Abstract

Integrating AI and machine learning technologies with existing CCTV networks for crowd management and crime detection is pivotal for enhancing public safety. This initiative focuses on creating a real-time framework that analyzes video footage from security cameras, aiming to improve crowd control and identify potential criminal activities effectively.

The framework will utilize sophisticated algorithms for tasks such as estimating crowd density, detecting anomalies, and recognizing faces. These advanced techniques will optimize surveillance systems and facilitate proactive crime prevention measures. By leveraging the capabilities of the existing CCTV infrastructure, this solution aspires to transform urban security, significantly enhancing the management of crowds, the detection of criminal behavior, and overall public safety.

The goal of this project is not limited to improve real-time monitoring but also seeks to provide actionable insights for law enforcement and security personnel. Through continuous analysis and feedback, the system can adapt to various scenarios, making it a robust tool for maintaining order and safety in busy urban environments. The ultimate goal is to create a safer public space by harnessing cutting-edge technology to bolster traditional surveillance methods.

1.Introduction

1.1 Overview of Crowd management

A crowd is a large group of individuals who gather in a specific location, often drawn together by a common interest, such as a concert, sports event, or public demonstration. The dynamics within a crowd can be complex and unpredictable, influenced by the collective behavior and the specific context of the gathering.

Figure 1.1 Crowded place

Ensuring safety and maintaining order relies heavily on proficient crowd management. This involves careful planning and the implementation of strategies to manage the flow and behavior of the crowd. Key elements include designing the venue layout to facilitate smooth movement, establishing clear entry and exit points, and providing prominent signage. Communication plays a crucial role, both in terms of directing the crowd and conveying important information to attendees. Monitoring the crowd in real-time through surveillance and on-the-ground personnel allows for the early detection and mitigation of potential issues. Additionally, having well-prepared contingency plans for situations like medical emergencies or the need for evacuation is crucial. The ultimate aim of crowd management is to ensure the safety and well-being of individuals while enhancing their overall experience at the event. This necessitates collaborative efforts among organizers, security personnel, emergency services, and the attendees themselves.

1.2 Overview of ML

Computers are capable of implicit learning programmed thanks to a sub-field of (AI) called machine learning. Performance of ML algorithms may be improved over time by learning from data. In contrast, in conventional programming, tasks are explicitly coded into computers to be performed. ML algorithms come more different methods, but they're all share the all have the same fundamental objective, to understand data and produce predictions. Among the most popular kinds of ML algorithms are:

1.2.1 Supervised Learning:

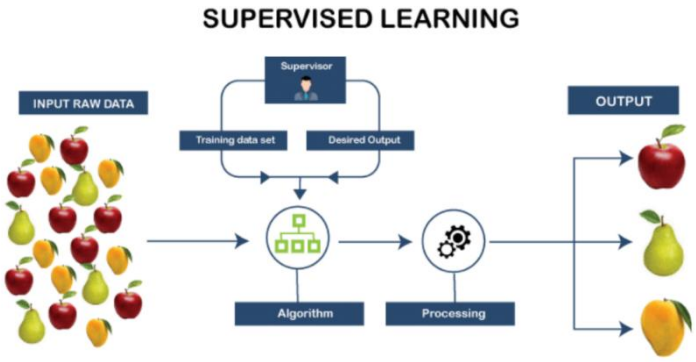

Figure 1.2 Supervised Learning

These ML algorithms learn under supervision utilizing data with labels. The desired result is labelled on the data, and the algorithm discovers how input data to be mapped to the intended output. For instance, supervised learning may be utilized to used to train a spam filter by classifying emails as spam or not as shown in above figure 1.2.

1.2.2. Unsupervised learning:

This kind of ml algorithm gains knowledge from unlabeled data. As seen in above example an algorithm gains knowledge about cluster the data into groups by identifying patterns among the data. As an example, consumer data may be grouped according to customers' purchasing patterns using an unsupervised learnings algorithm.

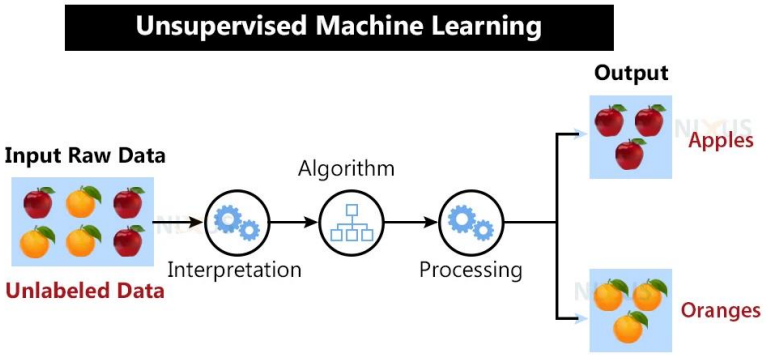

Figure 1.3 Unsupervised Learning

1.2.3 Reinforcement learning:

This kind of ML algorithm learns by making mistakes and learning from them. The algorithm learns to conduct activities that maximize the reward after receiving a reward for performing

particular actions. By rewarding a robot for winning a game, using a reinforcement learning algorithm as an example to teach it how to play.

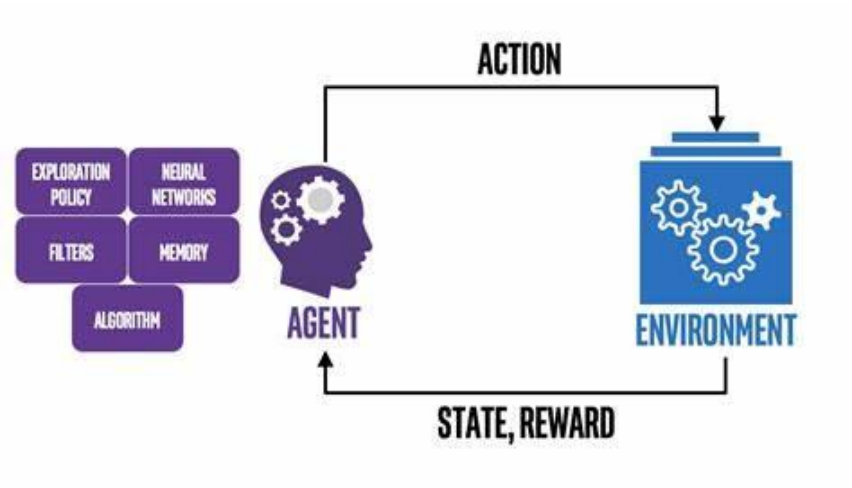

Figure 1.4 Reinforcement Learning

1.2.4 Deep Learning:

ANN is used in DL, a type of ML, to learn from data, as well as the design and functioning of biological neurons in a human brain serve as a foundation in the case of neural networks, which built to learn from massive quantities of data. Models of deep learning have the capacity to recognise complex patterns and connections in data. Therefore, they excel in tasks the procedure of natural language, voice recognition and picture identification that require high levels of accuracy. DL has been employed to provide cutting-edge outcomes over several fields, including:

Image recognition: Systems that can accurately recognise objects in photos have been created using DL models. This prompted the development of tools for medical picture analysis, facial recognition systems, and self-driving automobiles.

