HUMAN DETECTION THROUGH INFRARED SURVEILLANCE CAMERAS USING DEEP LEARNING MODELS

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

M. Saai Vinieth , 221910308036
Bhavin Patel , 221910308041
Pranav Reddy G , 221910308044

Under the esteemed guidance of

Mr. K.L.Narasimha Rao
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING GITAM

(Deemed to be University)

HYDERABAD

APRIL - 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GITAM SCHOOL OF TECHNOLOGY
GITAM

(Deemed to be University)

DECLARATION

I/We, hereby declare that the project report entitled "**HUMAN DETECTION THROUGH INFRARED SURVEILLANCE CAMERAS USING DEEP LEARNING MODELS**" is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date:

Registration No(s).	**Name(s)**	**Signature(s)**
221910308036	M. SAAI VINIETH	
221910308041	BHAVIN PATEL	
221910308044	PRANAV REDDY G	

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GITAM SCHOOL OF TECHNOLOGY

GITAM

(Deemed to be University)

CERTIFICATE

This is to certify that the project report entitled " **HUMAN DETECTION THROUGH INFRARED SURVEILLANCE CAMERAS USING DEEP LEARNING MODELS**" is a bonafide record of work carried out by **M. SAAI VINIETH (221910308036), BHAVIN PATEL (221910308041), PRANAV REDDY G (221910308044)** students submitted in partial fulfilment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

Project Guide	**Project Coordinator**	**Head of the Department**
Mr. K.L.Narasimha Rao	**Dr. S. Aparna**	**Mr. S.Phani Kumar**
Assistant Professor **Dept. of CSE**	**Assistant Professor** **Dept. of CSE**	**Professor & HOD** **Dept. of CSE**

ACKNOWLEDGEMENT

Our project report would not have been successful without the help of several people. We would like to thank the personalities who were part of our seminar in numerous ways, those who gave us outstanding support from the birth of the seminar.

We are extremely thankful to our honourable Pro-Vice-Chancellor, **Prof. D. Sambasiva Rao**,for providing the necessary infrastructure and resources for the accomplishment of our seminar.We are highly indebted to **Prof. N. Seetharamaiah**, Associate Director, School of Technology, for his support during the tenure of the seminar.

We are very much obliged to our beloved **Prof. S. Phani Kumar**, Head of the Department of Computer Science & Engineering, for providing the opportunity to undertake this seminar and encouragement in the completion of this seminar.

We hereby wish to express our deep sense of gratitude to **Dr. S Aparna**, Project Coordinator,Department of Computer Science and Engineering, School of Technology and to our guide, **Mr. K. L. Narasimha Rao**, Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the project report.

We are also thankful to all the Computer Science and Engineering department staff members who have cooperated in making our seminar a success. We would like to thank all our parents and friends who extended their help, encouragement, and moral support directly or indirectly in our seminar work.

Sincerely,

M. Saai Vinieth (221910308036)

Bhavin Patel (221910308041)

Pranav Reddy G (221910308044)

TABLE OF CONTENTS

ABSTRACT

Nocturnal video monitoring can be challenging due to changes in surrounding light. This makes it difficult to detect intruders in low light and identify objects or people when exposed to bright light. However, a multi-intensity IR-illuminator can help alleviate these problems by providing constantly changing light intensity. To achieve this, we will annotate the MI3 dataset to improve its ground-truth and train object detection models such as Faster R-CNN, YOLOv5, YOLOv7, and YOLOv8. We will compare the algorithms based on their precision, recall, and mAP metrics. We expect Faster R-CNN to have an average precision of 80, YOLOv5 to have 85, YOLOv7 to have 90, and YOLOv8 to have 92. By using a multi-intensity IR-illuminator and these models, we can greatly enhance the effectiveness of nocturnal video monitoring.

LIST OF FIGURES

SYMBOLS & ABBREVIATIONS

Sno.	Abbreviation	Meaning
1.	AP: Average Precision	A commonly used performance metric in object detection tasks.
2.	IoU: Intersection over Union	A measure of overlap between two bounding boxes.
3.	YOLO: You Only Look Once	State of the art object detection algorithm.
4.	ROI: Region of Interest	A subset of the input image that is proposed for further processing by the model.
5.	RPN: Region Proposal Network.	A component of Faster RCNN that proposes candidate object regions in an image.
6.	FPN: Feature Pyramid Networks	Feature Pyramid Networks are used to extract features at multiple scales.
7.	CNN: Convolutional Neural Network	A type of deep neural network commonly used in computer vision tasks.
8.	SSD: Single Shot Detector	A type of object detection model that performs classification and localization in a single pass.
9.	mAP: mean Average Precision	A variant of AP that averages the precision scores at different recall levels.
10.	CUDA: Compute Unified Device Architecture	A parallel computing platform and programming model developed by NVIDIA.
11.	VOC: Visual Object Classes	A dataset and evaluation metric used for object detection tasks.
12.	MI3: Multi Intensity Infrared Illumination	It is a dataset containing intensity varying video sequences of several indoor and outdoor scenes.

1. INTRODUCTION

Monitoring nighttime video is often challenging due to changes in ambient light. Low illumination makes it difficult to identify distant attackers, while bright light causes overexposure and hinders seeing nearby items. To address these issues, multi-intensity IR-illuminators are designed to adjust the light intensity regularly. Chan et al.[1] developed the MI3 database which includes video sequences of indoor and outdoor scenes with varying brightness levels. The database provides two types of ground truths, including people counting and foreground image pixel tagging without bounding box information. However, MI3 still makes assumptions, such as the absence of foreground in the first 100 frames, which may limit its accuracy. For example, it may not be possible to distinguish a bag from a person holding it, and the foreground ground-truths in the MI3 dataset may combine multiple objects.

The detection of foreground objects in multi-intensity IR videos is typically done using the Gaussian Mixture Model (GMM). However, this method often struggles to accurately handle complex foregrounds. While previous studies have shown qualitatively that multi-intensity illumination can improve image quality for distant and close objects at high and low intensity levels, this article aims to provide a quantitative evaluation of this effect using videos with different illumination intensities or channels[5]. Specifically, we will evaluate deep learning-based object detectors on the MI3 dataset to assess their performance under various lighting conditions and scenes. This study builds on the existing research on deep learning for object detection.

This study will evaluate both single-stage detectors, such as YOLOv5, YOLOv7, and YOLOv8, and two-stage detectors like Faster R-CNN [2]. As infrared images are utilised differently in various applications, it is not feasible to have a universal IR dataset. Instead, we establish a baseline for quantitative evaluation by conducting experiments using pre-trained models of the aforementioned detectors on the MI3 dataset to assess the impact of adopting multi-intensity illumination[3]. We can determine the required number of illumination intensities for object detection at various distances by examining the confidence value of deep learning-based object detection. Additionally, we can identify

a useful range in which one or more light intensities from the multi-intensity IR illuminator can produce reliable detection results[4].

1.1 PROBLEM DEFINITION

The problem addressed by this project is the detection of humans in low-light conditions, specifically during nighttime. Existing object detection systems have difficulty accurately detecting humans in such conditions due to challenges like varying levels of brightness and illumination, and complex foregrounds and object occlusion. This problem has significant implications for surveillance and security applications, where reliable detection of human presence is crucial. Therefore, the goal of this project is to develop an accurate and efficient object detection system that can operate effectively in low-light conditions and provide reliable results. The system should be able to handle the challenges of nighttime detection while achieving high detection accuracy and reducing false positives. The success of this project could lead to improved surveillance and security capabilities in various settings, including law enforcement, public safety, and military applications.

1.2 OBJECTIVE

The objective of this project is to develop an object detection system that can accurately and efficiently detect humans in low-light conditions, specifically at nighttime. The system must be able to handle the challenges posed by varying levels of brightness and illumination, as well as complex foregrounds and object occlusion. By achieving high detection accuracy and reducing false positives, this system can significantly improve surveillance and security in a variety of applications, such as law enforcement, transportation, and urban planning. The use of deep learning-based models, such as Faster R-CNN and YOLO, will be explored to create a robust and reliable detection system.

1.3 LIMITATIONS

The base paper on object detection in low-light conditions presents several limitations that need to be addressed to improve the accuracy and generalizability of the models. The annotation of data was done abruptly, which could lead to mislabeled or incomplete data, affecting the models'

performance. The performance metrics could also be improved by fine-tuning hyperparameters and using data augmentation techniques to generate more diverse data. Additionally, the paper did not explore other state-of-the-art object detection algorithms that could potentially outperform the models used. Lastly, the dataset used in the study may not be representative of real-world scenarios, limiting the models' generalizability. To overcome these limitations, future research could focus on improving the quality of data annotation, optimising performance metrics, exploring alternative algorithms, and using more diverse datasets.

1.4 OUTCOMES

The outcomes of the project would include the development of an accurate and efficient object detection system for detecting humans in low-light conditions, specifically at nighttime. The system should be able to handle the challenges presented by varying levels of brightness and illumination, and provide reliable results even in the presence of complex foregrounds and object occlusion. The project should result in high detection accuracy and reduced false positives, which could help improve surveillance and security in various applications. In addition, the project could lead to the exploration and evaluation of other state-of-the-art object detection algorithms that could potentially outperform the models used in the base paper. Finally, the project could contribute to the advancement of research in the field of computer vision and object detection in low-light conditions.

2. LITERATURE REVIEW

2.1 FUSION OF MULTI-INTENSITY IMAGE FOR DEEP LEARNING-BASED HUMAN AND FACE DETECTION

[1] In nighttime surveillance systems, ordinary IR-illuminators can cause misdetection of faraway objects due to insufficient illumination, and over-exposure of nearby objects due to excessive illumination. However, we have found a solution to this problem with the MI3 image dataset, which uses multi-intensity IR-illumination (MIIR) to provide more accurate detection of objects at varying distances. We have annotated the dataset to ensure its accuracy and used it to evaluate the effectiveness of various object detection methods, including SSD, YOLO, Faster R-CNN, and Mask R-CNN, by analysing their performance at different illumination intensities. By incorporating a tracking scheme and a new fusion method, we have achieved significant improvements in the detection of faces and objects over a wide range of distances. This approach represents a major breakthrough in nighttime surveillance and should be considered the new benchmark for object detection in low-light environments.

2.2 A NOVEL SYNCHRONOUS MULTI-INTENSITY IR ILLUMINATOR HARDWARE IMPLEMENTATION FOR NIGHTTIME SURVEILLANCE

[7] For nighttime surveillance, it's difficult for an ordinary IR illuminator (which has a fixed light intensity) to generate a one-size-fits-all light condition for objects at different distances. Often, we will get an overexposed (underexposed) image for an object which is too close to (too far from) the illuminator/camera in the nighttime. To partly resolve such problems, a multi-intensity IR illuminator was proposed which can emit periodically varying intensities of IR light toward an object. However, two major problems remain to be resolved when using such an illuminator, i.e., the "synchronisation issue" for background modelling and the "viewing issue" for manual monitoring. In this paper, we propose a novel synchronous multi-intensity infrared (SMIIR) illuminator hardware design to help address these two issues. Hopefully this innovation will open the door to a new era of nighttime surveillance.

2.3 FACE RECOGNITION USING BOTH VISIBLE LIGHT IMAGE AND NEAR-INFRARED IMAGE AND A DEEP NETWORK

[9] In recent years, computer vision has seen remarkable advancements through the use of deep neural networks. In particular, face recognition has greatly benefited from this technology. However, achieving high performance in face recognition models based on deep networks requires two critical factors: the structure of the neural network and the quality and quantity of training data. One of the biggest challenges faced by these models is the issue of illumination change. This can have a significant impact on the accuracy of face recognition algorithms in real-world scenarios. Unfortunately, collecting training data that encompasses the various levels of illumination found in the real world is difficult. In order to address this challenge, we propose a novel deep network model that leverages both visible light and near-infrared images to perform face recognition. Near-infrared images are less sensitive to changes in illumination, while visible light images contain important texture information. Our adaptive score fusion strategy and nearest neighbour algorithm ensure that information loss is minimised and final classification is performed accurately.

2.4 ROBUST LICENCE PLATE DETECTION IN NIGHTTIME SCENES USING MULTIPLE INTENSITY IR-ILLUMINATOR

[8] The effectiveness of video surveillance is compromised by poor visibility and low illumination at night. However, accidents are just as likely to occur at night as during the day, so there is still a great need for nighttime surveillance. Unfortunately, using a fixed-intensity infrared light source only works for a limited distance, resulting in underexposure or overexposure of objects that are too far or too close to the light source. This paper proposes an innovative solution to this problem: using a multiple intensity IR-illuminator to extend the effective distance of licence plate detection. By detecting licence plates under different illuminations based on the stroke width of the licence ID, and then integrating the results into a synthesised high dynamic range image, the proposed approach can improve the visibility of both licence plate regions and the background scene. The experimental results demonstrate that this method can effectively increase the monitored area both in depth and width, and enhance the security level of nighttime video surveillance. In other words, this new approach is a

game-changer for nighttime surveillance, improving both its effectiveness and coverage, and ultimately making us all safer.

