FPT UNIVERSITY

FPT UNIVERSITY

Bui Quang Minh - HE170776
Bui Quang Minh - HE170952
Pham Tien Anh - HE170096

Recognizing Restricted Area Intrusion With Computer Vision

GRADUATION THESIS
Major: Artificial Intelligence

Supervisor: PhD. Tran Van Ha

HA NOI - 2024

THANKS

First of all, we would like to express our sincere thanks to all the teachers at FPT University, Hanoi, especially the teachers and teachers working at the Faculty of Artificial Intelligence, for creating a lot of conditions and enthusiastically helping our students.

In particular, we would like to send a deep thank you to Dr. Tran Van Ha for wholeheartedly guiding, helping, and motivating us during our study at FPT University, Hanoi, in general, and in the future, throughout the course of research and the completion of the thesis in particular.

In addition, we would like to thank the teachers and fellows, and all the students who are currently working and participating in research at the shared Artificial Intelligence laboratory has share with us a lot of useful knowledge. Many of the things we learned from everyone helped us greatly during our research.

Finally, we would like to thank our family and friends for always accompanying us and encouraging us during difficult times. Everyone is always a source of motivation for us to strive to learn and improve ourselves.

ABSTRACT

With the global rise in population and increasing urbanization, video surveillance has become an essential tool for ensuring public safety and security. Traditional surveillance systems often struggle with real-time detection and tracking, limiting their effectiveness in monitoring restricted areas. To address these challenges, this thesis focuses on developing an efficient intruder detection system capable of accurately tracking individuals in real time. Our approach integrates the You Only Look Once (YOLO) object detection framework, known for its high-speed and accurate performance, with ByteTrack, a state-of-the-art multi-object tracking algorithm. By combining these two methods, our system can not only detect individuals but also track their movements continuously, ensuring robust monitoring of restricted areas. The experimental results demonstrate that the YOLOv8 model combined with ByteTrack achieves an impressive precision of 91.53% and is capable of real-time performance with a relatively fast inference time of 17.52 ms.

Keyword: *object detection, object tracking, YOLO,ByteTrack.*

Contents

List of figures

CHAPTER 1. INTRODUCTION

With the global rise in population and rapid urbanization, the demand for effective video surveillance systems has increased. Such systems have become crucial for maintaining public safety, securing sensitive areas, and managing large-scale urban environments. Despite numerous advancements in surveillance systems over the years[13], many existing methods continue to rely on frameworks has been around for a long time or lack functionality and effectiveness. These systems primarily focus on detecting and recording individuals and some may lack the ability to identify and respond to intrusions in real-time. These limitations can have significant consequences, particularly in situations where quick detection and action are essential for maintaining security. Without an effective intrusion detection system, restricted areas might become vulnerable to unauthorized access, theft, or other security breaches.

Our work's main goal is to develop a system that can detect people effectively and perform fast tracking in real-time. Furthermore, the system must have a mechanism to detect potential intruders and show alert. This feature ensures that restricted areas are closely monitored, minimizing the risk of unauthorized access and enabling responses to security threats. The system must be able to detect effectively with a high accuracy in real-time. This allows for practical applications in secure public surveillance and industrial safety, where reliable and cost-effective object detection is essential.

CHAPTER 2. RELATED WORK

2.1 Real-time object detection

In tackling problems such as intruder detection and surveillance management, modern object detectors like the latest YOLO models have become indispensable due to their ability to offer both high accuracy and speed, crucial for real-time security applications[15]. This balance between accuracy and processing speed makes YOLO ideal for systems that require fast processing[6]. Figure 2.1 shows different versions of the YOLO architecture.

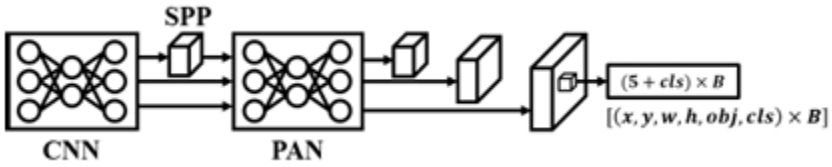

Figure 2.1: Architecture of YOLOv4, Scaled-YOLOv4, YOLOv5 r1–r7.[16]

Older versions of YOLO such as YOLOv3 have been used to detect humans in restricted areas near truck dumpers[18]. This particular system is aimed towards increasing workplace safety by preventing accidents from occurring in these dangerous areas due to the presence of people in those areas. Such a system is crucial for real-life work safety, as it enables real-time monitoring and immediate alerts to potential accidents. Figure 2.2 shows the framework of the model. An IP camera takes the video of the truck dumper zone, then a human detection algorithm analyses the video content to look for signs of people in the controlled zone. Using the fully convolutional neural network approach, the YOLO system detects people around blind areas and cuts off the dumper's movement when a person is in the restraining zone. There are two such servers willingly available among the server based control networks. While this system demonstrates exceptional effectiveness in detecting individuals, it lacks function to acknowledge user when individuals are in prohibited zone.