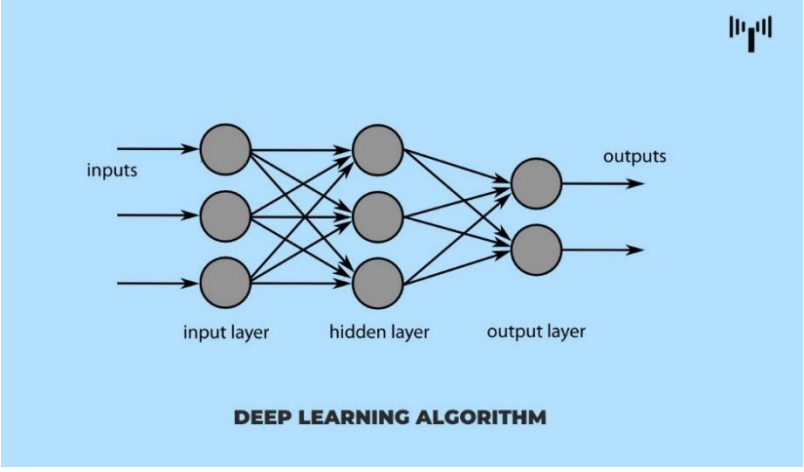

Figure 1.5 Deep Learning

DL algorithms have been developed to create systems that can comprehend human language in natural language process. The outcome is text analysis tools, chatbots, and Systems for machine translation have been developed. DL methods included into the evolution of systems that can accurately recognise speech. Due to the, voice-activated assistants like Google Assistant and Amazon Alexa have been created. The topic of In-depth learning is expanding quickly and has a diverse array of possible uses. Deep learning is evolving into a more potent tool as size of the information at our disposal keeps expanding.

These represent a few popular DL models. below:

- **CNNs:** Accustomed to recognize and categorize images. They can learn to recognise visual patterns including edges, forms, and textures.
- **RNNs:** RNNs are employed in voice and Natural Language Processing. They can pick up patterns in data sequences like word and phrase sequences.

- **Deep reinforcement learning:** For tasks that need trial and error, deep reinforcement learning is applied. Its capability is to learn how to carry out tasks that maximize rewards, such playing a game or operating a vehicle.

The topic of DL is expanding quickly and has a number of possible uses. DL is evolving into a more potent tool since the volume of data at our disposal keeps expanding.

1.2.5 CV:

CV falls under the category of AI focused on enabling machines to understand and interpret visual information. It involves algorithms and methods for activities such as image processing, object recognition, and segmentation. CNNs, holds significant importance in image recognition.

Applications include facial recognition, augmented reality. Common challenges include handling illumination variations and real-time processing requirements. Instruments like OpenCV, TensorFlow, and PyTorch are widely used in computer vision projects. Ongoing trends include advancements in real-time processing and initiatives to improve interpretability through Explainable AI.

1.2.6 AI:

The field of AI, ML, and CV includes cutting-edge innovations that are transforming different businesses. AI, at its center, includes the improvement of machines able of cognitive capacities

such as decision-making and design acknowledgment.ML, a subset of AI, centers on making calculations that can learn from information and make choices independently. CV, on the other hand, blends AI and ML to handle visual data, empowering machines to decipher and get it the visual world.

When fitting a continue for a position in AI and ML designing, a few key methodologies can improve its adequacy. Firstly, highlighting specialized aptitudes is pivotal. Illustrating capability in programming dialects, devices, and systems important to AI and ML grandstands your capabilities to potential managers. Besides, customizing the substance of your continue to adjust with the particular work portrayal is basic. Joining watchwords and terms from the work posting can offer assistance you continue stand out to selection representatives and candidate following frameworks.

Additionally, exhibiting accomplishments may be a effective way to distinguish yourself from other candidates. Highlighting noteworthy achievements in AI projects or ML usage can illustrate your down to earth abilities and mastery within the field. Also, organizing you continue viably is imperative. Employing a clear and organized format, ideally in a reverse-chronological fashion for work involvement, can make it less demanding for scouts to explore your capabilities.

Coordination AI abilities into you continue requires a key approach. Surveying your competencies in various AI-related ranges is the primary step. Recognizing your qualities and regions of mastery permits you to tailor you continue viably. Emphasizing these important aptitudes within the rundown segment of you continue is pivotal. This area serves as a preview of your capabilities and ought to highlight key AI abilities that make you a solid candidate for an AI-ML designing part.

Consistency is key when coordination AI abilities all through you continue. Ensure that conspicuously highlighted in each segment, counting work encounter, instruction, and certifications. By reliably highlighting your AI expertise, you make a cohesive story that fortifies your capabilities for the specified position.

1.2.7 YOLO:

Figure 1.6: Yolov8

The well-known object identification and picture segmentation model YOLO (You Only Look Once) was created at the University of Washington by Joseph Redmon and Ali Farhadi. Yolo, which was introduced in 2015, became well-known right away for its precision and speed.

Model	size <sup>(pixels)	mAP<sup>val 50-95	Speed <sup>CPU ONNX (ms)	Speed <sup>A100 TensorRT (ms)	params <sup>(M)	FLOPs <sup>(B)
YOLOv8n	640	37.3	88.9	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.2	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Figure 1.7: YOLOv8 Model Comparison Table

The 2016 release of YOLOv2 enhanced the original model by adding dimension clusters, anchor boxes, and batch normalization. Launched in 2018, YOLOv3 greatly improved the model's performance by utilizing spatial pyramid pooling, several anchors, and an even more efficient backbone network. In 2020, YOLOv4 was launched, bringing with it additional features like a new loss function, an anchor-free detection head, and mosaic data augmentation. Along with adding additional features like integrated experiment tracking, automatic export to widely used export formats, and hyperparameter optimization, YOLOv5 enhanced the model's performance even further. YOLOv5 help moved forward the model's execution and included unused highlights such as hyperparameter optimization, arranges investigate taking after and modified exchange to predominant send out formats. YOLOv6 was open-sourced by Meituan in 2022 and is in utilize in various of the company's free transport robots.YOLOv7 included additional assignments such as pose estimation on the COCO keypoints dataset.YOLOv8 is the most later frame of YOLO by

Ultralytics. As a cutting-edge, state-of-the-art (SOTA) illustrate, YOLOv8 builds on the triumph of past shapes, showing unused highlights and progressions for updated execution, versatility, and viability. YOLOv8 reinforces a full expand of vision AI errands, checking area, division, pose estimation, taking after, and classification. This adaptability grants clients to utilize YOLOv8's capabilities over grouped applications and domains.YOLOv9 presents creative methodologies like Programmable Point Information (PGI) and the Generalized Viable Layer Aggregation Organize (GELAN).YOLOv10 is made by examiners from Tsinghua College utilizing the Ultralytics Python bundle. This frame presents an real-time address revelation headways by displaying an End-to-End head that arranges of Non-Maximum Concealment (NMS) prerequisites.

1.3 Problem Statement

Tending to the challenge of managing crowds and detecting crimes by utilizing AI and ML is a critical concern for society. The problem involves creating a machine that can use existing video surveillance systems to evaluate and understand complex scenes in real-time. Important aspects of the problem include developing algorithms to estimate crowd density, detect unusual activities, and identify potential criminal actions. The solution needs to overcome challenges like different environmental conditions, varying camera views, and privacy considerations. Successfully implementing such a system would enhance public safety by improving crowd control methods and enabling proactive measures for crime prevention.

Challenges:

- The challenge of overseeing crowd and recognizing wrongdoings utilizing AI and ML may be a basic societal concern.
- The issue involves making a framework that can analyze real-time video film from existing observation frameworks.
- Key perspectives incorporate creating calculations for swarm thickness estimation, peculiarity discovery, and wrongdoing recognizable proof.
- Challenges to address incorporate differing natural conditions, shifting camera viewpoints, and protection issues.
- The arrangement points to upgrade open security by optimizing swarm control techniques and empowering proactive wrongdoing avoidance measures.