2.5 FASTER R-CNN: TOWARDS REAL TIME OBJECT DETECTION WITH REGION PROPOSAL NETWORKS

[8] Cutting-edge object detection models rely on region proposal algorithms to guess where objects are located. However, region proposal computation has traditionally been a bottleneck for detection networks, which has been alleviated somewhat by recent advancements such as SPPnet and Fast R-CNN. To solve this problem, we propose the Region Proposal Network (RPN), which shares full-image convolutional features with the detection network, making region proposals nearly cost-free. The RPN is a fully-convolutional network that predicts object bounds and objectness scores at every position, allowing it to generate high-quality region proposals that are used by Fast R-CNN for detection. By training the RPN and Fast R-CNN together to share convolutional features, we have achieved a state-of-the-art detection accuracy of 73.2% mAP on PASCAL VOC 2007 and 70.4% mAP on PASCAL VOC 2012, while maintaining a frame rate of 5fps on a GPU.

2.6 FACE DETECTION IN NIGHTTIME IMAGES USING VISIBLE-LIGHT CAMERA SENSORS WITH TWO-STEP FASTER REGION-BASED CONVOLUTIONAL NEURAL NETWORK

[4] The article presents a novel approach to nighttime face detection using a visible-light camera. Unlike conventional methods that use near-infrared or thermal cameras, the proposed method uses a single visible-light camera that is easily adjustable in terms of intensity and angle. To overcome the challenges of noise and image blur in long-distance night images, the authors propose a Two-Step Faster region-based convolutional neural network (R-CNN) method that sequentially performs body and face detection, and locates the face within the limited body area.The proposed method was evaluated using the Dongguk Nighttime Face Detection database (DNFD-DB1) and an open database from Fudan University. The results showed that the proposed method outperformed other existing face detectors and achieved higher accuracies than those without the proposed preprocessing.

3. PROBLEM ANALYSIS

3.1 PROBLEM STATEMENT

In our project, we aim to optimise object detection using Faster R-CNN, YOLOv5, YOLOv7, and YOLOv8 models by enhancing the MI3 dataset. To achieve this, we will create more detailed annotations to supplement the current ground-truth, which is insufficient for our purposes. This enhancement is expected to improve the accuracy and reliability of the models, leading to superior object detection results. To identify the best algorithm for object detection, we will compare Precision, Recall, and Mean Average Precision scores among the models. By doing so, we can select the most suitable model to detect objects accurately and efficiently.

3.2 EXISTING SYSTEM

In prior research studies, the MI3 dataset has served as the input data for a variety of object detection algorithms, including but not limited to SSD, YOLO, Faster R-CNN, and Mask R-CNN. These models have been implemented to detect and localise objects accurately and efficiently in various real-world scenarios. However, with the advancement in technology and the increasing complexity of object detection tasks, it is necessary to revisit and evaluate the existing models' performance and explore new techniques to achieve even better results.

3.3 FLAWS AND DISADVANTAGES OBSERVED

- Annotation of data was done abruptly and can be improved to enhance the accuracy of the models.
- Performance metrics could be improved by fine-tuning hyperparameters and using data augmentation techniques.
- The base paper did not explore other state-of-the-art object detection algorithms that could potentially outperform the models used.

- The dataset used may not be representative of real-world scenarios, limiting the generalizability of the models.

3.4 PROPOSED SYSTEM

The proposed system aims to utilise state-of-the-art object detection algorithms including Faster R-CNN, YOLOv5, YOLOv7, and YOLOv8 to improve the accuracy and reliability of object detection tasks. Additionally, a script will be developed for each algorithm to enable real-time visualisation of the trained models on a video stream.

The system should also meet non-functional requirements such as accuracy, reliability, and efficiency. The models should achieve high precision and recall scores, as well as high Mean Average Precision (mAP) scores. The models should also be able to process data quickly and efficiently, making use of the available hardware resources effectively.

To achieve these requirements, the system will implement appropriate data annotation techniques, fine-tune hyperparameters, and use data augmentation methods to improve the performance of the models. The system will also utilise appropriate performance metrics to evaluate the models and ensure their accuracy and reliability.

Overall, the proposed system aims to utilise state-of-the-art object detection algorithms and visualisation techniques to improve the accuracy and reliability of object detection tasks, meeting the functional and non-functional requirements of the system.

3.5 FUNCTIONAL REQUIREMENTS

A. Data Annotation:
   - Ability to annotate images and videos with object class labels and bounding boxes.

B. Training Models:
   - Ability to train object detection models on annotated data.

- Ability to fine-tune pre-trained models on new datasets.

C. Validating Models:
    - Ability to validate models on a validation dataset to ensure they are learning correctly.
    - Ability to monitor and visualise the training process and metrics.

D. Testing Models:
    - Ability to test models on a test dataset to evaluate their performance.
    - Ability to generate predictions on new, unseen data.

E. Performance Metrics:
    - Ability to compute performance metrics such as Precision, Recall, and Mean Average Precision to evaluate model performance.

F. Hyperparameter Tuning:
    - Ability to tune hyperparameters such as learning rate, batch size, and number of epochs to optimise model performance.

G. Data Augmentation:
    - Ability to apply data augmentation techniques such as random cropping, flipping, and rotation to increase the diversity of the training data.

H. Model Optimization:
    - Ability to optimise models for inference by reducing their size and complexity, while maintaining high accuracy.

I. Hardware Compatibility:
    - Ability to run models on a range of hardware, including CPUs and GPUs, and support for CUDA acceleration.

J. Generalizability:
    - Ability to generalise to new and unseen data by using diverse datasets and augmenting the existing dataset.

3.6 NON FUNCTIONAL REQUIREMENTS

Non-functional requirements (NFRs) describe the quality features of a software system that are evaluated based on standards other than responsiveness, usability, security, portability, and other important factors.

- Scalability: The models should be scalable, meaning they should be able to handle large datasets and perform well on multiple GPUs.
- Speed: The models should be fast and efficient, able to process images in real-time or near real-time.
- Accuracy: The models should have high accuracy rates in object detection tasks.
- Memory usage: The models should use memory efficiently and avoid memory leaks to prevent crashes and slowdowns.
- Ram Usage: The models should use RAM efficiently to minimise usage and prevent crashes.
- GPU Usage: The models should utilise GPUs efficiently to speed up computations.
- TPU Usage: The models should be optimised for TPUs for even faster computations.
- Runtime: The models should be able to process images within a reasonable time frame.
- Portability: The models should be portable, able to run on different hardware and software configurations.
- Interoperability: The models should be able to interact with other systems and software.
- Security: The models should be secure and protect user data.
- Maintainability: The models should be easy to maintain and update with new features and bug fixes.
- Usability: The models should be user-friendly and easy to use.
- Robustness: The models should be able to handle unexpected inputs and edge cases without crashing or giving incorrect results.