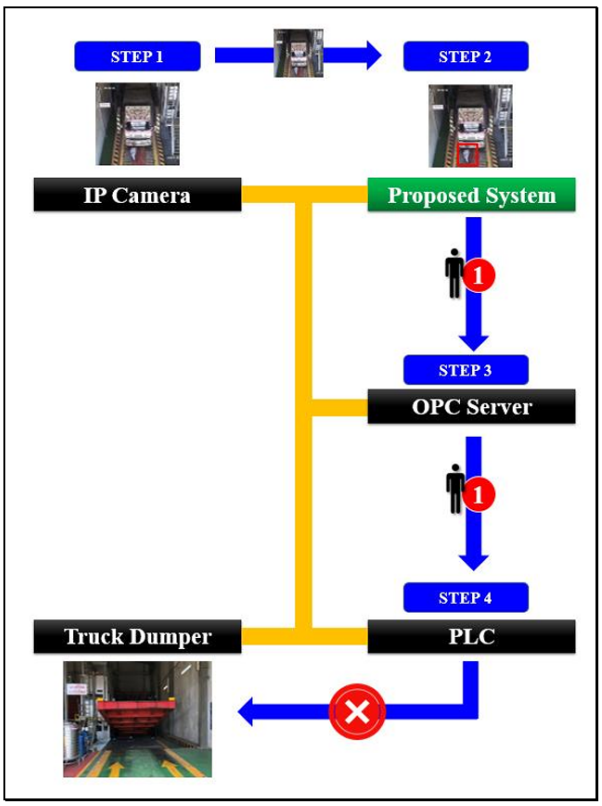

Figure 2.2: Human safety in the truck dumping system framework[18]

Other researchers have successfully combined the models of YOLOv3 and YOLOv3-tiny with the DeepSORT Algorithm for tracking problems[11].The system uses fast deep learning algorithms where YOLOv3 as the primary model is used for human detection and classification, in combination with the DeepSORT which is employed for tracking and counting individuals crossing an intrusion line, this function is essential in detecting unauthorized access. The low performance of the YOLOv3 model when deployed on central processing units (CPUs) is minimized by converting it to a TensorFlow format which is then deployed on graphical processing units (GPU) making it much more efficient than the traditional CPU based implementations. Figure 2.3 shows the example of the said model. This said, the system does have its weaknesses considering that it suffers from low performance with low resolution video inputs and is also limited to Nvidia CUDA compatibility for TensorFlow. It is important to note that although the YOLOv3-tiny model is quicker, it also notably loses accuracy in more complex settings. The study does point out the prospects of development – such as the usage of more precise configurations of YOLOv3 and better GPU hardware for achieving and performance results improvements.

Figure 2.3: Human detection and counting system using YOLOv3[11]

The Lightweight Convolutional Neural Network (L-CNN) has shown the capability to successfully perform human-object detection specifically from the video images of a smart surveillance system[12].Edge computing operates by performing data processing close to the source of the data, which helps to reduce delays in communication and lessens the strain on the network by minimizing the need to send large amounts of data to centralized servers. Leveraging edge computation and targeting low power embedded systems, the proposed model, L-CNN which incorporates ideas from depthwise separable convolution and Single Shot Multi-Box Detector (SSD) bases its search around the human objects present in the video frames. Figure 2.4 shows the result of the model. A prototype of the system was built on a Raspberry Pi 3 and tested using video surveillance streams from the field. According to the experimental results obtained L-CNN could apply satisfactory frame retrieval accuracy and show a fair frame processing speed, whilst being able to perform difficult tasks like recognition of partly hidden human objects. This makes it possible to describe L-CNN as a suitable technology for computing resource constrained smart surveillance systems. However, despite its advancements, the YOLO system can still struggle with relatively low frames per second, making it challenging to observe fast-moving objects or maintain smooth real-time surveillance.