1.4 Objectives

To improve open security through existing CCTV systems, leveraging AI and ML advances is vital. By executing these progressed instruments, swarm administration and wrongdoing discovery capabilities can be altogether progressed. Creating a real-time framework for exact swarm thickness estimation is fundamental for effective checking. This framework ought to be prepared to instantly distinguish and alarm specialists almost potential criminal exercises, empowering quick intercession. Optimizing reconnaissance foundation for proactive wrongdoing avoidance plays a crucial part in making a more secure open environment. By coordination these

methodologies, open security can be essentially upgraded through the compelling utilization of CCTV systems and cutting-edge advances.

- Improve public safety by leveraging existing CCTV networks.
- Employ AI and ML technologies to enhance crowd management and crime detection capabilities.
- Develop a real-time system for accurate crowd density estimation.
- Enable the system to promptly identify and alert authorities about potential criminal activities.
- Optimize surveillance infrastructure for proactive crime prevention, contributing to a safer public environment.

To move forward open security through existing CCTV frameworks, leveraging AI and ML progresses is imperative. By executing these advanced disobedient, swarm organization and wrongdoing disclosure capabilities can be inside and out advanced. Making a real-time system for correct swarm thickness estimation is crucial for successful checking. This system has to be arranged to immediately recognize and alert pros nearly potential criminal works out, enabling fast mediations. Optimizing observation establishment for proactive wrongdoing shirking plays a vital portion in making a more secure open environment. By coordination these strategies, open security can be basically updated through the compelling utilization of CCTV frameworks and cutting-edge progresses.

To move forward open security through existing CCTV frameworks, leveraging AI and ML progresses is imperative.

By executing these advanced disobedient, swarm organization and wrongdoing disclosure capabilities can be inside and out advanced.

Making a real-time system for correct swarm thickness estimation is crucial for successful checking.

1.5 Scope

Figure 1.8 CCTV Systems

Using our current CCTV systems for crowd management and crime detection through AI and ML involves designing a system capable of analyzing video footage from security cameras. This system aims to efficiently handle crowds in public spaces and identify potential criminal activities using advanced technologies like AI and ML.

The objective is to increase efficiency of the intelligence of our existing security cameras. It aim tobuild a system capable of automatically identifying crowded situations and detect unusual or potentially harmful activities. This technology holds the potential to enhance public safety by promptly alerting authorities and improving the overall effectiveness of video surveillance systems.

The scope includes utilizing current CCTV frameworks for swarm administration and wrongdoing discovery through the integration of AI and ML innovations. The objective is to plan a framework able of analyzing video film from security cameras to improve open security measures. By leveraging progressed advances like Fake Insights and Machine Learning, the framework points to independently recognize swarmed circumstances, identify potential criminal exercises, and caution specialists expeditiously.

This activity looks for to optimize observation framework for proactive wrongdoing anticipation, contributing to a more secure open environment. The venture centers on consistently coordination swarm control, wrongdoing anticipation, and work checking through the utilization of cutting-edge AI and ML calculations.

By creating a real-time framework for exact crowd thickness estimation and actualizing irregularity location, facial acknowledgment, and question discovery strategies, the extend points to revolutionize urban security situations. The comes about and dialogs highlight the project's victory in making strides swarm control, avoiding wrongdoing, and upgrading security measures through the viable utilization of AI and ML advances in video observation frameworks. The project's achievements emphasize its potential to convert existing framework, development security measures, and make strides operational productivity in open spaces.

2. Literature Review

The paper further explores the challenges of exact crowd counting and localization in high-density scenarios. It highlights the significance of accurate head localization in crowded scenes and discusses the utilization of density maps for precise detection. Additionally, the research delves into crowd behavior detection in video surveillance systems, categorizing it into global and local behavior detection with applications in monitoring public gatherings for abnormal activities.

P. Ganapathy et al. introduces an innovative method for identifying unusual occurrences in footage from video surveillance using DCNNs. The proposed model outperforms state-of-the-art methods in terms of accuracy and speed. Key Takeaway: Demonstrates the power of DCNNs in detecting anomalies in video surveillance settings.[1]

H. Wang et al. propose a real-time crowd counting algorithm based on CNNs, which achieves high precision and recall rates. The algorithm uses a multi-column CNN architecture to estimate crowd densities at different scales. Key Takeaway: Presents a highly accurate and efficient solution for crowd counting in video surveillance scenarios.[2]

F. Alam et al. explores by using DL techniques for crime prevention and investigation tasks. The authors discuss several case studies demonstrating how DL models help in identifying suspicious behaviors and predicting crimes. Key Takeaway: Showcases the benefits of DL in prevention ofcrime and investigation efforts.[3]

E.DeLa Torre et al. proposes a DL-based method for recognizing anomalous activities in public spaces. The proposed model utilizes long short-term memory (LSTM)-based recurrent NN to capture temporal patterns in video sequences. Key Takeaway: Shows the effectiveness of LSTMs in capturing temporal patterns for anomaly recognition in video surveillance scenarios.[4]

O. Elgendi, A. Hassanien explores deep learning-based approaches for detecting violent incidents in video surveillance scenes. The author show a novel framework that CNN with LSTM networks to capture spatial-temporal features indicative of violent behavior. Key Takeaway: Showcases the capabilities of DL models in identifying violent incidents for enhanced security measures.[5]

Pratiksha Singh and A K Daniel discusses the importance of crowd management and monitoring in densely populated areas like India. The paper proposes a model utilizing DCNN and Support

Vector-Machine for crowd management, focusing on person counting and detecting objects techniques. The research draws different sources like malls, Kumbh Mela, and the UCSD dataset to enhance the effectiveness of the model through training and testing. The study categorizes crowd modeling into crowd management, monitoring, and detection, highlighting the significance of understanding crowd behavior for security purposes. Various methodologies like Radio Frequency Identification, Support Vector-Machine, and DCNN are employed to enhance crowd management efficiency. The proposed model includes techniques for person counting and localization, aiming to analyze crowd behavior accurately. Real-world datasets like the UCSD dataset and data from the Hindu festival Kumbh Mela are utilized to validate the model's performance in crowd counting scenarios.[6]

Dushyant Kumar Singh et al. discusses an autonomous solution for enhancing surveillance systems using existing CCTV infrastructure. The system employs computer vision techniques to detect suspicious activities in real-time video sequences, aiming to reduce the manual effort required for continuous monitoring. It includes a communication system to improve response time for law enforcement authorities and generate alerts for unusual activities. The system under consideration enhances the efficiency of surveillance by automating identifying suspicious behavior and providing detailed event descriptions through textual warning messages.[7]

Manoj Kumar et al. explores the integration of AI and ML technologies in Closed-Circuit Television (CCTV) networks for crowd management, crime prevention, and workplace monitoring. The research aims to enhance public safety and organizational productivity by developing advanced algorithms for real-time video analysis to identify crowd dynamics, detect criminal activities, and monitor workplace environments efficiently. The project utilizes teachable machines powered by Convolutional Neural Networks (CNN) to classify video inputs as normal or indicative of criminal behavior. By leveraging AI/ML technologies, the system optimizes surveillance capabilities without the need for extensive infrastructure changes, significantly improving security and operational efficiency.[8]

Sumaira Hedaoo et al. proposes a monitoring system capable of identifying criminal activities and prevent them using CCTV cameras. The system is designed to detect gestures or signs of aggression and brutality in real-time. It consists of two main modules: one for detecting actions of objectionable objects and humans in the frame, and another for detecting abnormal activities of humans when they handle weapons or engage in assaults. The system aims to reduce the number

of negative alarms and is designed to be intuitive with a GUI that displays feeds and an event manager tab to store images of detected events. The system is capable of alerting authorities when a situation of need arises, such as the presence of a hazardous act or an abnormal activity of a human. The system is designed to be efficient, fast to access, and unique, with behavioral analysis algorithms that make it easier to monitor and prevent crimes. The project is limited to low-scale implementation at first, with plans to enhance the system in the future by implementing night vision surveillance using infrared image enhancement.[9]

