detailed_results": results.results_dict
    }
    print("Training complete!")
    return results, log_data
```

Numerical Result of yolov8n

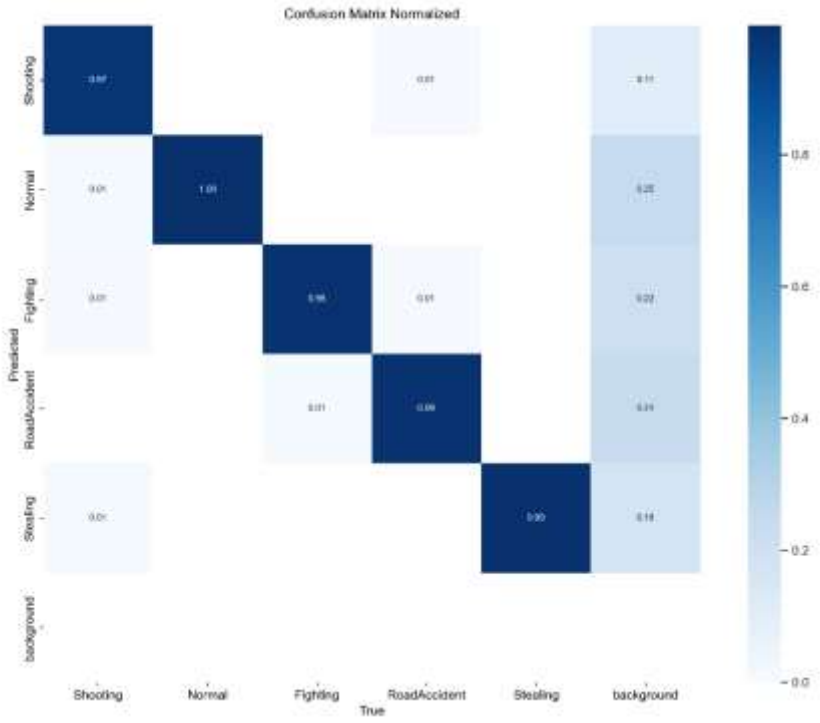
Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	15100	15100	0.993	0.985	0.995	0.995
Shooting	1249	1249	0.991	0.974	0.994	0.994
Normal	5478	5478	0.998	0.996	0.995	0.995
Fighting	2182	2182	0.996	0.981	0.995	0.995
RoadAccident	2248	2248	0.986	0.983	0.994	0.994
Stealing	3943	3943	0.996	0.991	0.995	0.995

Appendix 4

Numerical result of Yolov11n

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	15100	15100	0.995	0.992	0.995	0.995
Shooting	1249	1249	0.997	0.982	0.995	0.995
Normal	5478	5478	0.999	0.999	0.995	0.995
Fighting	2182	2182	0.996	0.987	0.995	0.995
RoadAccident	2248	2248	0.99	0.996	0.995	0.995
Stealing	3943	3943	0.995	0.996	0.995	0.995

Visual result of Yolov8n



***Appendix** 5*

Visual result of yolov11n