Figure 2.4: Human tracking and detecting using L-CNN[12]

2.2 Real-time object tracking

In high risk cases like the case of an intruder detection, surveillance and other related security systems, achieving both high precision and small processing time must be ensured in order to provide a real time efficiency. For example, if someone were to invade a designated area, it should be shown that that person was present and acted appropriately without a considerable amount of time loss. With the advanced algorithms such as SORT[5],Bot-SORT[4] and DeepSORT[17](Deep Learning-based SORT), it is also possible to extend the tracking of multiple detected objects over time and across frames, since this algorithm ensures the consistent identification of objects through several frames. By assigning each person that has been identified a specific identification number, DeepSORT not only recognizes the objects but also tracks them over time as they move within the scene. This feature is vital for providing an uninterrupted and steady trace of every entity, avoiding complications such as ID flipping or losing scope of the object altogether. StrongSORT[8] is a noticeable improvement of DeepSORT for the interaction's case where scenes are often cluttered, objects are fast moving or are momentarily occluded. One of the important enhancements of StrongSORT is improved Kalman filtering. Figure 2.5 shows the framework and performance between DeepSort and StrongSort. Kalman filtering is a statistical prediction algorithm that's primarily used for estimating the future location of the tracked object based on its last known

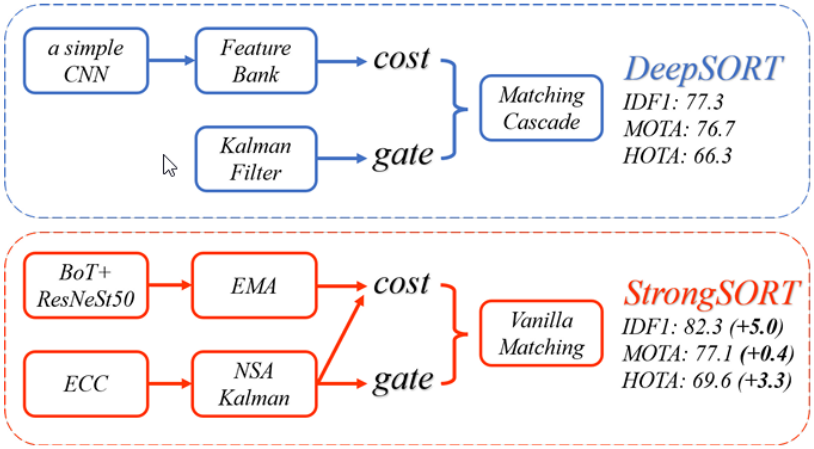

Figure 2.5: Framework comparison between DeepSORT and StrongSORT[8]

position and velocity. As with other trackers, the Kalman filter in StrongSORT has been adjusted to manage frequent position changes to allow the tracker to more closely anticipate and track object in the case where the object moves erratically or there is interference within the visual capture.

StrongSORT as well uses depth information in its tracking algorithm. This has a key advantage in 3D space or when an object can be either far or close to the camera. Depth information helps to identify the objects located at other distances and thus minimizes the ID switch over which many object track systems struggle. In traditional systems, the close proximity of several tracking objects, or their mutual movements, often leads to the situation when these objects are misidentified or swapped. Using depth perception, StrongSORT is capable of maintaining tracking of each object more accurately, in the presence of many other objects and even occlusion.

CHAPTER 3. METHODOLOGY

3.1 System overview

The system for detecting intrusions and showing alerts is created to monitor a restricted area and deliver on screen notifications to the users through security monitors. Essential components include cameras to view the area, along with a real-time video processing module. Figure 3.1 demonstrates the system pipeline. If the system detect an intrusion, it automatically triggers an alert on screen.