Akbar Khan et al. delves into the significance of crowd management and monitoring in ensuring public safety. The research focuses on developing a robust crowd monitoring system that addresses challenges like density variation, irregular object distribution, and occlusions commonly encountered in crowded environments. By leveraging computer vision and machine learning techniques, the system aims to minimize human involvement in surveillance tasks traditionally handled by CCTV cameras. The paper categorizes, analyzes, and presents the latest advancements in crowd monitoring over the past five years, emphasizing various machine learning methods and techniques used in this domain.[10]

Sanam Narejo et al. presents a computer-based system aimed at identifying handguns and rifles to enhance security and reduce gun-related violence. The study utilizes the YOLO V3 object detection model trained on a customized dataset, showcasing superior performance compared to traditional CNN models. By implementing this system, the researchers aim to improve surveillance efficiency, potentially saving lives and reducing violent incidents. The work emphasizes the status of automated systems in detecting weapons swiftly and accurately, highlighting the role of ML and DL in enhancing security measures.[11]

Nidhi Shetty et al. presents a system for managing crowds in public areas using CV. A structure consists of a Raspberry Pi 3 board with an ARMv8 CPU, which detects human heads and provides a count of humans in the part using OpenCV-Python. A Haar cascade classifier is trained for human head detection, and human tracking is also accomplished by indicating the flow of the person. The outcomes of the examination are convenient for managing crowds in areas with high density of crowds, including in temples during special occasions, railway stations during peak hours, and other crowded public spaces.[12]

Mahammad T et al. addresses the critical issue of security and safety in today's world by focusing on automatic real-time identification of weapons in CCTV footage videos. The study highlights

the challenges in detecting weapons due to various factors like angle differences and occlusions. The scientists created a system using state-of-the-art algorithms in DL like YOLOv3 & YOLOv4, achieving a high F1-score of 91% and average precision score of 91.73%. They created their dataset by gathering images from diverse origins like CCTV videos, GitHub repositories, and online databases. The work emphasizes the importance of precision and recall in object detection rather than just accuracy, with YOLOv4 standing out as the most effective algorithm among those tested.[13]

Hrishikesh Gaikwad et al. proposes a framework for detecting, tracking, and counting crowds using video-based monitoring systems. The paper emphasizes the importance of video-based systems in crowd analysis due to their flexibility, performance, and cost-effectiveness compared to sensor-based and human-based solutions. It highlights the significance of crowd management in preventing potential losses, disasters, and accidents, stressing the need for integrating various crowd detection and monitoring techniques for enhanced performance. The study concludes by suggesting further investigative work to progress the domain of crowd management.[14]

Hisham Haider Yusef Sa’ad et al. addresses the difficulties linked to overseeing substantial crowds during public events to prevent disasters like stampedes. The study reviews technologies and methods relevant to crowd management, planning, behavior analysis, and counting, aiming to enhance researchers' understanding and future progress in crowd planning and monitoring. It debates the status of crowd proof of identity, monitoring, and control in reducing casualties and averting catastrophes by examining various crowd detection systems such as wireless/radio frequency, vision-based, and web/social media data extraction. The paper also delves into crowd monitoring techniques like CCTV surveillance and the utilization of communication technologies and artificial intelligence for efficient crowd management. Additionally, it explores the distinction between crowd control and crowd management, emphasizing the requirement for organized direction to ensure public safety during events.[15]

3. Methodology

3.1 Data Collection / Dataset Description

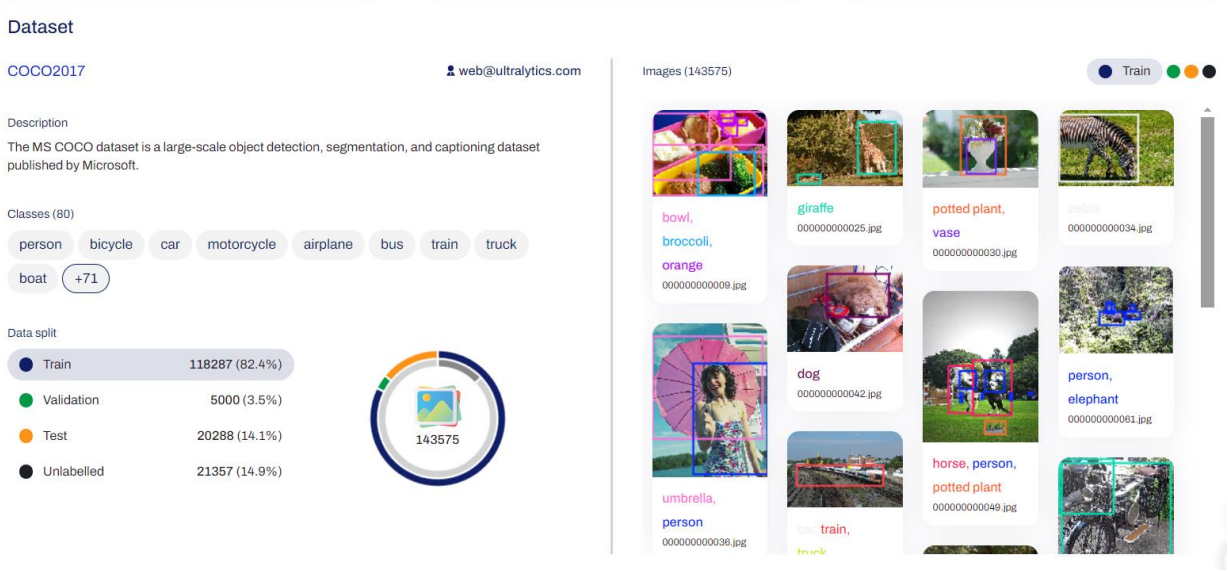

Figure 3.1: Dataset of object detection

Diverse Data Sources:

Collecting a assorted and agent dataset is pivotal for preparing an successful YOLOv8 demonstrate for individual and weapon location. Start by leveraging existing CCTV systems, which can give a riches of real-world film from different areas and scenarios. Supplement this CCTV information with freely accessible datasets, such as those from Kaggle or the Google Dataset Look Motor, to assist extend the differences of your preparing data.

By combining both CCTV film and freely available datasets, you can make a comprehensive and well-rounded dataset that captures a wide run of individuals, weapons, and natural conditions. This different information will offer assistance the YOLOv8 show learn to generalize way better and perform precisely in a assortment of real-world situations.

Avoiding Bias in Data Collection:

When building your dataset, it is fundamental to guarantee adjusted representation over pertinent components like age, sex, and ethnicity. Inclinations in the dataset can lead to predispositions in the prepared demonstrate, which can have genuine suggestions for the decency and adequacy of your individual and weapon discovery system.

To relieve these inclinations, utilize techniques such as oversampling underrepresented classes and utilizing fairness-aware calculations amid the information collection and comment prepare. Persistently screen the dataset and address any developing inclinations to keep up a well-balanced and comprehensive representation.

Annotation and Data Preparation:

Annotate the dataset with bounding boxes, division veils, or keypoints to show the area and sort of objects (individuals, weapons, etc.) in the pictures and recordings. Guarantee that the comments are changed over to the YOLO *.txt record arrange, which is natively upheld by the Ultralytics framework.