4. SYSTEM DESIGN

4.1 PROPOSED SYSTEM ARCHITECTURE:

4.1.1 Architecture Diagram Of Faster R-CNN

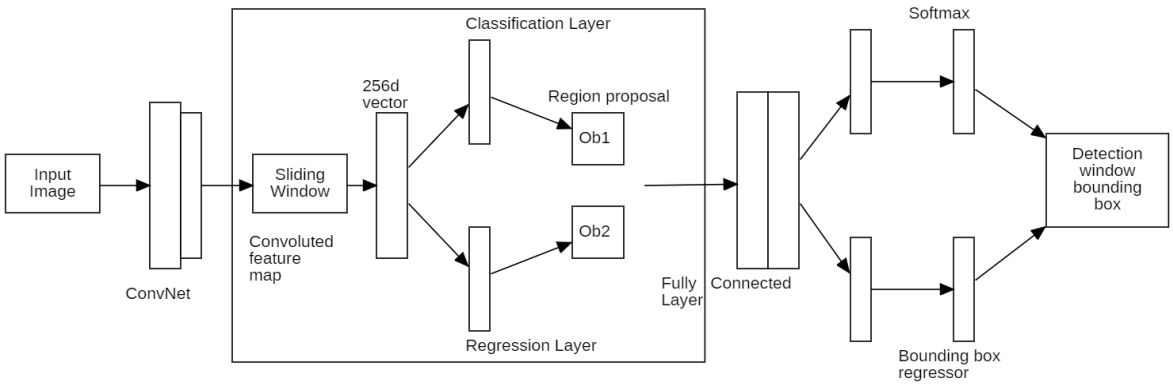


Fig 4.1.1.1 Architecture Diagram Of Faster R-CNN

4.1.2 Architecture Diagram Of YOLOv5

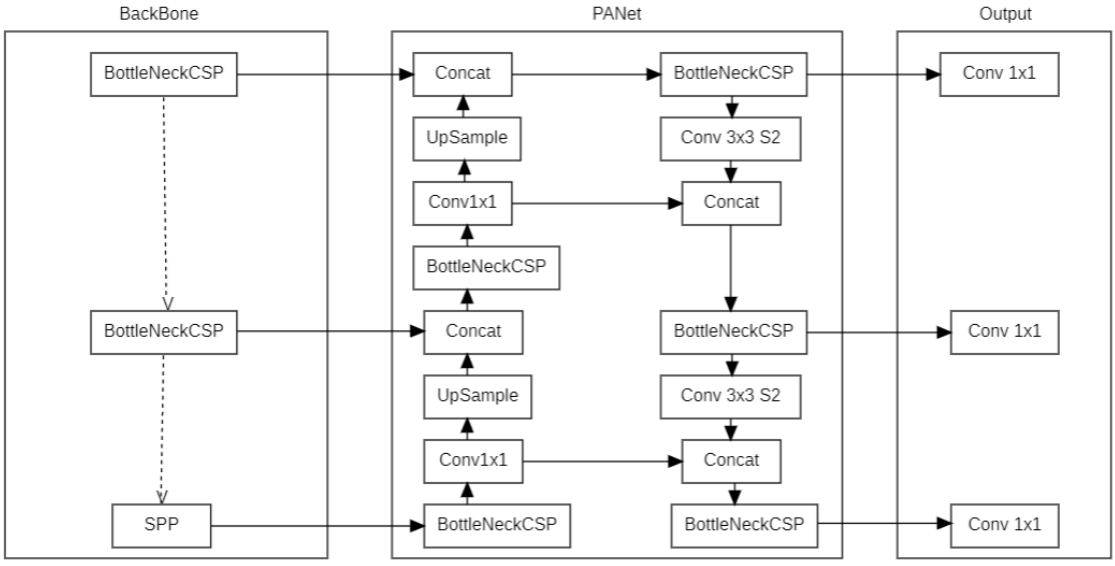


Fig 4.1.2.1 Architecture Diagram Of YOLOv5

4.1.3 Architecture Diagram Of YOLOv7

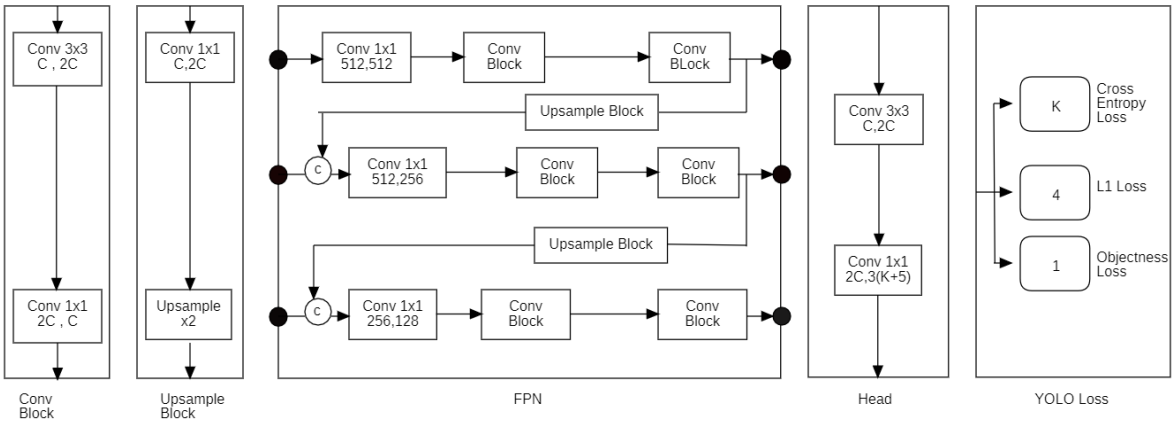


Fig 4.1.3.1 Architecture Diagram Of YOLO v7

4.1.4 Architecture Diagram Of YOLOv8

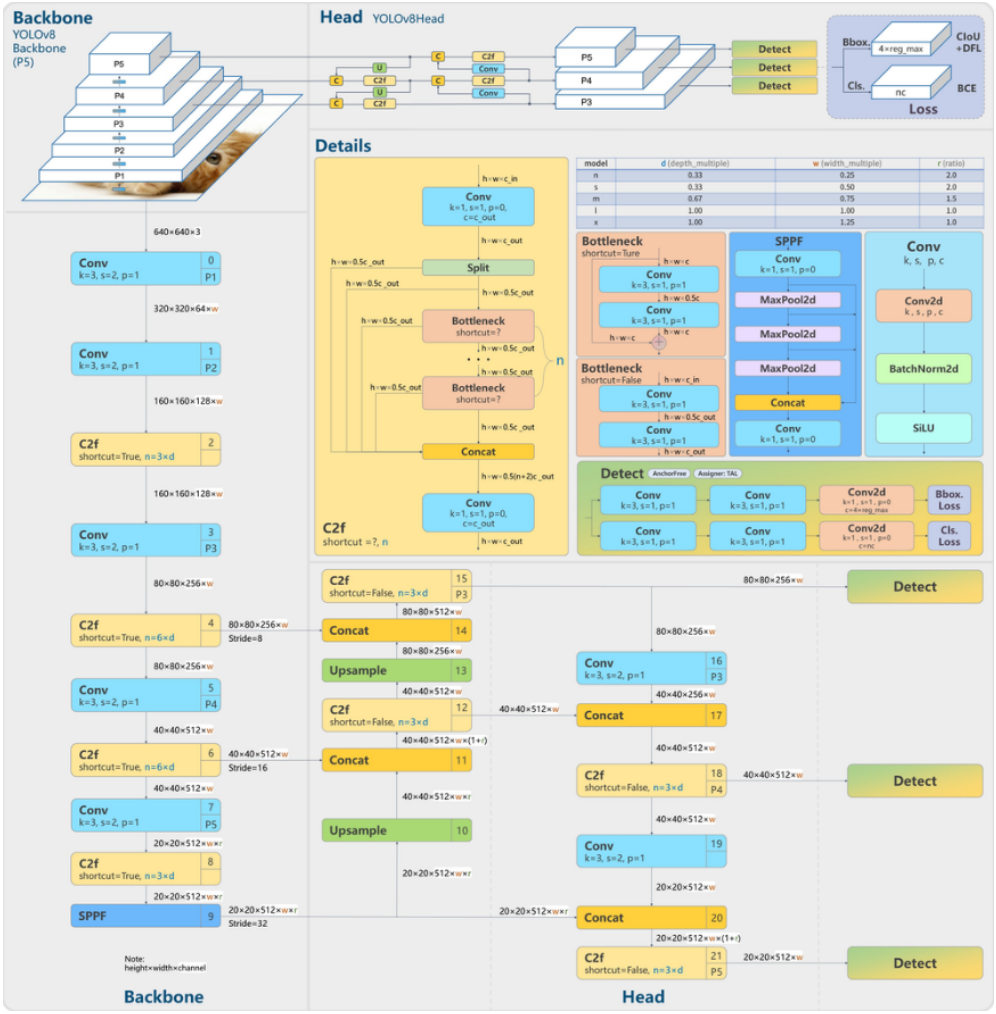


Fig 4.1.4.1 Architecture Diagram Of YOLOv8

4.2 UML DIAGRAMS:

Unified Modeling Language (UML) is a powerful tool that can be used to visualise the design and structure of complex software systems. By creating visual models and diagrams, UML can help developers better understand the relationships and interactions between different components of a system, making it easier to identify potential issues and improve overall performance.

4.2.1 Advantages:

The advantages of using UML-based use case diagrams are:

- Clarity and Communication: Use case diagrams help to clarify system functionality and communicate it to stakeholders effectively.
- Visual Representation: Use case diagrams provide a visual representation of the system's use cases and their relationships, making it easier to understand the system's behaviour.
- Flexibility: UML allows developers to modify and refine the use case diagram during the project, allowing for greater flexibility in the design process.
- Reusability: Developers can reuse use cases in other projects or modify existing ones, saving time and effort.
- Improved Understanding: UML provides a standardised notation that helps developers better understand the system, reducing the risk of misunderstandings and errors.

4.2.2 Use Case Diagram:

Use case diagrams in UML are graphical representations that illustrate the interactions between a system and its users or external entities. They provide a high-level overview of the system's functionality and scope, helping to capture its requirements and identify its actors. In the proposed system, use case diagrams can be used to model the behaviour of the system, including the steps involved in downloading and annotating datasets, training and testing models, and deducing performance metrics.

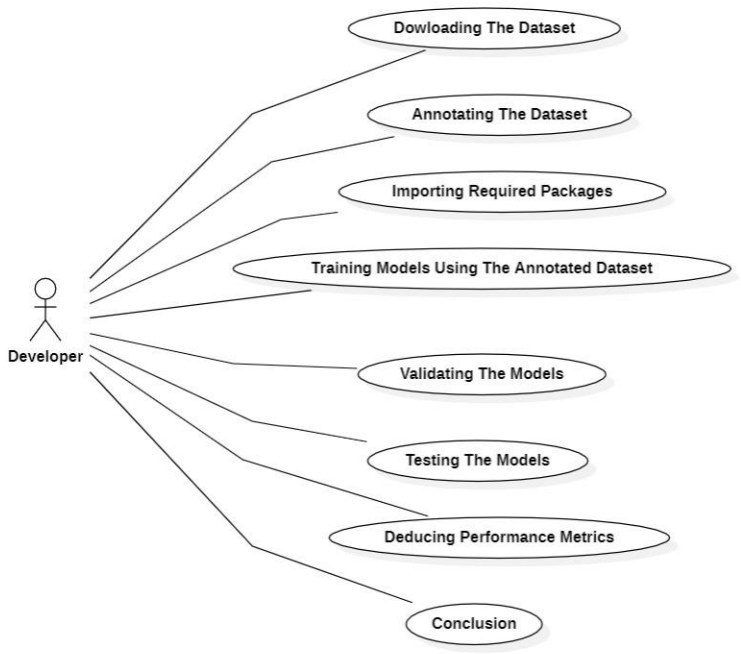


Fig 4.2.2.1 Use Case Diagram

4.2.3 Class Diagram:

Class diagrams in UML are used to model the classes, attributes, methods, and relationships within a system. They provide a visual representation of the structure of the system and help to organise and document the code. Class diagrams also assist in understanding the system's architecture and in communicating it to stakeholders.

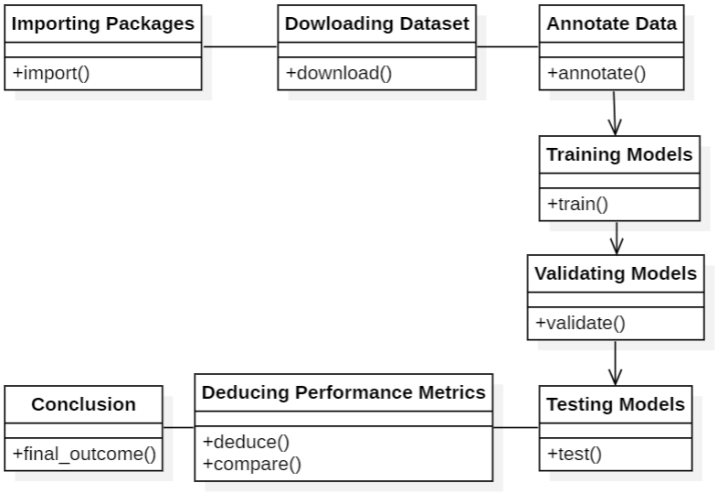


Fig 4.2.3.1 Class Diagram

4.2.4 Activity Diagram:

Activity diagrams in UML are used to model the flow of control in a system, typically focusing on the business and operational processes. They provide a graphical representation of the steps, decisions, and events involved in a process or workflow. Activity diagrams are especially useful in modelling complex systems with multiple processes and activities, as they provide a clear and concise view of the system's behaviour.

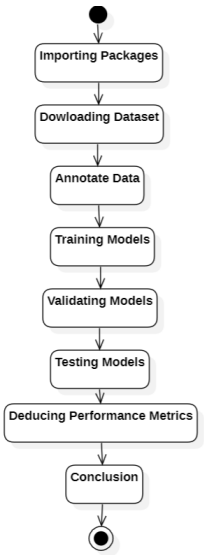


Fig 4.2.4.1 Activity Diagram

4.2.5 Sequence Diagram:

A sequence diagram in UML is a type of interaction diagram that shows how objects interact with each other in a particular scenario or use case. It represents the order of interactions between objects and the messages exchanged between them. Sequence diagrams provide a clear visual representation of the interactions between objects and help developers to understand the flow of control between objects in a system.

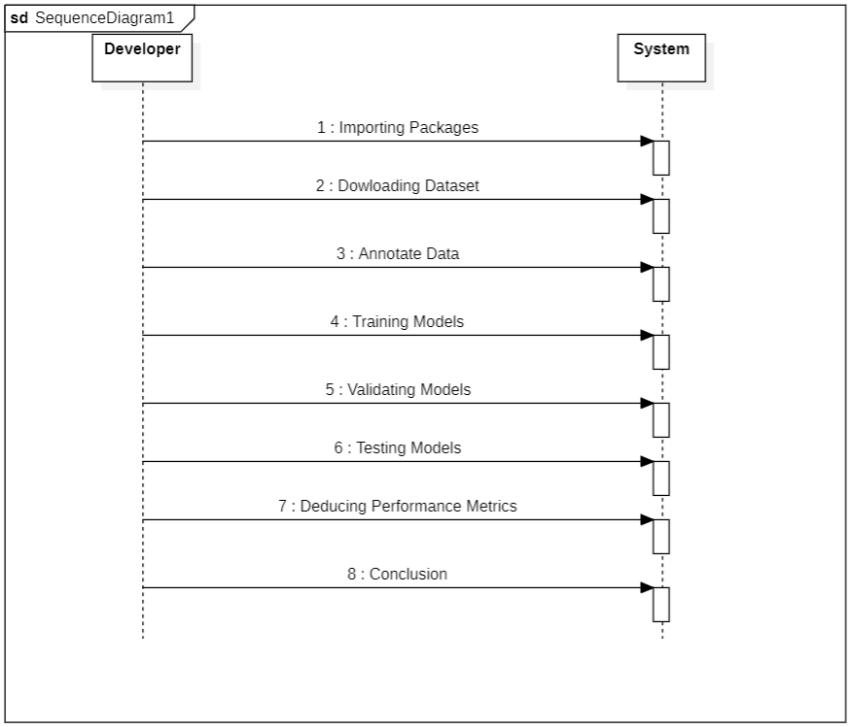


Fig 4.2.5.1 Sequence diagram

5. IMPLEMENTATION

5.1 OVERVIEW OF TECHNOLOGIES

5.1.1 Python

Python is a versatile and widely-used programming language that has gained significant popularity in the data science and machine learning domains. Its extensive range of libraries and frameworks make it a go-to language for developers and researchers working on machine learning projects.

5.1.2 Google Colab

Google Colab is a cloud-based platform for developing and executing machine learning code. It provides a free, easy-to-use environment that allows users to experiment with different models and algorithms without having to worry about configuring their own local machine.

5.1.3 Visual Studio Code

VSCode is a lightweight, open-source code editor that supports various programming languages, including Python. It provides developers with a wide range of features, such as debugging, code completion, and version control integration, that help streamline the development process.

5.1.4 Tensorflow

TensorFlow is an open-source machine learning framework developed by Google. It is commonly used for developing and deploying deep learning applications, including those based on Faster RCNN, YOLOv5, YOLOv7, and YOLOv8. TensorFlow provides users with a range of tools for building, training, and deploying machine learning models, making it a go-to choice for many developers and researchers.

5.1.5 Pytorch

PyTorch is another popular open-source machine learning framework that is widely used for deep learning applications. It is known for its ease-of-use and flexibility, making it a popular choice for developers and researchers who want to experiment with different models and algorithms.

5.1.6 CUDA

CUDA is a parallel computing platform developed by NVIDIA that enables GPUs to perform complex calculations in parallel. It can significantly speed up the training and inference of deep learning models, including those based on Faster RCNN, YOLOv5, YOLOv7, and YOLOv8. This makes it an essential technology for developers who want to train and deploy large-scale deep learning models quickly and efficiently.