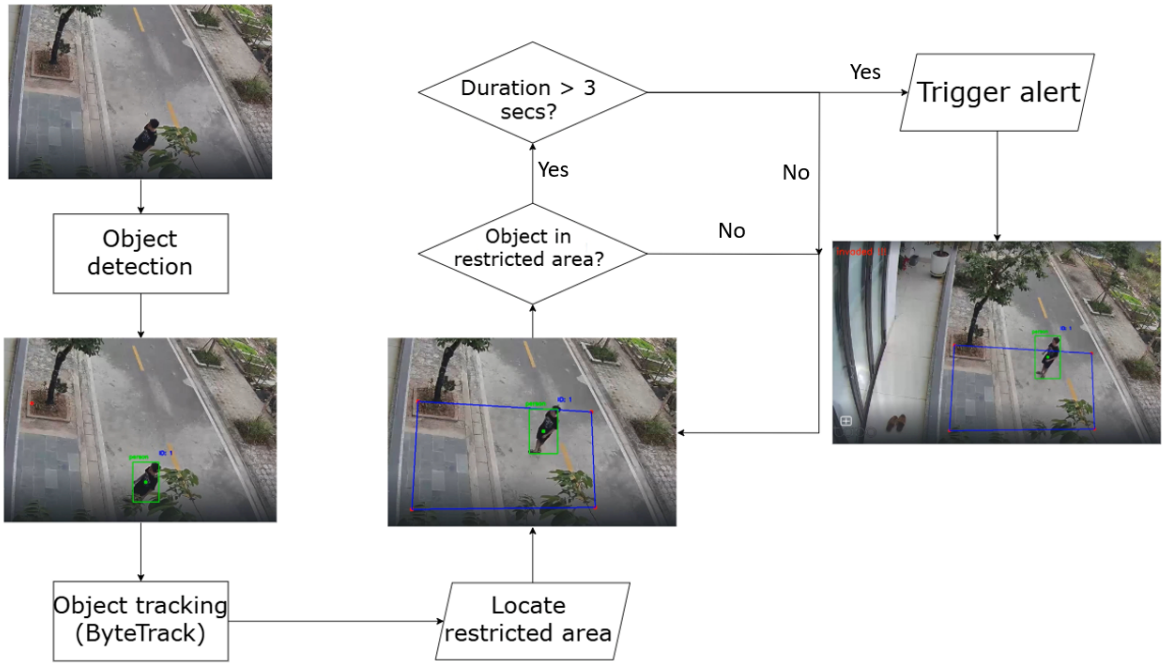

Figure 3.1: System pipeline

The pipeline of the system, as illustrated in the figure 3.1, begins with object detection, where the system identifies people in the surveillance footage or live surveillance cameras. Firstly, the model will detect people and enclosed them in bounding boxes, demonstrating the model's ability to recognize objects in the scene. Next, the model employs object tracking using ByteTrack, a robust tracking

algorithm that assigns unique IDs to detected objects and continuously tracks their movement across frames. This process ensures that the detected person is consistently tracked by preserving their unique ID as they move. Tracking is essential to determine whether an individual enters a restricted zone and remains there for a specified duration.

The system determines the restricted area using coordinate points obtained through mouse clicks on the screen. The restricted area is marked using a boundary which is a blue polygon, allowing the model to determine whether a detected person is within the prohibited zone. By continuously monitoring the person's movement and position, the model evaluates whether the individual violates the predefined boundary. The final step involves decision-making and alert triggering. If the detected person remains in the restricted area for more than 3 seconds, the system triggers an alert that shows on screen. Otherwise, tracking continues, ensuring that only people who stay too long in the restricted area are flagged as violations.