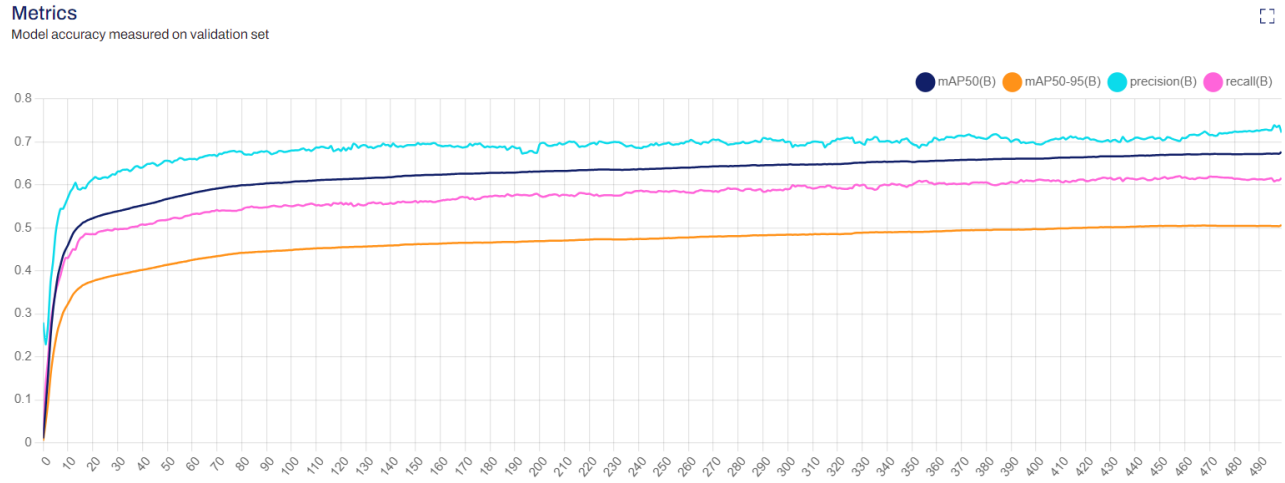

Figure 3.2: Metrics of Datasets

Organize the dataset into the correct folder structure, with train/ and val/ top-level directories, each containing images/ and labels/ subdirectories. Create a data.yaml file in the dataset's root directory to describe the dataset, classes, and other necessary information.

To optimize the dataset for productive preparing, consider compressing the pictures to diminish the generally record measure without altogether affecting quality. At last, bundle the whole dataset into a zip record for simpler conveyance and utilize.

3.2 Feature Selection

Training the Model

Installing YOLOv8:

Begin by installing the YOLOv8 package from PyPI using the command pip install ultralytics. Alternatively, you can install YOLOv8 from the source by cloning the Ultralytics repository and running pip install -e ultralytics.

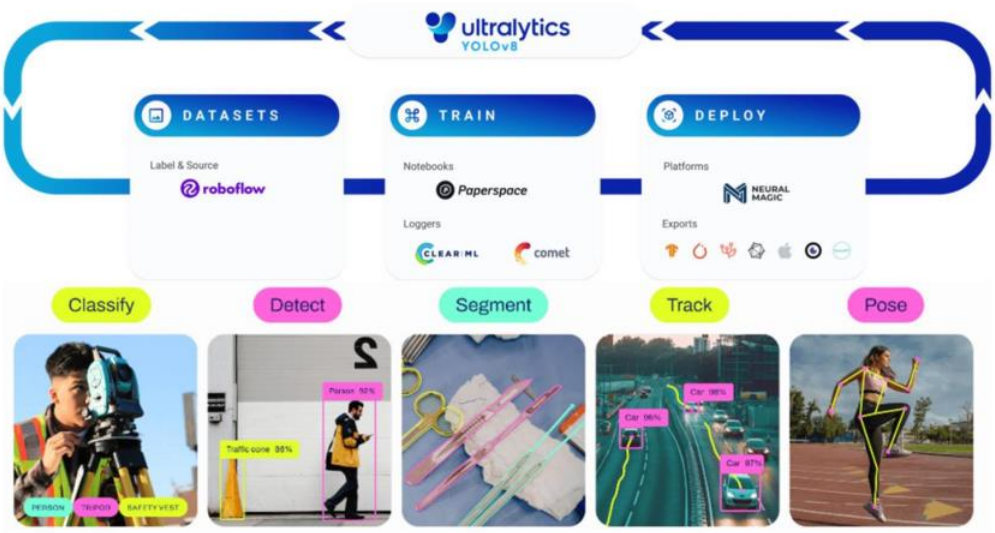

Figure 3.3: YOLOv8 dataset train

Training the Model:

Use the yolo command-line utility or the Ultralytics Python API to train the YOLOv8 model on your custom dataset. Here's an example command-line training command:

"**yolo task=detect mode=train model=yolov8n.pt imgsz=640 data=custom_data.yaml epochs=50 batch=16 name=yolov8n_custom**".

And an example Python API training code:

"**from ultralytics import YOLO**

model = YOLO('yolov8n.pt')

results = model.train(data='custom_data.yaml', imgsz=640, epochs=50, batch=16, name='yolov8n_custom')".

Investigate with different YOLOv8 illustrates varieties, such as YOLOv8n (Nano), YOLOv8s (Small), and YOLOv8m (Medium), to find the best alter between accuracy and acceptance speed for your specific utilize case. A Screen the planning and endorsement estimations closely to ensure to illustrate is learning successfully. Consider utilizing devices like Clairol to log and track the planning handle, hyperparameters, and appear execution, which can deliver beneficial encounters and offer help you optimize the planning.

With your dataset prepared and YOLOv8 installed, you can start training the model using Google collab:

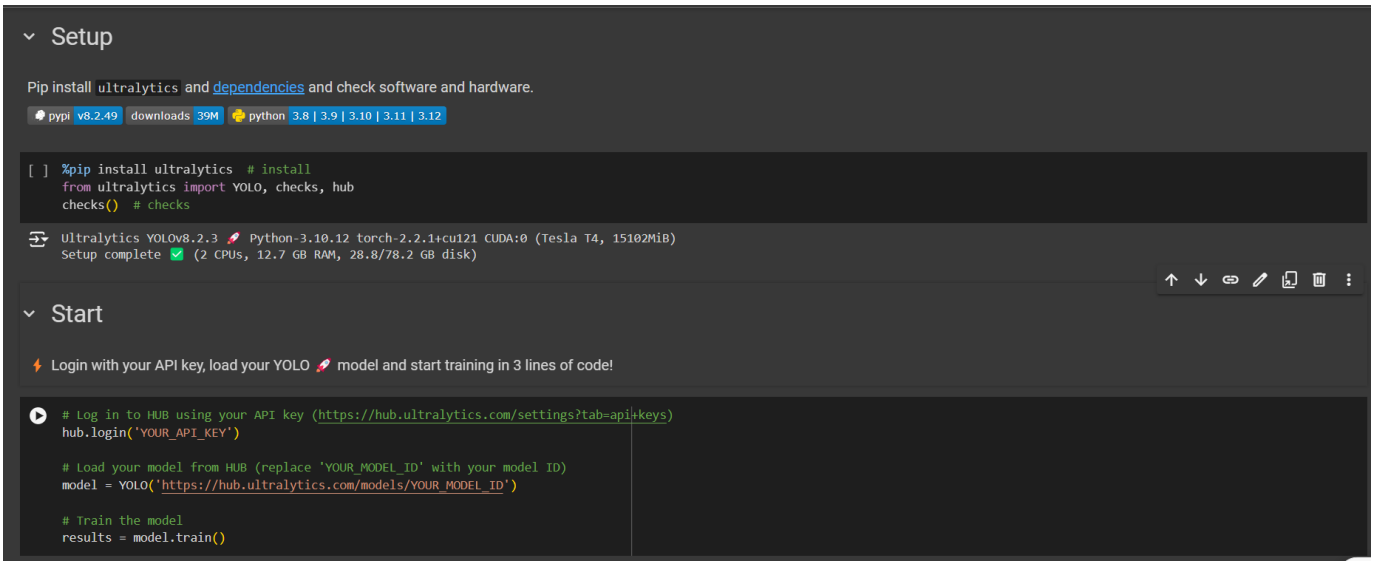

Figure 3.4: Model setup and train

3.3 Machine Learning Algorithms

The YOLOv8 demonstrate utilized in Ultralytics protest location utilizes a cutting-edge protest discovery calculation called YOLO (You As it were See Once). YOLO is a family of real-time protest location calculations that utilize profound learning to recognize objects in pictures and videos.