5.1.7 OpenCV

OpenCV is an open-source computer vision library that provides tools for image and video processing, as well as feature detection and recognition. It can be used to preprocess images and videos before feeding them into a deep learning model based on Faster RCNN, YOLOv5, YOLOv7, or YOLOv8, helping to improve the accuracy of the model.

5.1.8 Git

Git is a popular version control system that allows developers to track changes to their code and collaborate on projects. It can be used to manage the code for a deep learning project that involves training models based on Faster RCNN, YOLOv5, YOLOv7, or YOLOv8, helping to ensure that code changes are properly documented and tracked.

5.1.9 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows developers to create and share documents that contain live code, equations, visualisations, and narrative text. It is often used to

experiment with different deep learning models and algorithms based on Faster RCNN, YOLOv5, YOLOv7, or YOLOv8, providing an interactive environment that makes it easy to test and refine models.

5.2 WORKFLOW

- Import Packages: Import necessary packages and libraries required for the code, such as TensorFlow or PyTorch.
- Import Dataset: Download the dataset and import it, either locally or through a cloud-based platform like Google Colab.
- Data Preprocessing: Preprocess the data by resizing, augmenting, or normalising the images, as well as converting labels into the appropriate format.
- Split Dataset: Split the dataset into training, validation, and testing sets, using a suitable ratio.
- Building the Model: Choose a suitable architecture, such as Faster RCNN, YOLOv5, YOLOv7, or YOLOv8, and train the model on the training dataset.
- Hyperparameter Tuning: Optimise the hyperparameters of the model, such as the learning rate, batch size, or number of epochs, to achieve better performance.
- Validation: Validate the model on the validation dataset to ensure that it is not overfitting and is generalising well to new data.
- Testing: Use the test dataset to evaluate the performance of the model in terms of accuracy, precision, recall, and F1 score.
- Fine-tuning and Optimization: Fine-tune the model based on the results of testing, and optimise it further for better performance.
- Deployment: Deploy the model for real-world use, either through a web application, mobile app, or embedded device, depending on the requirements.

5.3 LIBRARIES IMPORTED

5.3.1 Tensorflow

TensorFlow, on the other hand, is a deep learning framework that has a strong focus on scalability and production-readiness. It was originally developed by Google and has become one of the most widely used deep learning frameworks in the industry. TensorFlow offers a wide range of tools and libraries for deep learning tasks, such as data preprocessing, model building, and training, as well as deployment and serving. TensorFlow also supports distributed training across multiple GPUs and machines, making it suitable for large-scale deep learning applications. Additionally, TensorFlow can be used with other languages besides Python, such as C++, Java, and Go.

5.3.2 Numpy

NumPy is a library used for numerical computing and array operations. It provides a fast and efficient way to perform mathematical operations on large datasets, making it an essential library for many machine learning tasks. NumPy is particularly useful for tasks such as data preprocessing and feature engineering.

5.3.3 Pandas

Pandas is a library used for data analysis and manipulation. It provides powerful data structures and tools for working with structured data, including tabular data and time-series data. Pandas is widely used in data preprocessing and cleaning, as well as exploratory data analysis.

5.3.4 Scikit – Learn

Scikit-learn is a popular machine learning library used for tasks such as classification, regression, and clustering. It provides a wide range of algorithms and tools for machine learning,

including support vector machines, random forests, and neural networks. Scikit-learn is widely used in industry and research for a wide range of machine learning tasks.

5.3.5 PyTorch

PyTorch is a popular deep learning framework that offers a dynamic computational graph, making it easy to develop and experiment with new neural network architectures. It has gained popularity due to its ease of use, flexibility, and strong community support. PyTorch is designed to work seamlessly with Python, making it easy to integrate with other libraries and tools commonly used in the Python ecosystem, such as NumPy and OpenCV. PyTorch also includes Torchvision, a library that provides datasets, models, and transforms specific to computer vision tasks.

5.3.6 Torchvision

Torchvision is a PyTorch library that provides datasets, models, and transforms specific to computer vision tasks. It provides easy-to-use tools for loading and preprocessing image data, as well as pre-trained models for tasks such as object detection, segmentation, and classification.

5.3.7 Keras

Keras is a high-level deep learning API that can be used with TensorFlow or Theano. It provides a simple and intuitive way to build and train neural networks, making it an excellent choice for beginners and experts alike. Keras also supports a wide range of deep learning architectures and has a strong focus on user experience.

5.4 DATASET

5.4.1 Multi Intensity Illumination Infrared Dataset

The Multi Intensity Illumination Infrared (MI3) dataset is a game-changer for object detection and tracking applications in the infrared spectrum[6]. It offers a remarkable collection of over 7000 annotated frames that capture humans, animals, and vehicles under various lighting conditions. MI3 is an exceptional resource that facilitates the development of cutting-edge computer vision and deep learning algorithms that can detect and track objects with great accuracy and speed. What sets this dataset apart is its use of a multi-intensity IR illuminator, which enhances the visibility of objects even in low-light conditions. MI3 is an indispensable tool for researchers and practitioners seeking to advance the field of video monitoring applications [10]. The wealth of data and unique features of this dataset make it a valuable resource that is sure to drive innovation and transform the way we approach object detection and tracking in the infrared spectrum.

Fig 5.4.1.1 Sample Dataset Image

5.4.2 Annotations

Annotations are an essential component of object detection algorithms that enable them to learn and identify objects in images or videos. Accurate and comprehensive annotations are crucial for training algorithms to detect and classify objects with high precision and recall. The Multi Intensity Illumination Infrared (MI3) dataset, which consists of infrared videos captured in various lighting conditions, includes over 7000 annotated frames containing a wide range of object categories such as humans, vehicles, and animals. The annotations for this dataset were created using Roboflow, a tool that simplifies the annotation process by providing an easy-to-use interface for labelling objects in images or videos.

5.5 ALGORITHMS IMPLEMENTED

5.5.1 Region Proposal Networks (RPNs)

RPNs are used in Faster R-CNN to generate object proposals before object detection. They are a type of neural network that takes an input image and outputs a set of object proposals, which are regions in the image that are likely to contain objects.

5.5.2 Feature Pyramid Networks (FPNs)

FPNs are used in many object detection models, including YOLOv5 and YOLOv8, to extract features at multiple scales. They are a type of neural network that combines features from different layers of a CNN to create a feature pyramid, which can then be used for object detection at different scales.

5.5.3 Anchor Boxes

Anchor boxes are used in many object detection models, including YOLO, to predict the location and size of objects. They are a set of predefined boxes of different sizes and aspect ratios that

are placed on a grid over the image, and the model predicts the offset from each anchor box to the corresponding object.

5.6 ARCHITECTURES IMPLEMENTED

5.6.1 Faster R-CNN

Faster R-CNN: Faster R-CNN is an object detection architecture that consists of two main components: the Region Proposal Network (RPN) and the Fast R-CNN detector. The RPN generates a set of region proposals that are likely to contain objects, and the Fast R-CNN detector uses these proposals to classify and refine the object detections. The RPN is a fully convolutional network that predicts objectness scores and bounding box offsets for each anchor box in an image. The Fast R-CNN detector uses the proposed regions as input and extracts features using a region of interest (RoI) pooling layer, followed by fully connected layers for classification and regression. Faster R-CNN has been widely used in various computer vision tasks, such as object detection, instance segmentation, and tracking.

5.6.2 YOLOv5

YOLOv5 is a popular single-stage object detection architecture that is known for its small and efficient model architecture. It uses anchor boxes to improve the accuracy of object detection and classification. YOLOv5 has a backbone network consisting of a series of convolutional layers and a neck network that merges features from different layers. It uses a set of convolutional layers to predict objectness scores and bounding box offsets for each anchor box. YOLOv5 is capable of achieving high accuracy with fast inference times, making it suitable for real-time applications.

5.6.3 YOLOv7

YOLOv7 is an improved version of YOLOv5 that is known for its better accuracy and faster training times. YOLOv7 uses a multi-scale feature fusion module that combines features from different layers of the neural network to improve object detection accuracy. The architecture uses anchor boxes to predict objectness scores and bounding box offsets for each anchor box. YOLOv7 also has a backbone network consisting of a series of convolutional layers and a neck network that merges features from different layers.

5.6.4 YOLOv8

YOLOv8 is another improvement over previous versions of YOLO, with even better accuracy and faster training times. YOLOv8 uses a novel backbone network called the Efficient Channel Attention (ECA) network, which improves the efficiency and accuracy of the model by learning to focus on important feature channels. It also uses anchor boxes to predict objectness scores and bounding box offsets for each anchor box. YOLOv8 has a neck network that merges features from different layers of the neural network and predicts objectness scores and bounding box offsets for each anchor box. YOLOv8 is a state-of-the-art object detection architecture and has been widely used in various computer vision tasks.

6. TESTING & VALIDATION

6.1 SYSTEM TESTING

6.1.1 Accuracy Testing

This involves measuring the accuracy of the models on a test dataset by comparing the predicted labels to the ground truth labels. Various metrics can be used to evaluate accuracy, such as mean average precision (mAP) or intersection over union (IoU).

6.1.2 Hyperparameter Tuning

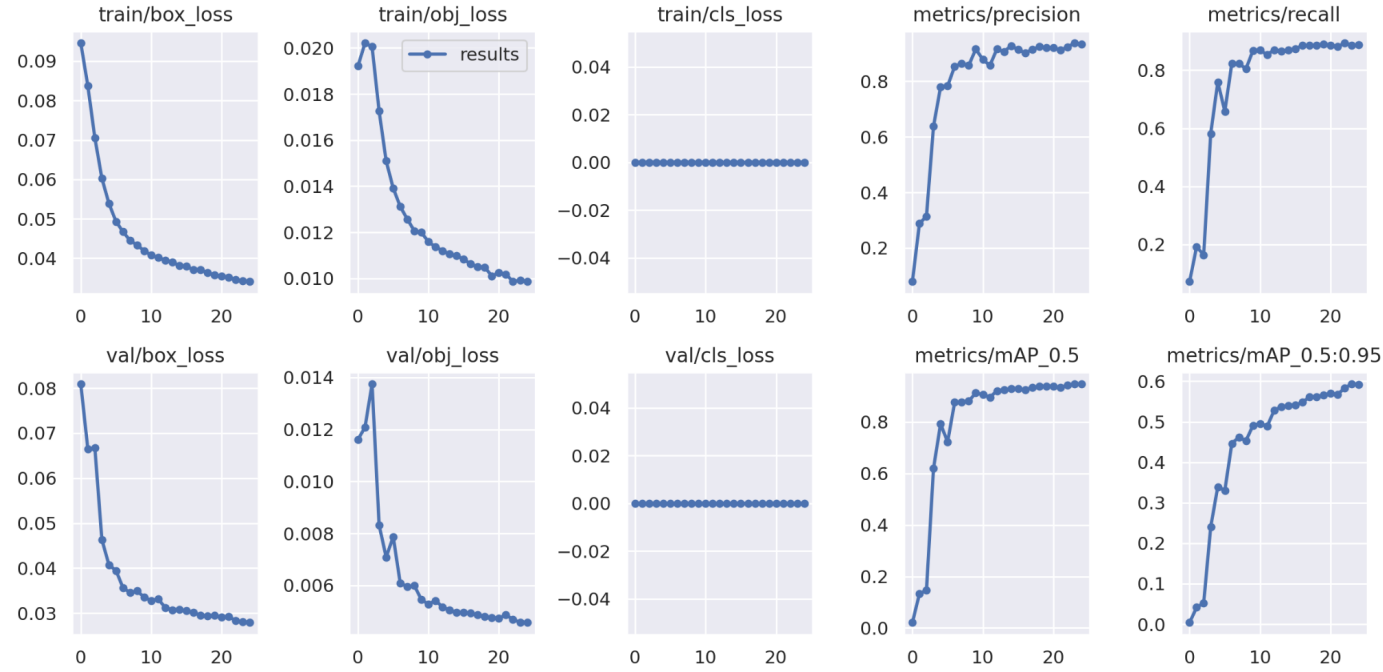
This involves adjusting the model's hyperparameters, such as learning rate, batch size, and optimizer, to optimise its performance on the test dataset. Hyperparameter tuning can be done using techniques such as grid search or random search.

6.1.3 Performance Metrics

- Mean Average Precision (mAP): The most commonly used metric for evaluating object detection models, mAP measures the average precision across different intersections over union (IoU) thresholds for object detection. It ranges from 0 to 1, where higher values indicate better performance.
- Intersection over Union (IoU): This metric measures the overlap between the predicted bounding box and the ground truth bounding box. IoU ranges from 0 to 1, where 1 indicates a perfect overlap.
- Precision: The ratio of true positive detections to the total number of predicted detections. It measures how accurate the model is in predicting the presence of an object.
- Recall: The ratio of true positive detections to the total number of ground truth objects. It measures how well the model is able to detect all instances of an object in the image.