3.1.1 Model's detection and tracking details

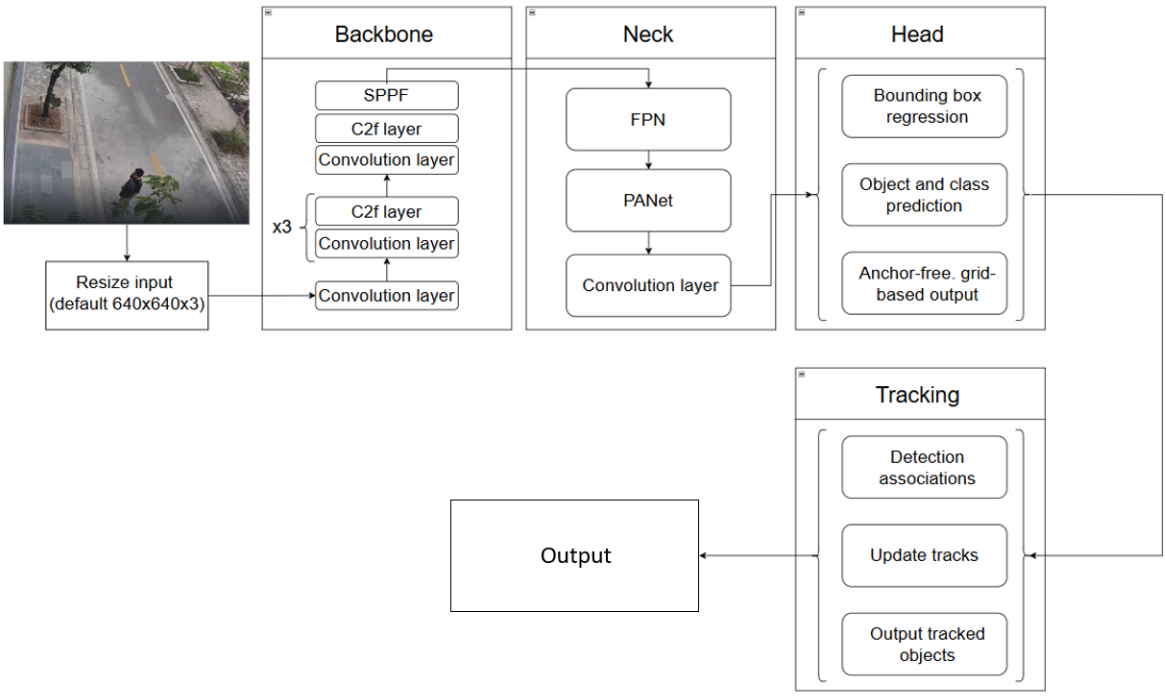

Figure 3.2: Model's workflow detail

Figure 3.2 illustrates the model workflow. Firstly an input image is extracted from surveillance footage or live surveillance camera. The input image resized to a

default resolution of 640x640x3 to ensure compatibility with the neural network. This resized input is then passed into the backbone section, where multiple convolutional layers, C2f layers, and a Spatial Pyramid Pooling-Fast (SPPF) module extract essential features from the image. These feature extraction layers help in capturing spatial and semantic information, which is crucial for accurate object detection. Following feature extraction, the processed data moves into the neck section, which consists of a Feature Pyramid Network (FPN) and Path Aggregation Network (PANet). These components enhance the representation of features at multiple scales, allowing the model to detect objects of varying sizes effectively. The neck refines the extracted features through additional convolutional layers before forwarding them to the next stage.

In the head section, the refined features are utilized to perform key detection tasks. This includes bounding box regression, which determines the precise location of detected objects, and object and class prediction, which classifies detected objects within the image. The model employs an anchor-free, grid-based output, optimizing detection performance by reducing computational complexity and improving efficiency in object localization. Once objects are detected, the system proceeds to the tracking phase. This involves detection association, where objects detected in consecutive frames are linked together, ensuring continuity in tracking. The model then updates tracks, maintaining unique object identities over time, and finally produces the output of tracked objects. This step is crucial in maintaining consistent object identification across multiple frames, enabling real-time tracking for surveillance or security applications. Finally, the model outputs the final results, which include detected and tracked objects along with their corresponding locations and class labels. This structured pipeline, integrating detection and tracking mechanisms, ensures high accuracy and efficiency in identifying and following objects in dynamic environments.

3.1.2 Prohibited zone mechanism

The prohibited zone mechanism is designed to monitor and detect unauthorized presence within a predefined restricted area. In the system, a restricted area is created using sets of coordinates obtained through user clicks on the screen. These coordinates will form an invisible polygon using Polygon() method from the Shapely[9] library. The polylines() method in OpenCV[7] will then be used to make

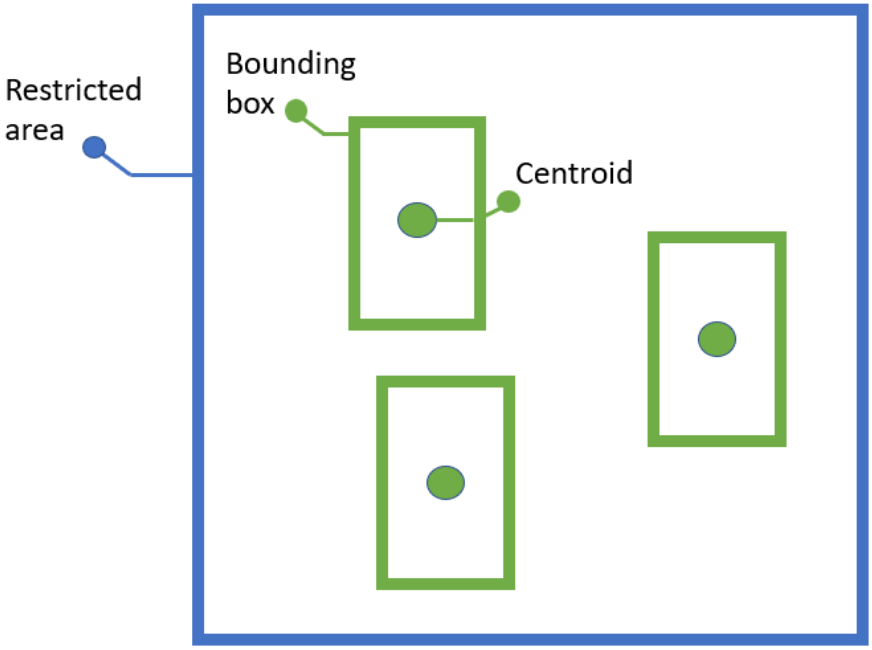

Figure 3.3: Prohibited zone function

the polygon visible by drawing a continuous blue boundary around the restricted area. This area is designated as off-limits, and any object entering it is subject to further analysis to determine if an alert should be triggered. Figure 3.3 shows the diagram of the prohibited zone function.