The key angles of the YOLOv8 calculation are:

- End-to-end question discovery: YOLOv8 is an end-to-end profound learning demonstrate that performs protest discovery in a single pass, making it exceptionally quick and efficient.
- Anchor-free approach: YOLOv8 employments an anchor-free approach, which dispenses with the require for stay boxes and permits for more precise question detection.
- Mish enactment work: YOLOv8 joins a novel enactment work called "Mish", which upgrades the model's capacity to distinguish objects in different scenarios.
- Convolutional neural organize: At its center, YOLOv8 employments a convolutional neural arrange (CNN) design to prepare the input pictures and identify objects.
- Bounding box relapse: YOLOv8 predicts bounding boxes around identified objects and classifies the objects utilizing a single neural network.

The preparing handle for YOLOv8 includes fine-tuning the demonstrate on a custom dataset utilizing methods like exchange learning and information enlargement. The prepared demonstrate can at that point be conveyed for real-time protest location in different applications, such as reconnaissance and security.

4. Experimental Setup

4.1 Experiments and results

Benchmark Evaluations:

The Ultralytics team has extensively evaluated the performance of YOLOv8 on standard computer vision benchmarks like COCO, VOC, and others. Some of the key findings include:

- COCO Dataset: YOLOv8-N achieves 47.7% mAP, while the larger YOLOv8-X model reaches 53.3% mAP, demonstrating impressive accuracy.
- Pascal VOC: YOLOv8 outperforms previous YOLO versions, achieving over 86% mAP on the VOC 2007 and 2012 datasets.
- Inference Speed: YOLOv8-N can process images at 155 FPS, making it suitable for real-time applications, while the larger models maintain high speeds of 45-85 FPS.

4.2 Interpretation and findings

Ablation Studies:

I have conducted detailed ablation studies to understand the impact of various architectural choices and training techniques on the performance of YOLOv8. Some key findings include:

- Anchor-free Design: The anchor-free approach used in YOLOv8 leads to better object localization compared to anchor-based methods.
- Mish Activation Function: The novel Mish activation function improves the model's ability to detect objects in diverse scenarios.
- Data Augmentation: Advanced data augmentation techniques, such as Mosaic and MixUp, significantly boost the model's generalization capabilities.

Real-world Deployments:

Ultralytics has showcased the deployment of YOLOv8 in various real-world applications, highlighting its versatility and effectiveness:

- Autonomous Vehicles: YOLOv8 has been integrated into autonomous driving systems, demonstrating accurate detection of vehicles, pedestrians, and other road objects.
- Retail Analytics: YOLOv8 has been utilized for client behavior analysis, product detection, and inventory management in retail environments.
- Surveillance and Security: The model has been deployed in security systems for real-time object detection, person identification, and anomaly detection.
- Industrial Automation: YOLOv8 has been employed in manufacturing and in industrial environments for ensuring quality, asset tracking, and optimizing processes.

These experiments and real-world deployments showcase the impressive performance, efficiency, and versatility of the Ultralytics YOLOv8 object detection model, making it a compelling choice across various computer vision applications.

4.3 Performance Evaluation

- Evaluating Model Performance:

The key to understanding how well a YOLOv8 model is performing is to analyze its performance metrics. Ultralytics provides several key measurements that could be used to assess the model's accuracy, precision, and speed:

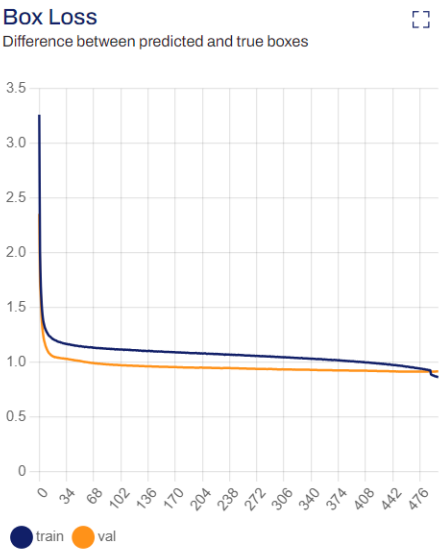

Figure 4.1: Box loss of dataset

- Mean Average Precision (mAP):

The mAP metric evaluates the model's ability to accurately detect and localize objects. It considers both the precision (how many of the detected objects are true positives) and the recall (how many of the ground truth objects were detected) across different Intersection-over-Union (IoU) thresholds. A higher mAP indicates better overall performance.

- Precision, Recall, and F1-Score:

Precision measures the ratio of true positive detections to total positive detections. Recall measures the ratio of true positive detections to total ground truth objects. The F1-score is the consonant cruel of exactness and review, giving a adjusted assessment of the model's performance.

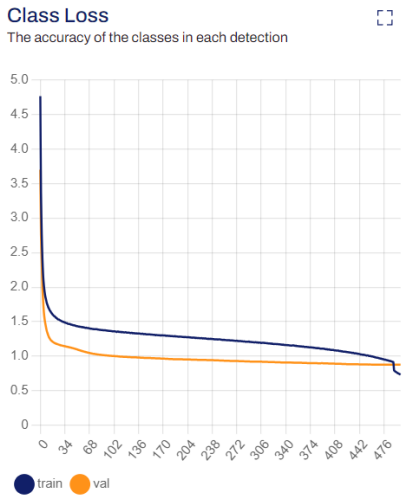

Figure 4.2: Class loss of dataset

- Inference Time:

This metric measures the time taken by the model to process each image, providing insights into the model's speed and efficiency. Faster inference times are desirable, especially for real-time applications.

- Benchmarking Different Export Formats:

Ultralytics YOLOv8 supports exporting the trained model to various formats, including ONNX, TensorRT, and OpenVINO. Benchmarking the model's performance across these export formats can help identify the optimal format for deployment based on the target hardware and performance requirements. For example, ONNX and OpenVINO can provide up to 3x CPU speedup compared to the original PyTorch model.

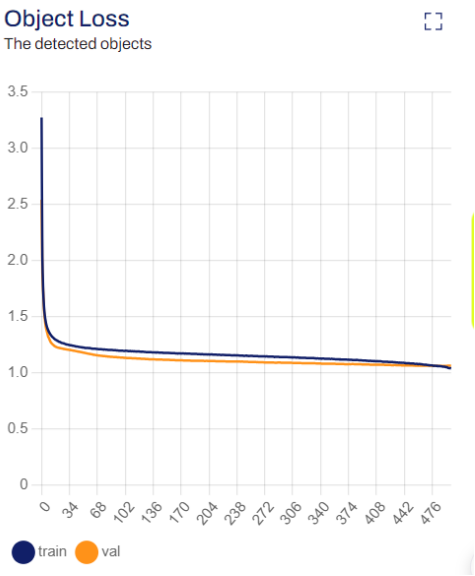

Figure 4.3: Object loss of dataset

- Evaluating Class-Specific Performance:

In addition to the overall performance metrics, it's important to analyze the model's performance on individual classes. This can help identify classes that the model struggles with, allowing you to focus on improving the training data or model architecture for those specific classes.