- F1 score: The harmonic mean of precision and recall, F1 score is a popular metric for evaluating binary classification tasks like object detection. It ranges from 0 to 1, where higher values indicate better performance.

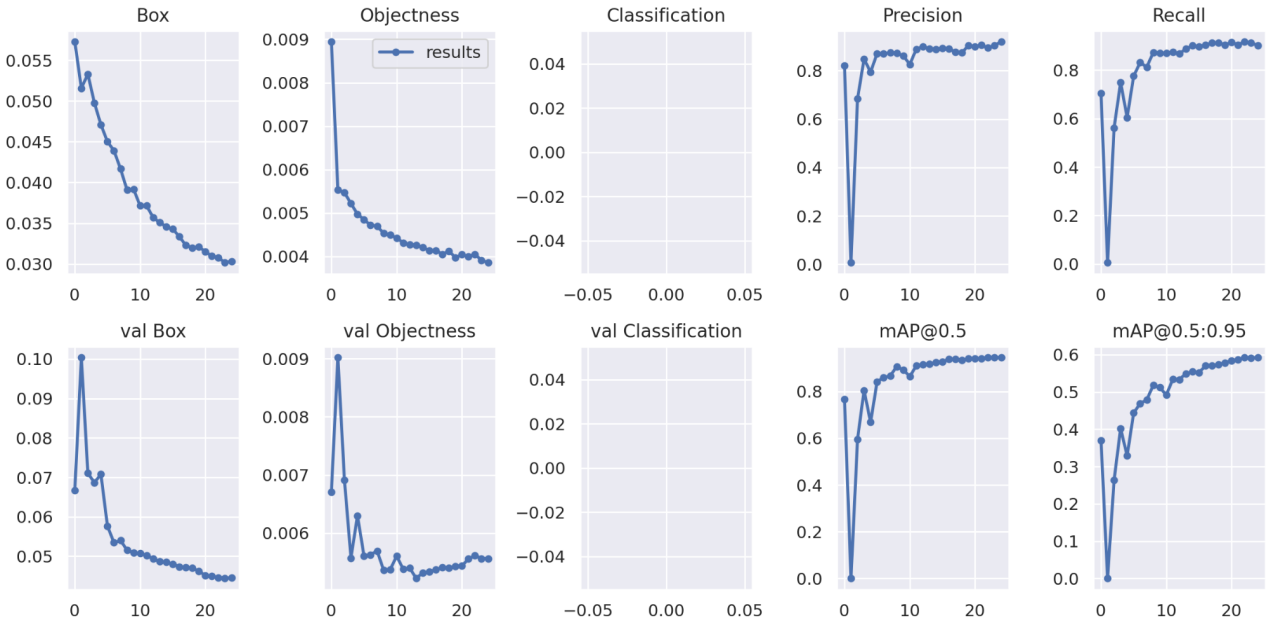
6.2 PERFORMANCE METRICS FOR YOLOv5



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP_0.5:0.95	val/box_loss	val/obj_loss	val/cls_loss	x/lr0	x/lr1	x/lr2
2	0	0.094592	0.019212	0	0.079287	0.07245	0.020846	0.0040552	0.080923	0.01161	0	0.070082	0.0033242	0.0033242
3	1	0.083776	0.020215	0	0.28775	0.19256	0.13216	0.0415	0.066494	0.012089	0	0.039819	0.0063939	0.0063939
4	2	0.070535	0.020051	0	0.31464	0.16397	0.14715	0.052558	0.066712	0.01375	0	0.0092909	0.0091996	0.0091996
5	3	0.060269	0.017261	0	0.638	0.58198	0.62014	0.24069	0.04632	0.008322	0	0.008812	0.008812	0.008812
6	4	0.053874	0.015098	0	0.77946	0.75929	0.79286	0.33942	0.040711	0.0070853	0	0.008812	0.008812	0.008812
7	5	0.049334	0.013916	0	0.78296	0.65855	0.7225	0.3306	0.039464	0.0078732	0	0.008416	0.008416	0.008416
8	6	0.046712	0.013114	0	0.8532	0.82364	0.87673	0.44646	0.035615	0.006083	0	0.00802	0.00802	0.00802
9	7	0.044501	0.012571	0	0.86479	0.82309	0.87647	0.46197	0.034568	0.0059567	0	0.007624	0.007624	0.007624
10	8	0.043354	0.012059	0	0.85685	0.80455	0.88153	0.45321	0.034916	0.0059931	0	0.007228	0.007228	0.007228
11	9	0.041882	0.011994	0	0.91598	0.8678	0.91283	0.49093	0.033541	0.0054499	0	0.006832	0.006832	0.006832
12	10	0.040769	0.011585	0	0.87791	0.86845	0.90588	0.49537	0.032797	0.0052783	0	0.006436	0.006436	0.006436
13	11	0.040176	0.011363	0	0.8567	0.85319	0.89662	0.48853	0.033114	0.0054198	0	0.00604	0.00604	0.00604
14	12	0.039542	0.0112	0	0.91618	0.87	0.921	0.52776	0.031205	0.0051679	0	0.005644	0.005644	0.005644
15	13	0.038985	0.011054	0	0.90575	0.86571	0.92505	0.53733	0.030689	0.0050427	0	0.005248	0.005248	0.005248
16	14	0.038179	0.010987	0	0.92804	0.86845	0.93041	0.5397	0.030864	0.0049657	0	0.004852	0.004852	0.004852
17	15	0.037927	0.010821	0	0.9137	0.87417	0.92868	0.54169	0.030578	0.0049604	0	0.004456	0.004456	0.004456
18	16	0.037114	0.010626	0	0.90228	0.88463	0.92424	0.54793	0.030152	0.0049303	0	0.00406	0.00406	0.00406
19	17	0.037035	0.01049	0	0.91392	0.88562	0.93468	0.5612	0.029538	0.0048766	0	0.003664	0.003664	0.003664
20	18	0.036305	0.010482	0	0.92467	0.88513	0.93831	0.56183	0.029391	0.0048011	0	0.003268	0.003268	0.003268
21	19	0.035718	0.010099	0	0.92113	0.88894	0.9398	0.56589	0.029461	0.0047546	0	0.002872	0.002872	0.002872
22	20	0.035477	0.010253	0	0.92114	0.88465	0.93922	0.57033	0.029146	0.0047337	0	0.002476	0.002476	0.002476
23	21	0.035134	0.010179	0	0.91212	0.8806	0.93442	0.56729	0.029283	0.0048684	0	0.00208	0.00208	0.00208
24	22	0.034631	0.0098596	0	0.92317	0.89342	0.94236	0.5833	0.028282	0.0046872	0	0.001684	0.001684	0.001684
25	23	0.034234	0.00992	0	0.93754	0.88513	0.9468	0.59336	0.028033	0.0045773	0	0.001288	0.001288	0.001288
26	24	0.034091	0.0098702	0	0.93383	0.88792	0.94782	0.59196	0.027911	0.0045712	0	0.000892	0.000892	0.000892

Fig 6.2.1 Performance Metrics for YOLOv5

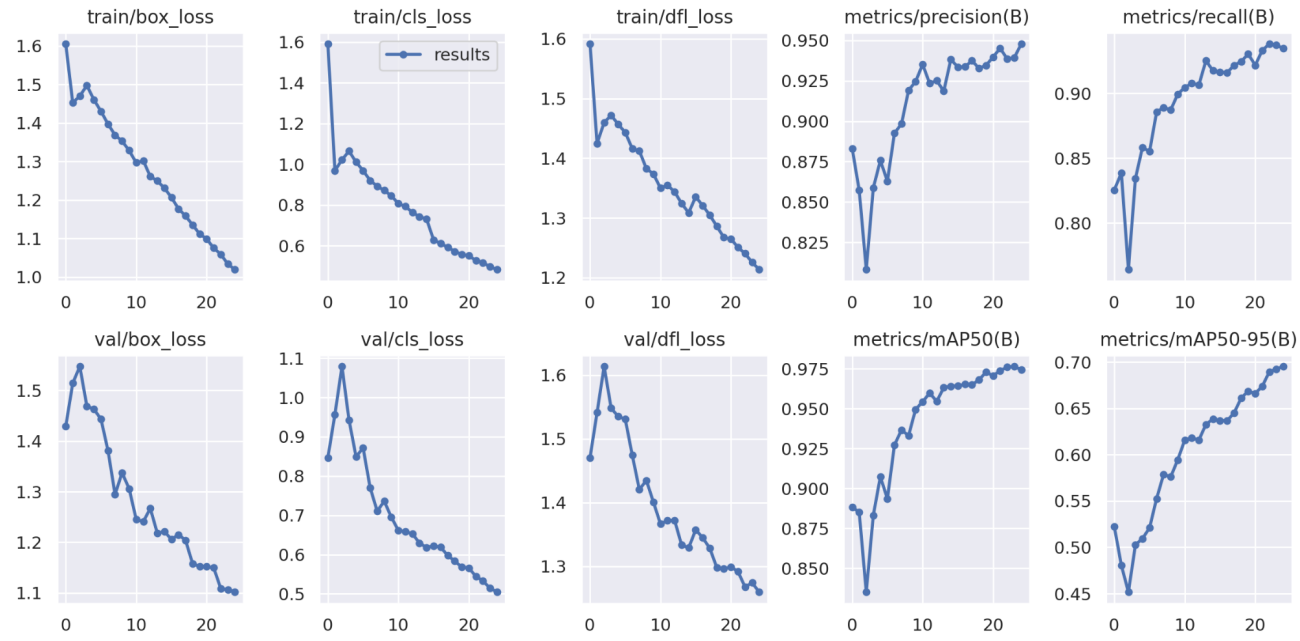
6.3 PERFORMANCE METRICS FOR YOLOv7



0/24	9.94G	0.05728	0.008943	0	0.06622	19	640	0.8206	0.7045	0.7661	0.3699	0.06671	0.006707	0
1/24	10.5G	0.05158	0.005535	0	0.05712	42	640	0.007067	0.00572	0.000256	5.012e-05	0.1004	0.009021	0
2/24	10.5G	0.0533	0.005471	0	0.05877	28	640	0.6841	0.562	0.5961	0.2636	0.07105	0.006916	0
3/24	10.5G	0.04977	0.005219	0	0.05499	37	640	0.8483	0.7493	0.8053	0.4022	0.06869	0.005571	0
4/24	10.5G	0.04709	0.004972	0	0.05206	25	640	0.7938	0.6044	0.6686	0.3287	0.07077	0.006299	0
5/24	10.5G	0.04501	0.00485	0	0.04986	37	640	0.871	0.7755	0.8421	0.4438	0.05763	0.005607	0
6/24	10.5G	0.04391	0.004718	0	0.04862	38	640	0.8712	0.8317	0.8605	0.4692	0.05345	0.005628	0
7/24	10.5G	0.04167	0.004698	0	0.04637	22	640	0.8741	0.8122	0.8672	0.4786	0.05401	0.005693	0
8/24	10.5G	0.03909	0.004533	0	0.04362	36	640	0.8715	0.8726	0.9068	0.5179	0.05157	0.005366	0
9/24	10.5G	0.03918	0.004493	0	0.04367	39	640	0.8621	0.87	0.8935	0.5118	0.05083	0.005368	0
10/24	10.5G	0.03715	0.004427	0	0.04158	22	640	0.8257	0.8704	0.8644	0.492	0.0507	0.005604	0
11/24	10.5G	0.03715	0.004314	0	0.04147	23	640	0.8884	0.8742	0.9101	0.5339	0.05025	0.005381	0
12/24	10.5G	0.0357	0.004276	0	0.03997	23	640	0.8996	0.8674	0.916	0.5336	0.04936	0.005398	0
13/24	10.5G	0.03509	0.004254	0	0.03934	30	640	0.8907	0.888	0.9182	0.5492	0.04862	0.00522	0
14/24	10.5G	0.03457	0.004213	0	0.03878	22	640	0.8876	0.9004	0.9242	0.5541	0.04851	0.005318	0
15/24	10.5G	0.03429	0.004132	0	0.03842	30	640	0.8932	0.897	0.9281	0.552	0.04805	0.005335	0
16/24	10.5G	0.03337	0.00414	0	0.03751	25	640	0.8905	0.9037	0.9387	0.57	0.04738	0.005368	0
17/24	10.5G	0.0323	0.004053	0	0.03636	19	640	0.8771	0.9116	0.938	0.5702	0.04714	0.005406	0
18/24	10.5G	0.032	0.004124	0	0.03613	25	640	0.8748	0.9133	0.9343	0.5735	0.04697	0.005403	0
19/24	10.5G	0.03209	0.003976	0	0.03606	34	640	0.9037	0.9037	0.9403	0.5785	0.04624	0.005428	0
20/24	10.5G	0.03152	0.004042	0	0.03556	22	640	0.8988	0.9141	0.9416	0.584	0.0451	0.005435	0
21/24	10.5G	0.03095	0.003997	0	0.03495	32	640	0.9051	0.9046	0.9417	0.5864	0.04493	0.005555	0
22/24	10.5G	0.03078	0.00405	0	0.03483	26	640	0.894	0.9171	0.945	0.5925	0.04462	0.005613	0
23/24	10.5G	0.03016	0.003913	0	0.03408	28	640	0.9041	0.9122	0.9459	0.591	0.04439	0.005563	0
24/24	10.5G	0.03034	0.003857	0	0.0342	34	640	0.9194	0.902	0.9457	0.5924	0.04457	0.005562	0