Objects detected in the scene are enclosed within green bounding boxes. The system computes the centroid of each bounding box (marked as green dots) to track the precise location of the object within the frame. The centroid is calculated by averaging the horizontal and vertical positions of the top-left and bottom-right corners of the bounding box, the centroid is determined as the midpoint of the bounding box. This centroid-based approach ensures that even if an object partially enters the restricted area, it is only considered a violation if its centroid crosses into the designated region.

Once an object's centroid is detected within the restricted area, the system initiates a tracking mechanism to assess whether the intrusion is temporary or not. This is achieved using the contains() method from the Shapely[9] library, which checks if the object's centroid falls within the defined restricted zone. If the object remains inside for a predefined duration (e.g., three seconds), the system flags the event as a violation and trigger an alert that shows on screen. This approach

helps minimize false alarms caused by accidental intrusions while ensuring accurate monitoring of unauthorized access. Overall, this prohibited zone detection mechanism enhances security by effectively identifying intrusions and reducing unnecessary alerts. It is particularly useful in applications such as surveillance, industrial safety, and automated access control, where ensuring restricted area compliance is critical. Figure 3.4 shows the result of the said function after being implemented in the code.

Figure 3.4: Prohibited zone function example

3.2 Implement model

3.2.1 Object detecting

Developed by the Ultralytics team, YOLOv8[14] has significant improvements for real-time object detection. Traditional methods often used sliding window and anchor boxes, YOLOv8 improves speed and accuracy through advancements like anchor box free, a feature pyramid network, spatial attention modules, and data augmentation techniques. Although YOLOv8 has no official papers, insights from various studies show that YOLOv8 builds on YOLOv7 and utilizes a Darknet-53 neural network architecture for effective feature extraction before applying its object detection algorithm. Figure 3.5 shows the architecture of the YOLOv8 model.

The C2f (Cross-Stage Partial with Fusion) block in YOLOv8 is an advanced feature extraction module designed to improve efficiency and performance in object detection. It enhances feature representation while keeping computational costs manageable. The block follows a structured sequence of operations that facilitate