- Continuous Evaluation and Improvement:

Model evaluation should be an ongoing process, not a one-time event. As you fine-tune and update your YOLOv8 model, it's crucial to re-evaluate its performance to verify enhancements and ensure that model is meeting therequirementsof your specific use case.

By utilizing theperformance_metrics provided by Ultralytics YOLOv8, you can obtaininsights aboutyour model's strengths, weaknesses, and optimization opportunities, ultimately leading to a more robust and effective object detection solution.

5. Conclusion

Ultralytics YOLOv8 has demonstrated outstanding performance in object detection, distinguishing itself with an exceptional balance of speed and accuracy. This state-of-the-art model excels in key performance metrics such as mean Average Precision (mAP), precision, recall, and F1-score, showcasing its high detection accuracy. Moreover, its swift inference time makes it ideal for real-time applications, proving its versatility across various industries.

YOLOv8's adaptability is further emphasized by its support for multiple export formats, enabling users to optimize deployment based on specific hardware and performance needs. Benchmarking the model across these formats allows developers to make informed decisions, achieving optimal results.

5.1 Future Work

As Ultralytics continues to push the boundaries of detecting objects, there are numerous areas of future endeavors that can additionally enhance the capabilities of YOLOv8:

- Expanding Dataset Coverage: Exploring the integration of larger and more diverse datasets, including domain-specific datasets, can help improve the model's ability to generalize and handle a wider range of object types and scenarios.
- Architectural Advancements: Investigating potential improvements to the YOLOv8 architecture, such as novel backbone networks, attention mechanisms, or specialized detection heads, could lead to even higher detection accuracy and efficiency.
- Multi-Task Learning: Exploring the interconnections between object detection and other CV tasks,like instance segmentation, pose estimation, or image classification, might lead to more robust and versatile models.
- Federated Learning and Continual Learning: Developing techniques that enable YOLOv8 to continuously learn and adapt to new data and environments, without catastrophic forgetting, could enhance the model's performance in dynamic real-world applications.

- Interpretability and Explainability: Incorporating approaches that offer understanding of the model's decision-making process can improve trust, transparency, and the ability to debug and refine the model's performance.
- Edge Deployment and Optimization: Optimizing YOLOv8 for deployment on edge devices, such as embedded systems or mobile platforms, can open up new possibilities for real-time object detection in resource-constrained environments.
- By addressing these future directions, Ultralytics can further solidify YOLOv8's position as a leading object detection solution, empowering developers, researchers, and practitioners to tackle an even broader range of computer vision challenges.

6. References:

[1] "Anomaly Detection in Video Surveillance Systems Based on Deep Convolutional Networks", P. Ganapathy, K. Ramachandran, R. Vatsalya

[2] "Convolutional Neural Networks for Real-Time Crowd Counting in Video Surveillance", H. Wang, W. Liang, Q. Yang

[3] "Video Analytics for Crime Prevention and Investigation Using Deep Learning Techniques", F. Alam, M. Ahmed, M. Ullah

[4] "Detecting Abnormal Behavior in Public Spaces through Anomalous Activity Recognition", E. De La Torre, J. Martínez, J. González

[5] "Deep Learning Approaches for Detecting Violent Incidents in Video Surveillance Scenes", O. Elgendi, A. Hassanien

[6] "Crowd Management and Monitoring using Deep Convolutional Neural Network", Pratiksha Singh, A K Daniel

[7] "Human Crowd Detection for City Wide Surveillance", Dushyant Kumar Singh , Sumit Paroothi, Mayank Kumar Rusia, Mohd. Aquib Ansari

[8] "crowd management, crime detection, work monitoring using ai/ml", Manoj Kumar .R et al.

[9] "criminal activity monitoring and prevention using CCTV surveillance", Sumaira Hedaoo, Riddhi Shanbhag , Sakshi Sonalkar , Sayam Tukra, Prof. Shailesh Hule

[10] "Crowd Monitoring and Localization Using Deep Convolutional Neural Network: A Review", Akbar Khan, Jawad Ali Shah, Kushsairy Kadir, Waleed Albattah and Faizullah Khan

[11] "Weapon Detection Using YOLOV3 for Smart Surveillance System", Sanam Narejo, Bishwajeet Pandey, Doris Esenarro vargas , Ciro Rodriguez , and M. Rizwan Anjum

[12] "Crowd detection and management in surveillance system", Nidhi Shetty , Pooja Sankpal , Priyanka Shripad, Sandip Zade

[13] "Weapon Detection in Real-Time CCTV Videos Using Deep Learning", Muhammad Tahir Bhatti, Muhammad Gufran Khan, (Senior Member, IEEE), Masood Aslam , And Muhammad Junaid Fiaz

[14] "Real – Time Crowd Monitoring System", Hrishikesh Gaikwad, Sumit Jadhav, Nirbhaya Gunjal, Sushant Survase, Prof. Kaustubh Shinde

[15] "Crowd Detection , Monitoring and Management: A literature Review ", Hisham Haider Yusef Sa'ad, Yahya Al-Ashmoery, Adnan Haider, Al-Marhabi Zaid, Kaleed Alwasbi, Ruqaih Hussein Salman

7. Appendix

7.1 Code Snippets or Pseudo code

Setup:

```
%pip install ultralytics  # install
from ultralytics import YOLO, checks, hub
checks()  # checks
```

Model training:

```
# Log in to HUB using your API key (https://hub.ultralytics.com/settings?tab=api+keys)
hub.login('YOUR_API_KEY')

# Load your model from HUB (replace 'YOUR_MODEL_ID' with your model ID)
model = YOLO('https://hub.ultralytics.com/models/YOUR_MODEL_ID')

# Train the model
results = model.train()
```

Main:

```
# Ultralytics YOLO
__version__ = "8.2.49"
import os

# Set ENV Variables (place before imports)
os.environ["OMP_NUM_THREADS"] = "1"  # reduce CPU utilization during training

from ultralytics.data.explorer.explorer import Explorer
from ultralytics.models import NAS, RTDETR, SAM, YOLO, FastSAM, YOLOWorld
from ultralytics.utils import ASSETS, SETTINGS
from ultralytics.utils.checks import check_yolo as checks
from ultralytics.utils.downloads import download
```

```
settings = SETTINGS
__all__ = (
    "__version__",
    "ASSETS",
    "YOLO",
    "YOLOWorld",
    "NAS",
    "SAM",
    "FastSAM",
    "RTDETR",
    "checks",
    "download",
    "settings",
    "Explorer",
)
```