Fig 6.3.1 Performance Metrics forYOLOv7

6.4 PERFORMANCE METRICS FOR YOLOv8



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
2	0	1.6054	1.5907	1.5915	0.88322	0.82555	0.88831	0.52233	1.4287	0.84694	1.4704	0.070082	0.0033242	0.0033242
3	1	1.4523	0.96859	1.4247	0.85723	0.83857	0.88509	0.48054	1.5142	0.95672	1.5413	0.039819	0.0063939	0.0063939
4	2	1.47	1.0219	1.4598	0.80819	0.76406	0.83508	0.45165	1.5473	1.0791	1.6139	0.0092909	0.0091996	0.0091996
5	3	1.4974	1.0646	1.4719	0.85874	0.83448	0.88314	0.50273	1.4687	0.94272	1.5483	0.008812	0.008812	0.008812
6	4	1.4604	1.0107	1.4568	0.87568	0.85844	0.90731	0.50945	1.4627	0.84909	1.5352	0.008812	0.008812	0.008812
7	5	1.4303	0.96896	1.4432	0.86286	0.8551	0.89349	0.52093	1.4434	0.87149	1.531	0.008416	0.008416	0.008416
8	6	1.3971	0.92022	1.416	0.89253	0.88561	0.92711	0.55259	1.381	0.77015	1.4743	0.00802	0.00802	0.00802
9	7	1.3685	0.89235	1.4122	0.89839	0.8892	0.93667	0.57879	1.2946	0.71058	1.4205	0.007624	0.007624	0.007624
10	8	1.3528	0.8715	1.3826	0.91903	0.88728	0.93297	0.57597	1.3372	0.73586	1.4349	0.007228	0.007228	0.007228
11	9	1.3283	0.8455	1.3736	0.92459	0.89943	0.94946	0.59444	1.3058	0.69532	1.4006	0.006832	0.006832	0.006832
12	10	1.2967	0.80703	1.3498	0.93512	0.90467	0.95429	0.61586	1.2453	0.66156	1.3664	0.006436	0.006436	0.006436
13	11	1.3021	0.79321	1.3548	0.92339	0.90777	0.95972	0.61808	1.2407	0.65876	1.3719	0.00604	0.00604	0.00604
14	12	1.261	0.7633	1.3438	0.92509	0.90651	0.95449	0.61579	1.2668	0.65292	1.3721	0.005644	0.005644	0.005644
15	13	1.2493	0.74279	1.3241	0.91861	0.92531	0.96312	0.63261	1.218	0.62914	1.3333	0.005248	0.005248	0.005248
16	14	1.2307	0.73146	1.3087	0.93825	0.91754	0.96404	0.63855	1.2214	0.61809	1.3289	0.004852	0.004852	0.004852
17	15	1.2065	0.62941	1.3355	0.9335	0.91661	0.96431	0.63695	1.2057	0.62245	1.3573	0.004456	0.004456	0.004456
18	16	1.177	0.6105	1.3204	0.9337	0.91611	0.96523	0.63664	1.2152	0.61844	1.3446	0.00406	0.00406	0.00406
19	17	1.1584	0.59338	1.3047	0.93746	0.92167	0.96486	0.64512	1.2041	0.59792	1.3281	0.003664	0.003664	0.003664
20	18	1.1352	0.57179	1.2859	0.93275	0.92469	0.96802	0.66111	1.158	0.58325	1.2976	0.003268	0.003268	0.003268
21	19	1.1117	0.5584	1.2672	0.93458	0.93041	0.97283	0.66858	1.1523	0.56811	1.2959	0.002872	0.002872	0.002872
22	20	1.0985	0.55067	1.2647	0.93974	0.92178	0.97054	0.666	1.152	0.56579	1.299	0.002476	0.002476	0.002476
23	21	1.0749	0.52818	1.2508	0.94521	0.93327	0.97379	0.67398	1.1496	0.54446	1.2915	0.00208	0.00208	0.00208
24	22	1.0588	0.51694	1.2403	0.9385	0.93826	0.97589	0.68945	1.1087	0.5328	1.2674	0.001684	0.001684	0.001684
25	23	1.0333	0.49683	1.2257	0.93933	0.93728	0.9764	0.69257	1.1067	0.51451	1.2741	0.001288	0.001288	0.001288
26	24	1.0191	0.48328	1.2136	0.94787	0.9347	0.9742	0.69528	1.1017	0.50444	1.2594	0.000892	0.000892	0.000892