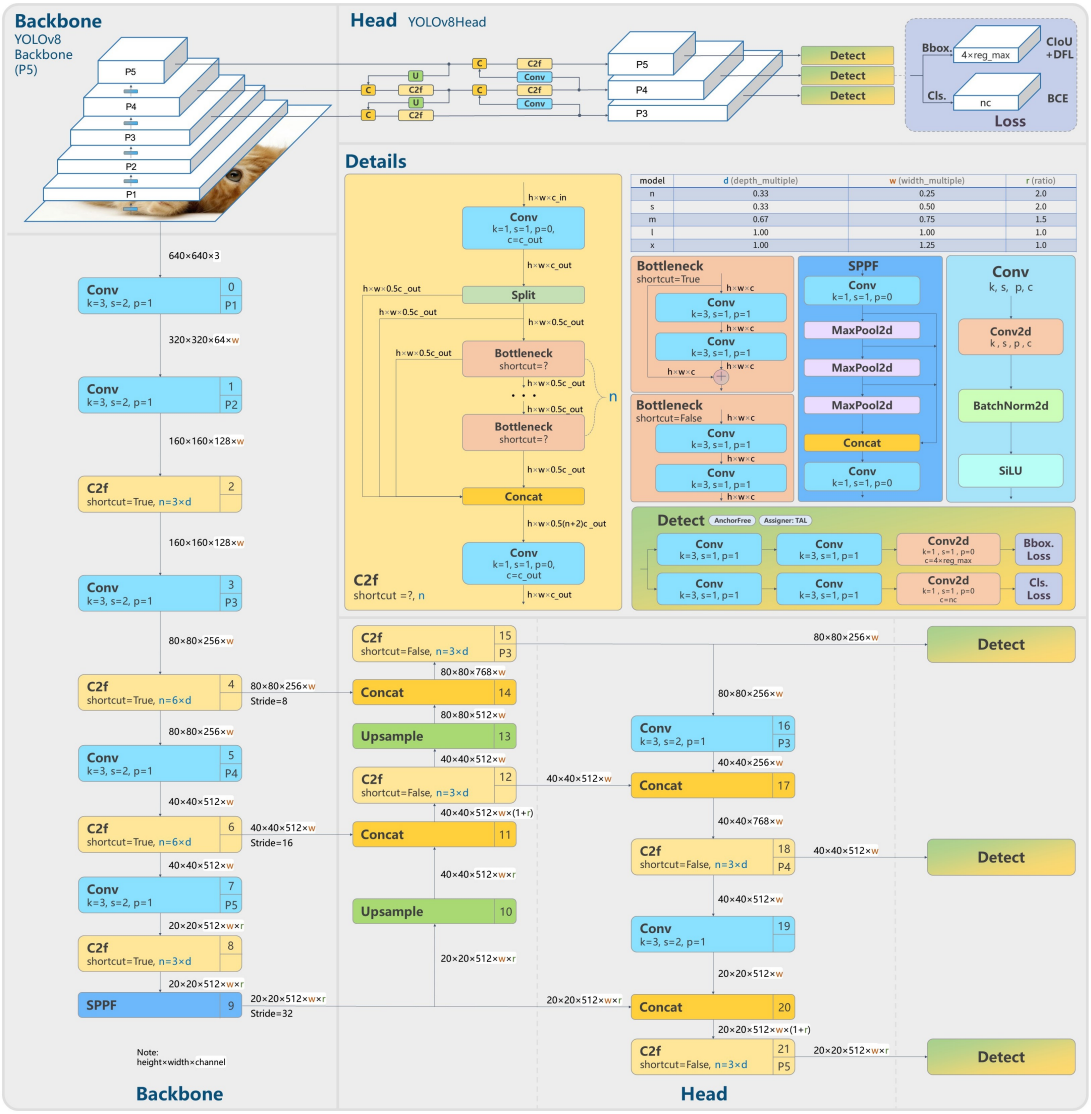

Figure 3.5: YOLOv8 structure[1]

information flow, gradient propagation, and fusion of diverse feature representations. The process begins with an initial 1×1 convolution applied to the input feature map of size h × w × c. This convolution adjusts the number of channels to c_out while maintaining the spatial dimensions. The main purpose of this step is to control the number of channels going into the block, ensuring efficient computation. Figure 3.6 illustrates the C2f blocks architecture.

After the convolution, the feature map is split into two equal parts. One part is directly passed forward as a shortcut, while the other half undergoes further processing through a series of Bottleneck layers. This design is inspired by Cross-Stage Partial Networks (CSPNet), where only a portion of the features is transformed, reducing redundant computation while preserving important information.

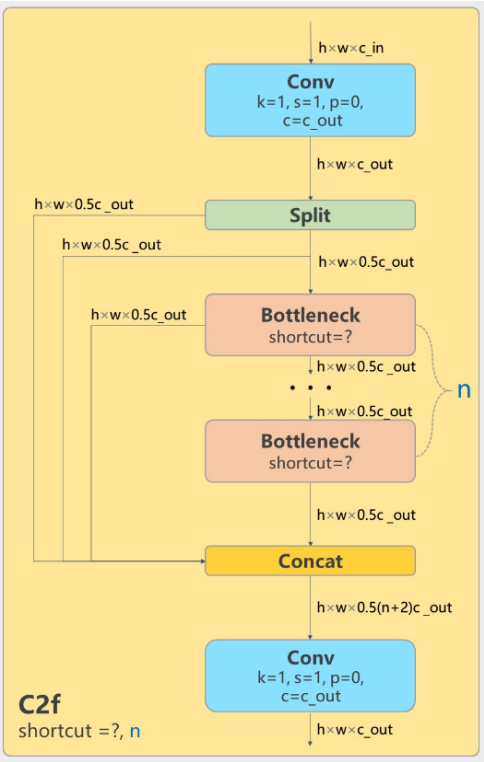

Figure 3.6: C2f blocks in YOLOv8 architecture

Within the transformed branch, the bottleneck layers apply a sequence of transformations, each consisting of convolutions, batch normalization, and activation functions. These bottleneck layers enhance feature extraction by capturing spatial and contextual information while maintaining efficient memory usage. The residual connections in these layers, indicated as shortcut, might be optionally enabled, helping with gradient flow and avoiding vanishing gradient issues.

Once the bottleneck transformations are completed, their outputs are concatenated along the channel dimension with the unprocessed shortcut features. This results in an expanded feature map with a total of (0.5(n+2) * c_out) channels, effectively fusing both raw and processed information. The concatenation operation allows the network to retain both low-level and high-level features, improving detection accuracy. Finally, another 1×1 convolution is applied to regulate the number of output channels back to c_out, ensuring that the feature map remains compatible with the next layers of the model. This final step compacts the information while retaining the rich feature representations generated by the bottleneck layers.