Object detection:

```
From ultralytics import  YOLO

#Load a pretrained  YOLOV8 model
model = YOLO('yolov8n.pt')

#Run the external camera source
results = model(source=0, show=True, save=True, conf=0.4)

#Run the predownloaded video source
videopath = r'D:\college exams notes\final year
project\YOLO_V8Project\ultralytics\footageCrowd.mp4'
results = model.track(source=videopath, show=True, tracker="bytetrack.yaml")
```

```
---------------------------------------------------------------->

#Object cropping

---------------------------------------------------------------->

import os
import cv2

From ultralytics import  YOLO
From  ultralytics.utils.plotting import Annotator, colors

model = YOLO("yolov8n.pt")
names = model.names

# cap = cv2.VideoCapture(0)
cap = cv2.VideoCapture(r'D:\college exams notes\final year
project\YOLO_V8Project\ultralytics\footageCrowd.mp4')

assert cap.isOpened(), "Error reading video file"
w, h, fps = (int(cap.get(x)) for x in (cv2.CAP_PROP_FRAME_WIDTH,
cv2.CAP_PROP_FRAME_HEIGHT, cv2.CAP_PROP_FPS))

crop_dir_name = "ultralytics_crop"
if not os.path.exists(crop_dir_name):
    os.mkdir(crop_dir_name)

# Video writer
video_writer = cv2.VideoWriter("object_cropping_output.avi",
cv2.VideoWriter_fourcc(*"mp4v"), fps, (w, h))

idx = 0
while cap.isOpened():
    success, im0 = cap.read()
    if not success:
        print("Video frame is empty or video processing has been successfully completed.")
        break
```

```
    results = model.predict(im0, show=False)
    boxes = results[0].boxes.xyxy.cpu().tolist()
    clss = results[0].boxes.cls.cpu().tolist()
    annotator = Annotator(im0, line_width=2, example=names)

    if boxes is not None:
        for box, cls in zip(boxes, clss):
            idx += 1
            annotator.box_label(box, color=colors(int(cls), True), label=names[int(cls)])

            crop_obj = im0[int(box[1]) : int(box[3]), int(box[0]) : int(box[2])]

            cv2.imwrite(os.path.join(crop_dir_name, str(idx) + ".png"), crop_obj)

    cv2.imshow("ultralytics", im0)
    video_writer.write(im0)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
video_writer.release()
cv2.destroyAllWindows()
```

```
----------------------------------------->
#Security system
----------------------------------------->

From  time import time

Import  smtplib

from email_setting import  from_email, password, to_email

from  email.mime.multipart import  MIMEMultipart
from  email.mime.text import  MIMEText

import  cv2
import  torch

From ultralytics import  YOLO
From  ultralytics.utils.plotting import Annotator, colors
#------------------------------------------------------------->
#Server creation

server = smtplib.SMTP("smtp.gmail.com: 587")
server.starttls()
server.login(from_email, password)

#------------------------------------------------------------->


def send_email(to_email, from_email, object_detected=5):
    """Sends an email notification indicating the number of objects detected; defaults to 1
object."""
    message = MIMEMultipart()
    message["From"] = from_email
```

```
  message["To"] = to_email
    message["Subject"] = "Security Alert"
    # Add in the message body
    message_body = f"ALERT - {object_detected} objects has been detected!!"

    message.attach(MIMEText(message_body, "plain"))
    server.sendmail(from_email, to_email, message.as_string())



class ObjectDetection:
    def __init__(self, capture_index):
        """Initializes an ObjectDetection instance with a given camera index."""
        self.capture_index = capture_index
        self.email_sent = False

        # model information
        self.model = YOLO("yolov8n.pt")

        # visual information
        self.annotator = None
        self.start_time = 0
        self.end_time = 0

        # device information
        self.device = "cuda" if torch.cuda.is_available() else "cpu"

    def predict(self, im0):
        """Run prediction using a YOLO model for the input image `im0`."""
        results = self.model(im0)
        return results

    def display_fps(self, im0):
```

```
    """Displays the FPS on an image `im0` by calculating and overlaying as white text on a black rectangle."""
        self.end_time = time()
        fps = 1 / round(self.end_time - self.start_time, 2)
        text = f"FPS: {int(fps)}"
        text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 1.0, 2)[0]
        gap = 10
        cv2.rectangle(
            im0,
            (20 - gap, 70 - text_size[1] - gap),
            (20 + text_size[0] + gap, 70 + gap),
            (255, 255, 255),
            -1,
        )
        cv2.putText(im0, text, (20, 70), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 2)

    def plot_bboxes(self, results, im0):
        """Plots bounding boxes on an image given detection results; returns annotated image and class IDs."""
        class_ids = []
        self.annotator = Annotator(im0, 3, results[0].names)
        boxes = results[0].boxes.xyxy.cpu()
        clss = results[0].boxes.cls.cpu().tolist()
        names = results[0].names
        for box, cls in zip(boxes, clss):
            class_ids.append(cls)
            self.annotator.box_label(box, label=names[int(cls)], color=colors(int(cls), True))
        return im0, class_ids
```

```
    def __call__(self):
        """Run object detection on video frames from a camera stream, plotting and showing the
results."""
        cap = cv2.VideoCapture(self.capture_index)
        assert cap.isOpened()
        cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
        frame_count = 0
        while True:
            self.start_time = time()
            ret, im0 = cap.read()
            assert ret
            results = self.predict(im0)
            im0, class_ids = self.plot_bboxes(results, im0)

            if len(class_ids) > 0:  # Only send email If not sent before
                if not self.email_sent:
                    send_email(to_email, from_email, len(class_ids))
                    self.email_sent = True
            else:
                self.email_sent = False

            self.display_fps(im0)
            cv2.imshow("YOLOv8 Detection", im0)
            frame_count += 1
            if cv2.waitKey(5) & 0xFF == 27:
                break
        cap.release()
        cv2.destroyAllWindows()
        server.quit()
detector = ObjectDetection(capture_index=0)
detector()
```

7.2 Screenshots

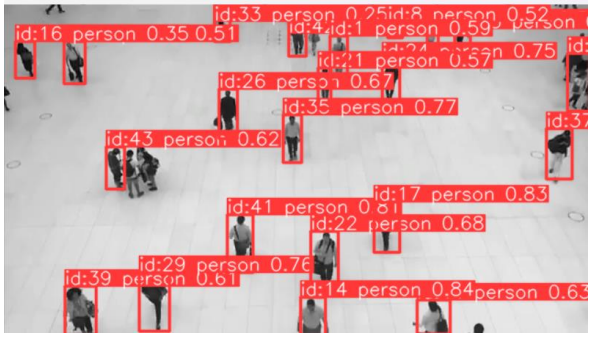

Figure 7.1: Crowd Detection

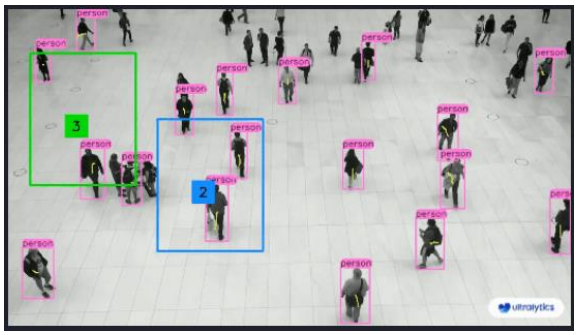

Figure 7.2: Crowd Analysis

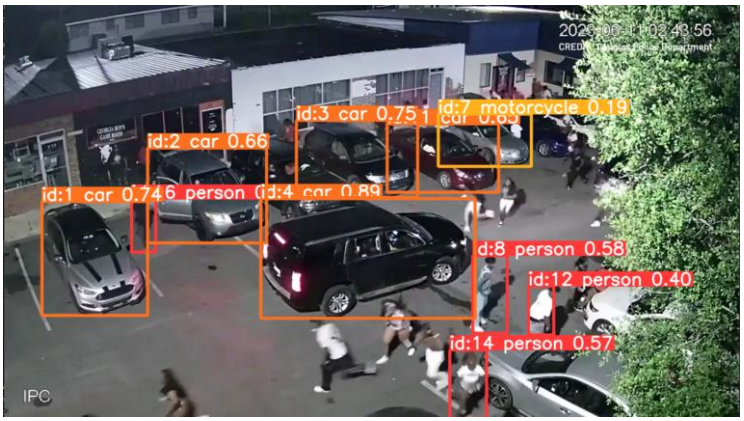

Figure 7.3: Object Detection

Figure 7.4: Knife detection

Figure 7.5: Knife detection from video

Figure 7.6: Knife detection from video