Fig 6.4.1 Performance Metrics for YOLOv8

6.5 CONFUSION MATRIX FOR YOLOv5

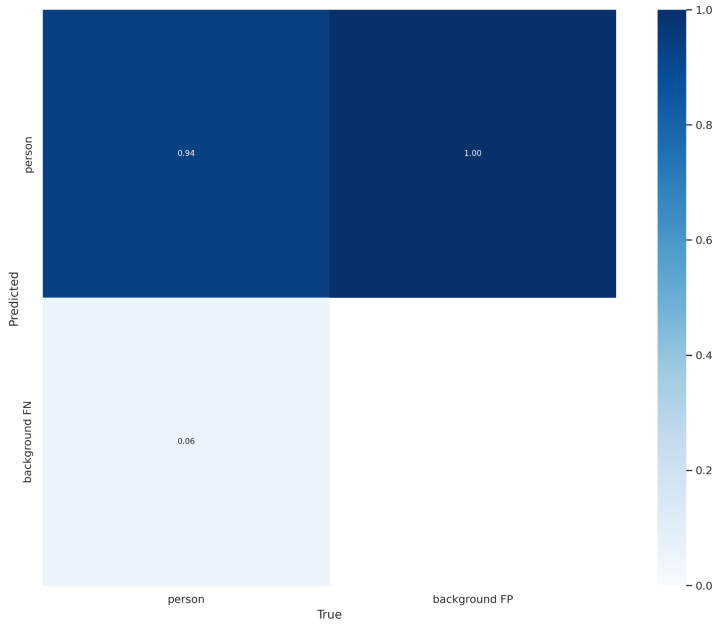


Fig 6.5.1 Confusion Matrix for YOLOv5

6.6 CONFUSION MATRIX FOR YOLOv7

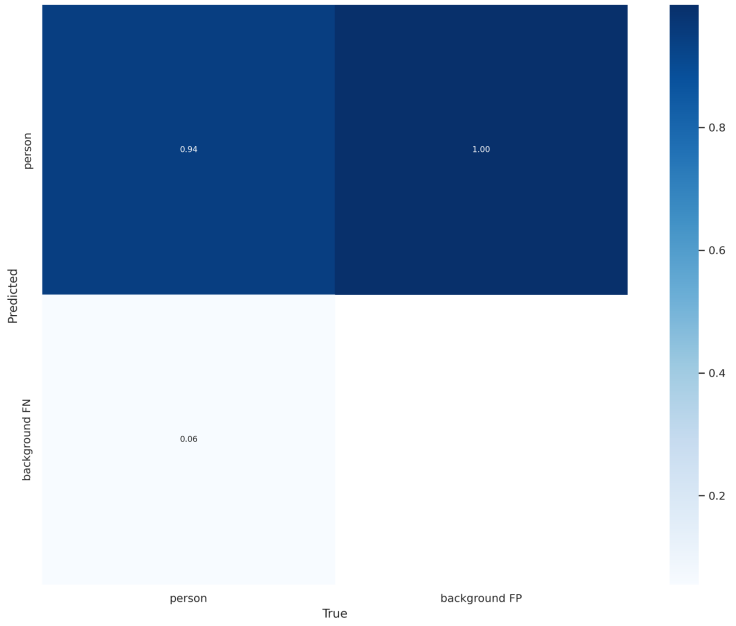


Fig 6.6.1 Confusion Matrix for YOLOv7

6.7 CONFUSION MATRIX FOR YOLOv8

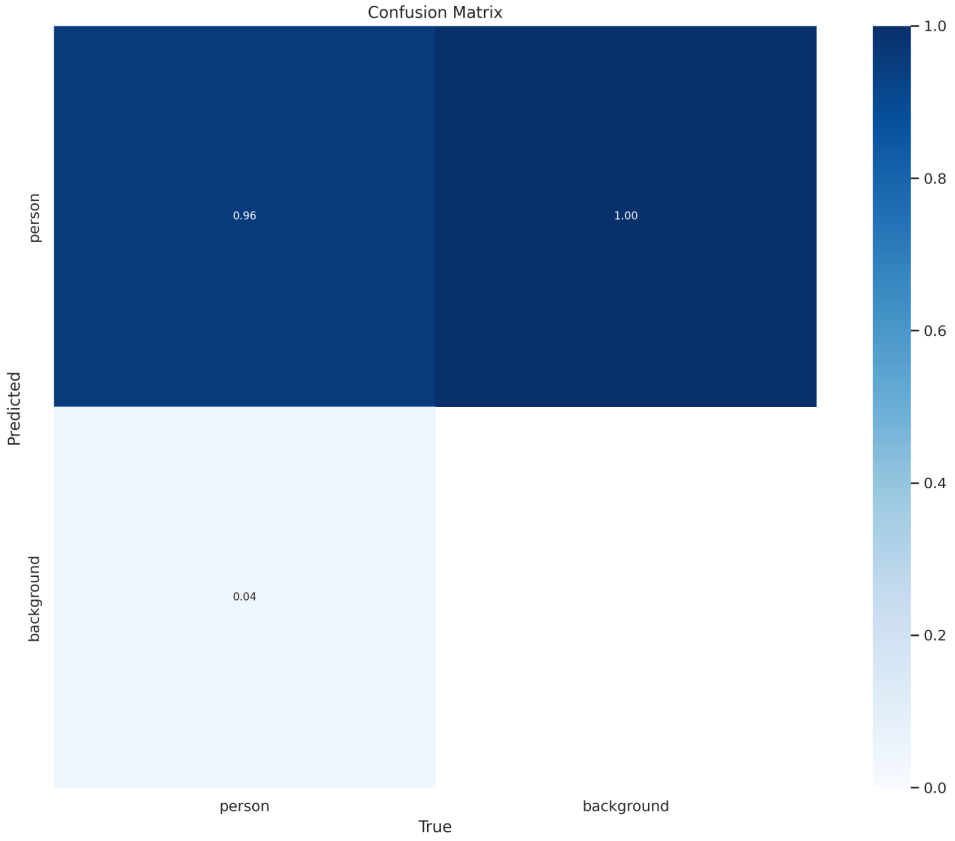


Fig 6.7.1 Confusion Matrix for YOLOv8

7. RESULT ANALYSIS

7.1 TEST RESULTS FOR FASTER R-CNN

Fig 7.1.1 Test Results for Faster R-CNN

7.2 TEST RESULTS FOR YOLOv5

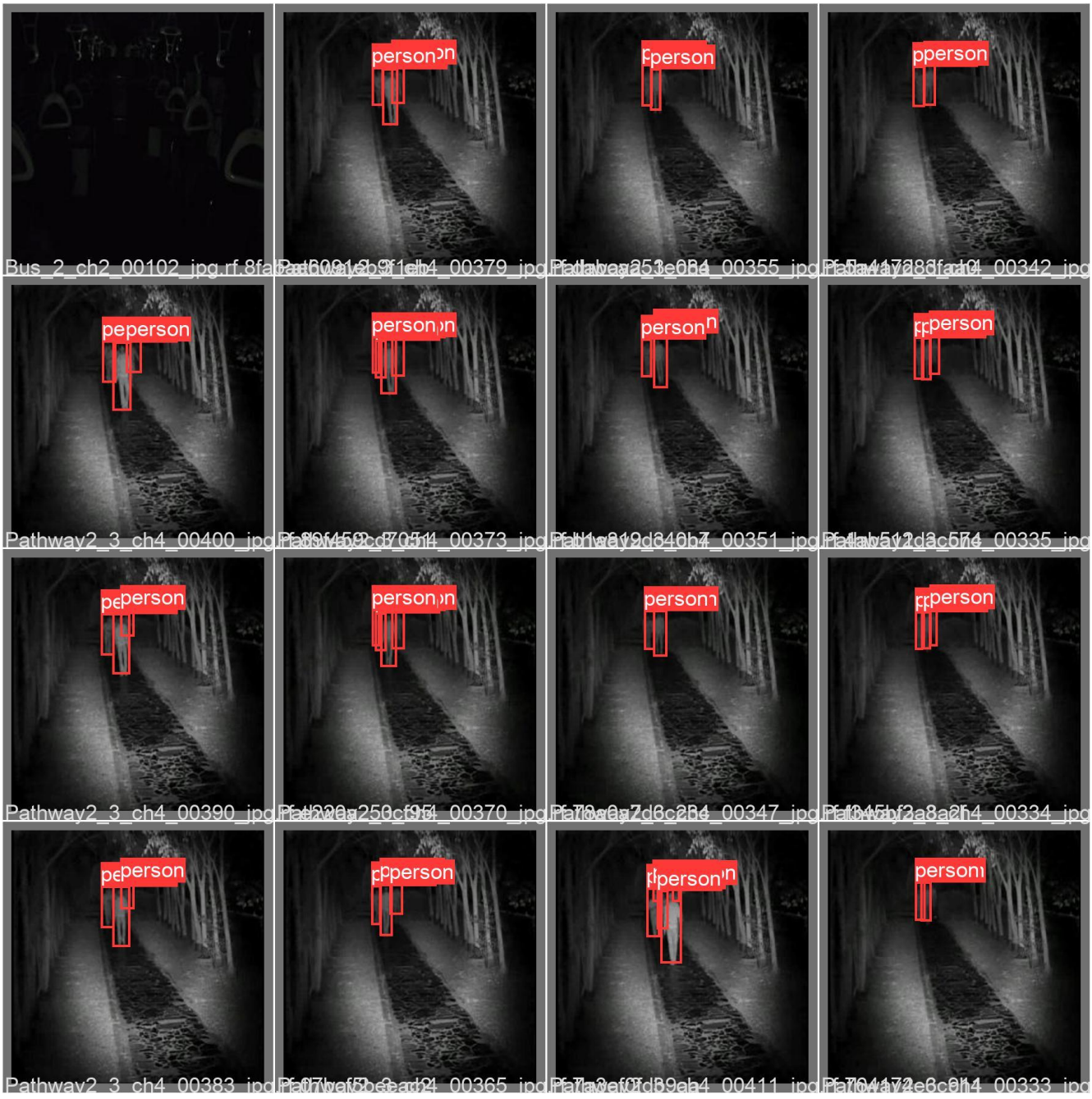


Fig 7.2.1 Test Results for YOLOv5

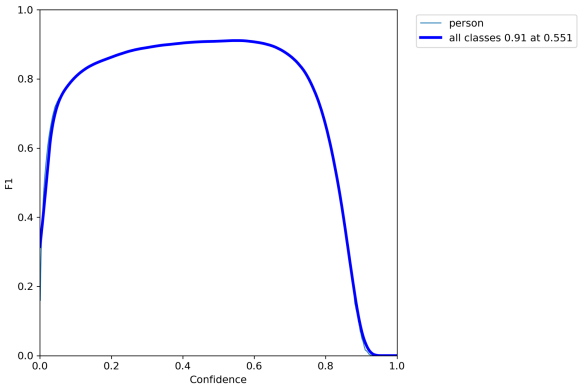


Fig 7.2.2 F1_curve for YOLOv5

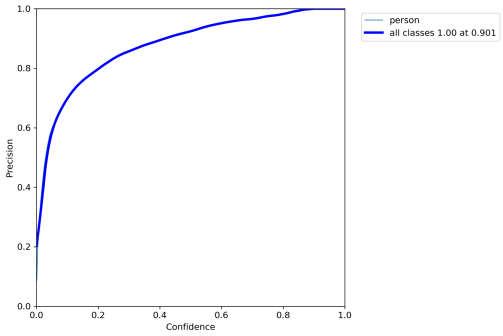


Fig 7.2.3 P_curve for YOLOv5

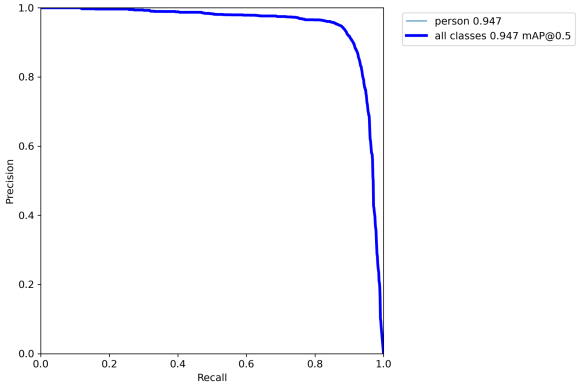


Fig 7.2.4 PR_curve for YOLOv5

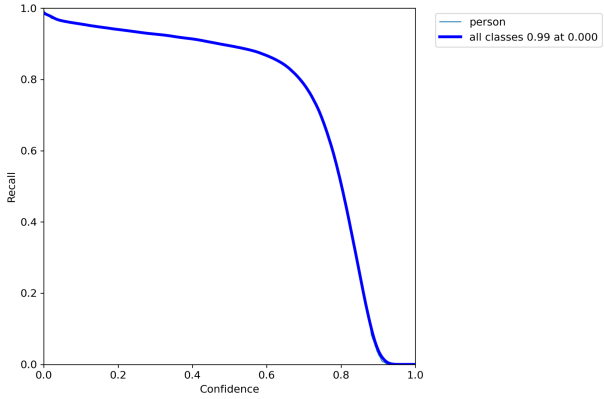


Fig 7.2.5 R_curve for YOLOv5

7.3 TEST RESULTS FOR YOLOv7

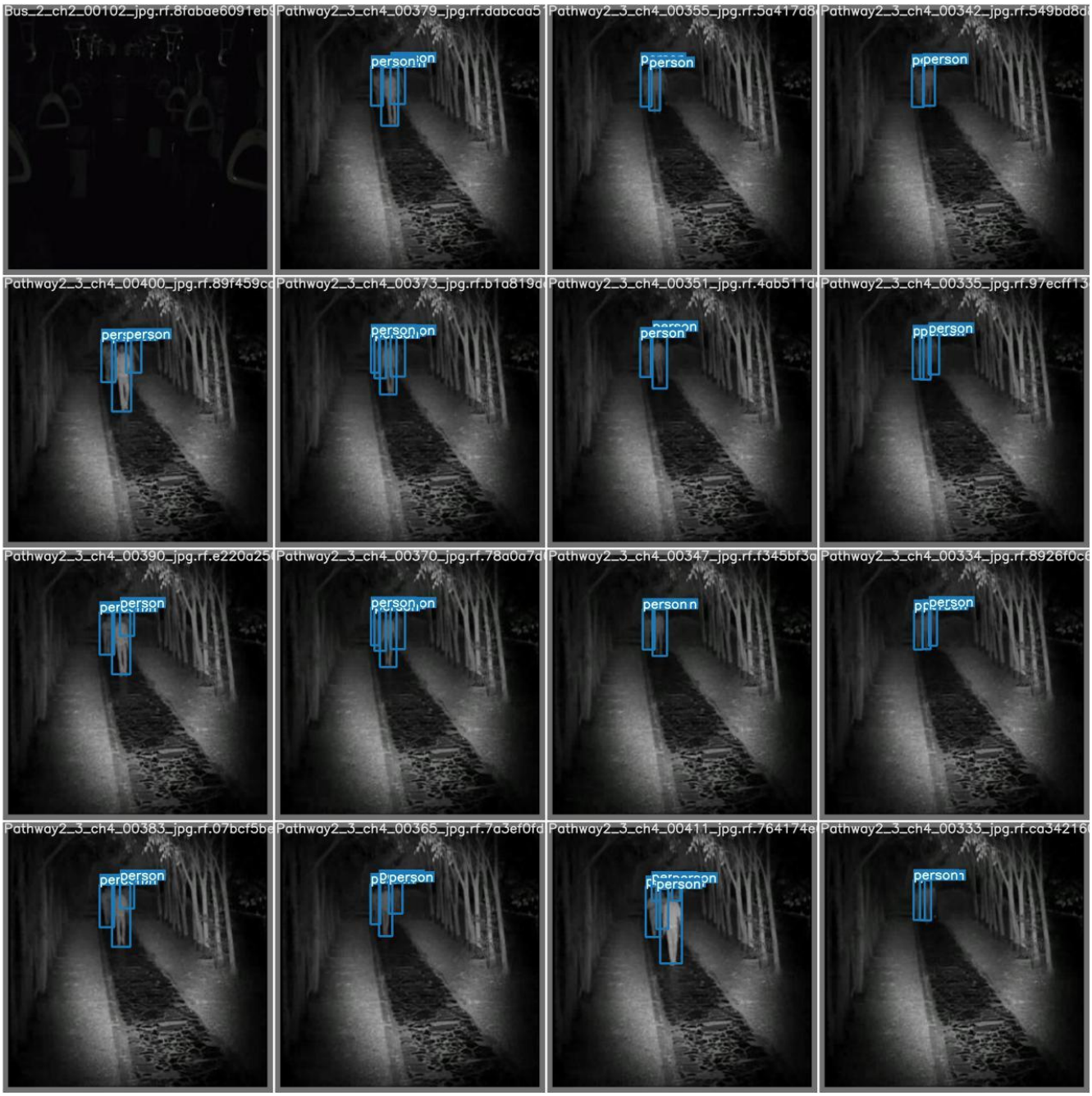


Fig 7.3.1 Test Results for YOLOv7

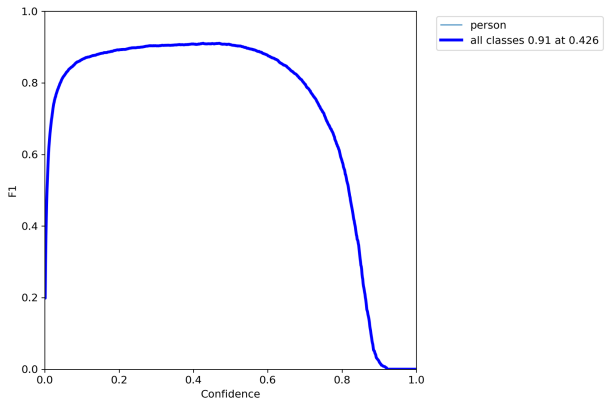


Fig 7.3.2 F1_curve for YOLOv7

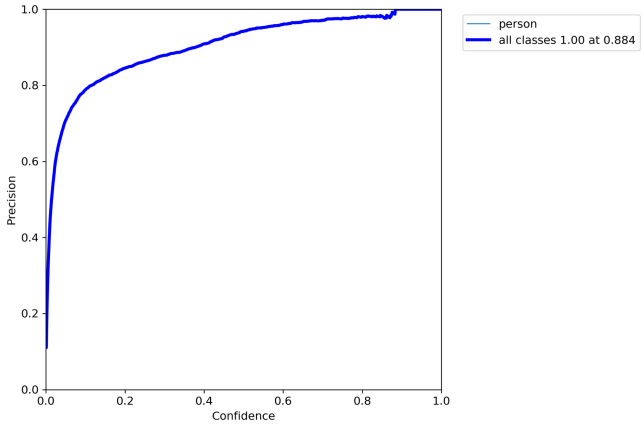


Fig 7.3.3 P_curve for YOLOv7

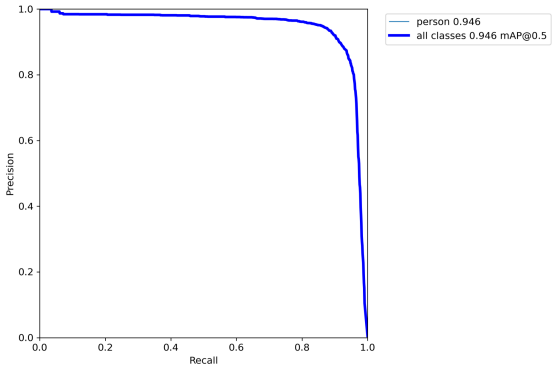


Fig 7.3.4 PR_curve for YOLOv7

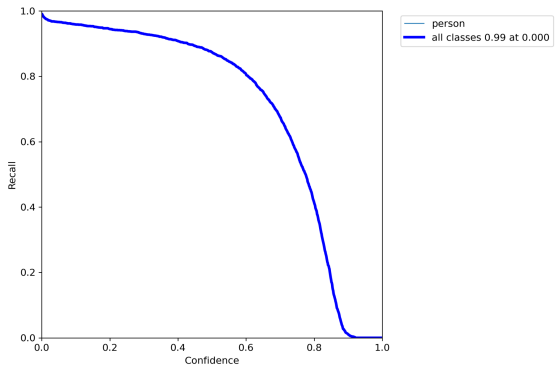


Fig 7.3.5 R_curve for YOLOv7

7.4 TEST RESULTS FOR YOLOv8

Fig 7.4.1 Test Results for YOLOv8

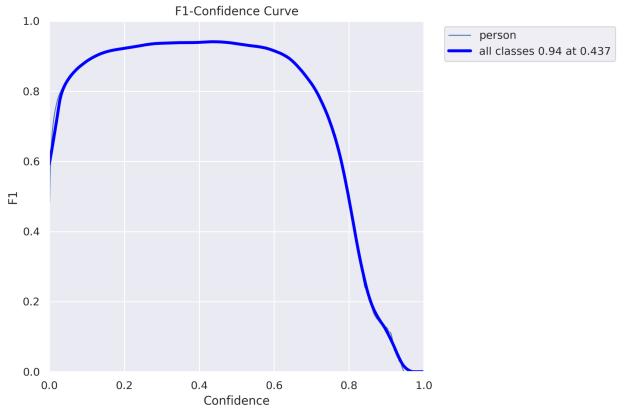


Fig 7.4.2 F1_curve for YOLOv8

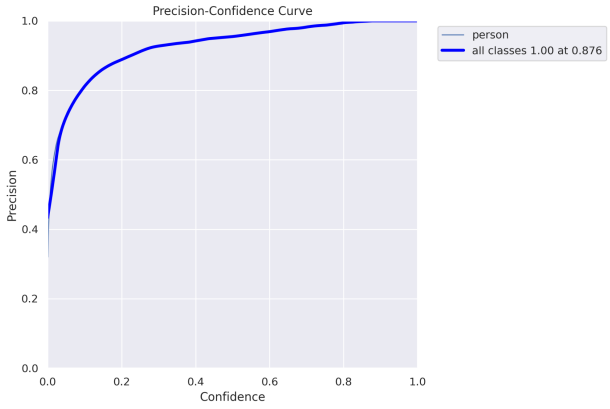


Fig 7.4.3 P_curve for YOLOv8

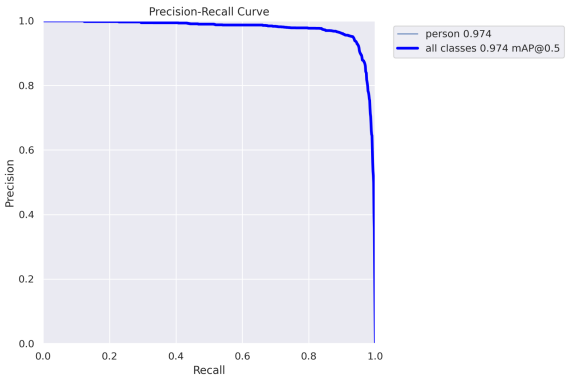


Fig 7.4.4 PR_curve for YOLOv8

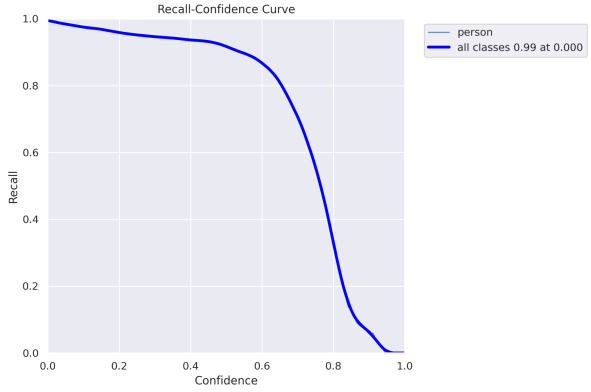


Fig 7.4.5 R_curve for YOLOv8

7.5 RESULTS FOR TRAINING MODELS

7.5.1 Result Summary Of Faster R-CNN

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.539
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.924
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.581
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.434
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.549
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.658
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.407
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.643
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.645
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.576
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.655
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.736
[02/04 18:19:17 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 53.939 | 92.397 | 58.066 | 43.399 | 54.909 | 65.836 |
[02/04 18:19:17 d2.evaluation.coco_evaluation]: Per-category bbox AP:
| category   | AP   | category   | AP   |
|:-----------|:-----|:-----------|:-----|
| Humans     | nan  | person     | 53.939 |
OrderedDict([('bbox',
              {'AP': 53.93878933098454,
               'AP50': 92.39651255025055,
               'AP75': 58.066374311522245,
               'APs': 43.39933373214965,
               'APm': 54.9090118121319,
               'APl': 65.8355938576447,
               'AP-Humans': nan,
               'AP-person': 53.93878933098454})])
```

Fig 7.5.1.1 Result Summary of Faster R-CNN

7.5.2 Result Summary of YOLOv5

```
custom_YOLOv5s summary: 232 layers, 7246518 parameters, 0 gradients, 16.7 GFLOPs
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 52/52 [00:12<00:00,  4.18it/s]
                 all       1663       2098      0.938      0.886      0.947      0.593
```

Fig 7.5.2.1 Result Summary of YOLOv5

7.5.3 Result Summary of YOLOv7

```
Epoch   gpu_mem       box       obj       cls     total    labels  img_size
24/24     10.5G   0.03034  0.003857         0    0.0342        34       640: 100% 365/365 [05:01<00:00,  1.21it/s]
           Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 52/52 [00:32<00:00,  1.61it/s]
             all        1663        2098       0.919       0.902       0.946       0.592
```

Fig 7.5.3.1 Result Summary of YOLOv7

7.5.4 Result Summary of YOLOv8

```
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs
val: Scanning /content/datasets/MI3Human-1/valid/labels.cache... 1663 images, 338 backgrounds, 0 corrupt: 100% 1663/1663 [00:00<?, ?it/s]
                 Class      Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 104/104 [00:27<00:00,  3.82it/s]
                   all        1663        2098      0.942      0.933      0.971      0.691
Speed: 0.6ms pre-process, 9.4ms inference, 0.0ms loss, 1.3ms post-process per image
```

Fig 7.5.4.1 Result Summary of YOLOv8

8. CONCLUSION

Based on the comparison of the proposed approach and the base paper, it is evident that the implemented models, namely Faster RCNN, YOLOv5, YOLOv7, and YOLOv8, have outperformed the previous models. The base paper models achieved an average precision (AP) of 57.69 (SSD), 77.99 (YOLOv4), 68.66 (Faster RCNN), and 73.03 (Mask R-CNN), respectively. In contrast, the proposed models achieved a map50 (mean average precision at 50% intersection over union) score of 92.39 (Faster RCNN), 94.7 (YOLOv5), 94.6 (YOLOv7), and 97.1 (YOLOv8), respectively.The significant improvement in performance can be attributed to the advanced features of the proposed models, such as multi-intensity IR illuminator for better lighting conditions, improved ground-truth annotations, and the use of deep learning frameworks such as PyTorch and Tensorflow. These improvements have enabled the proposed models to detect objects with greater accuracy and speed, making them more efficient and effective for video monitoring applications.

The results of the proposed models indicate a promising future scope for the development of even more accurate and efficient object detection models. Further research can be conducted to explore the incorporation of more advanced features such as 3D object detection, adaptive learning rate, and data augmentation techniques. These advancements can improve the models' performance and make them more robust and reliable for video monitoring applications.

9. REFERENCES

[1] P. J. Lu and J. -H. Chuang, "Fusion of Multi-Intensity Image for Deep Learning-Based Human and Face Detection," in IEEE Access, vol. 10, pp. 8816-8823, 2022. https://doi.org/10.1109/ACCESS.2022.3143536

[2] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, ‘‘YOLOv4: Optimal speed and accuracy of object detection,’’ 2020, arXiv:2004.10934.
https://doi.org/10.48550/arXiv.2004.10934

[3] J. Redmon and A. Farhadi, ‘‘YOLOv3: An incremental improvement,’’2018, arXiv:1804.02767/
https://doi.org/10.48550/arXiv.1804.02767

[4] S. Cho, N. Baek, M. Kim, J. Koo, J. Kim, and K. Park, ‘‘Face detection in nighttime images using visible-light camera sensors with two-step faster region-based convolutional neural network,’’ Sensors, vol. 18, no. 9, p. 2995, Sep. 2018.
https://doi.org/10.3390/s180929

[5] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779-788.
https://doi.org/10.1109/CVPR.2016.91

[6] Chan, C. H., Chen, H. T., Teng, W. C., Liu, C. W., & Chuang, J-H. (2015). MI3: Multi-intensity infrared illumination video database. In *2015 Visual Communications and Image Processing, VCIP 2015* [7457860] (2015 Visual Communications and Image Processing, VCIP 2015). Institute of Electrical and Electronics Engineers Inc..
https://doi.org/10.1109/VCIP.2015.7457860

[7] Wen Chih Teng and Jen-Hui Chuang "A novel synchronous multi-intensity IR illuminator hardware implementation for nighttime surveillance", Proc. SPIE 9407, Video Surveillance and Transportation Imaging Applications 2015, 940711 (4 March 2015);
https://doi.org/10.1117/12.2083477

[8] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015).
https://arxiv.org/pdf/1506.01497.pdf

[9]Kai Guo, Shuai Wu, Yong Xu,"Face recognition using both visible light image and near-infrared image and a deep network".
https://doi.org/10.1016/j.trit.2017.03.001

[10] Yi-Ting Chen, Jen-Hui Chuang, Wen-Chih Teng, Horng-Horng Lin and Hua-Tsung Chen, "Robust licence plate detection in nighttime scenes using multiple intensity IR-illuminator," 2012 IEEE International Symposium on Industrial Electronics, Hangzhou, 2012, pp. 893-898. https://doi.org/10.1109/ISIE.2012.6237207

ANNEXURE A: SAMPLE CODE SNIPPET

```
from ultralytics import YOLO
import cv2,os,time
from postprocessing import *
model = YOLO("../CustomWeights/YOLOv8/best.pt")
class_list = model.model.names
scale_show = 100
video = cv2.VideoCapture(0)
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
filename = 'output_' + time.strftime("%Y%m%d_%H%M%S")+ '.mp4'
fps = int(video.get(cv2.CAP_PROP_FPS))
width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))
while True:
    ret, frame = video.read()
    if ret:
        results = model.predict(frame)
        labeled_img = draw_box(frame, results[0], class_list)
        display_img = resize_image(labeled_img, scale_show)
        cv2.imshow('Frame', display_img)
        out.write(display_img)
        if cv2.waitKey(25) & 0xFF == ord('q'):  break
    else:  break
video.release()
out.release()
cv2.destroyAllWindows()
```

ANNEXURE B: USER MANUAL FOR THE PROJECT HUMAN DETECTION THROUGH INFRARED SURVEILLANCE CAMERAS USING DEEP LEARNING MODELS

Introduction:

This user manual provides instructions on how to use the human detection models developed using Faster RCNN, YOLOv5, YOLOv7, and YOLOv8 on the MI3 dataset for detecting humans in the night time. The models have been trained to detect humans under low-light and multi-intensity illumination conditions, which makes them suitable for applications such as video monitoring.

Hardware and Software Requirements:

The models require a computer with a minimum of 8GB RAM and a GPU with at least 2GB VRAM. The software requirements are as follows:

- Python 3.6 or later
- PyTorch 1.7.0 or later
- OpenCV 4.2.0 or later
- Numpy 1.19.2 or later

Installation

- Install Python 3.6 or later from https://www.python.org/downloads/
- Install PyTorch using the instructions provided on https://pytorch.org/get-started/locally/
- pip install opencv-python
- pip install numpy

Usage:

- Clone or download the project repository from Github.
- Download the MI3 dataset from https://mi3.sjtu.edu.cn/resource.html and place it in the data directory of the project.
- Open the terminal and navigate to the project directory.
- Run the following command to detect humans using Faster RCNN:
  python detect.py --model faster_rcnn --data_dir data --output_dir output
- Run the following command to detect humans using YOLOv5:
  python detect.py --model yolov5 --data_dir data --output_dir output
- Run the following command to detect humans using YOLOv7:
  python detect.py --model yolov7 --data_dir data --output_dir output
- Run the following command to detect humans using YOLOv8:
  python detect.py --model yolov8 --data_dir data --output_dir output

Troubleshooting:

If you encounter any errors during installation or usage, please refer to the Github repository for the project or contact the project owner for support.

Conclusion:

This user manual provides a step-by-step guide on how to use the human detection models developed using Faster RCNN, YOLOv5, YOLOv7, and YOLOv8 on the MI3 dataset for detecting humans in the night time. These models can be used for a variety of applications such as video monitoring, surveillance, and security.