The Bottleneck block is a fundamental feature extraction unit, consisting of two 3×3 convolution layers with stride 1 and padding 1. It comes in two variations:

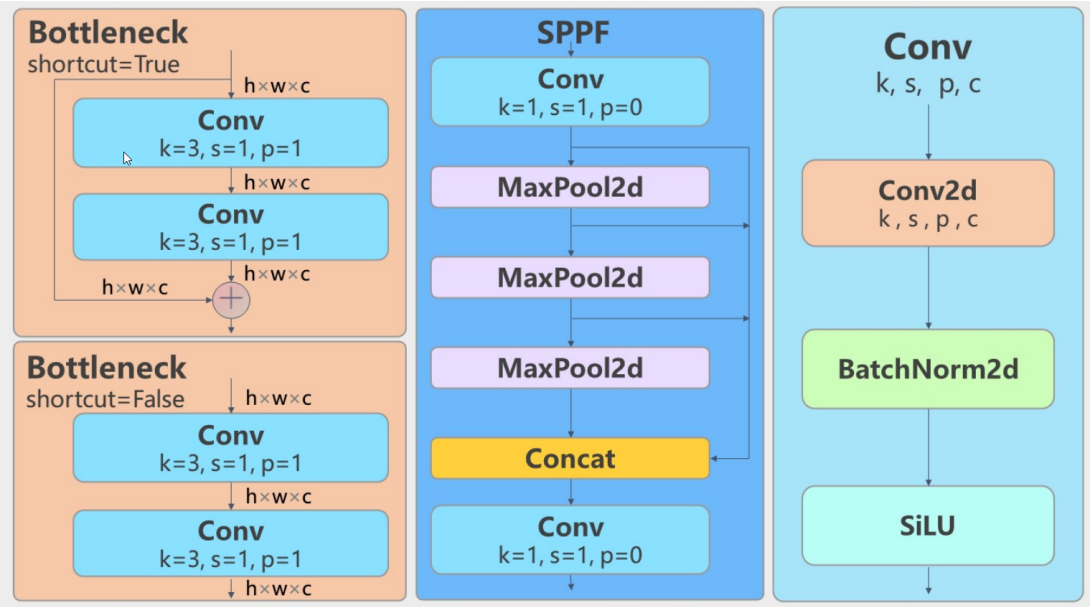

Figure 3.7: Bottleneck, SPPF and Conv blocks in YOLOv8 architecture

one with a shortcut (residual connection) and one without. Figures 3.7 shows the architecture of the Bottleneck blocks, SPPF and Conv blocks. The shortcut-enabled bottleneck introduces a direct connection from the input to the output, enabling efficient gradient propagation and mitigating vanishing gradient issues. This residual connection helps retain essential information and stabilizes training. On the other hand, the shortcut-disabled bottleneck simply passes data through two convolution layers without an identity connection, which may be useful in deeper architectures where feature transformation is prioritized.

The SPPF block (Spatial Pyramid Pooling-Fast) is a specialized structure designed to enhance the receptive field of the network, enabling it to capture features at multiple scales efficiently. It starts with a 1×1 convolution for feature compression, followed by three sequential MaxPool2d layers, each reducing the spatial resolution. The outputs from different levels of pooling are then concatenated, effectively aggregating multi-scale features. Finally, another 1×1 convolution refines the fused representation, ensuring that the feature map is both compact and informative. The SPPF block helps models recognize objects at different scales while keeping computational costs lower than traditional pyramid pooling layers.

The Conv block represents the standard convolutional unit in modern CNN architectures. It consists of a 2D convolution layer (Conv2d), followed by Batch Normalization (BatchNorm2d) and the SiLU (Sigmoid Linear Unit) activation function. BatchNorm2d helps stabilize training by normalizing feature distributions,

preventing internal covariate shifts. The SiLU activation function is a smooth, non-linear activation that enhances gradient flow and improves network expressiveness, leading to better feature learning compared to traditional activations like ReLU.

A key function of the neck is upsampling and downsampling feature maps to align their resolutions. Upsampling, often achieved through bilinear interpolation or transposed convolutions, increases the spatial resolution of lower-resolution feature maps, making them compatible with higher-resolution ones. These upsampled feature maps are then combined—via addition or concatenation—with corresponding higher-resolution feature maps from the backbone. This operation preserves spatial details critical for detecting small objects while simultaneously leveraging the semantic richness of deep-layer features. After aggregation, the neck applies additional convolutional layers to refine the fused features, reducing noise and distilling the most relevant information for the detection task. The output of the neck consists of high-quality, multi-scale feature maps optimized for detecting objects at different locations and scales in the image.

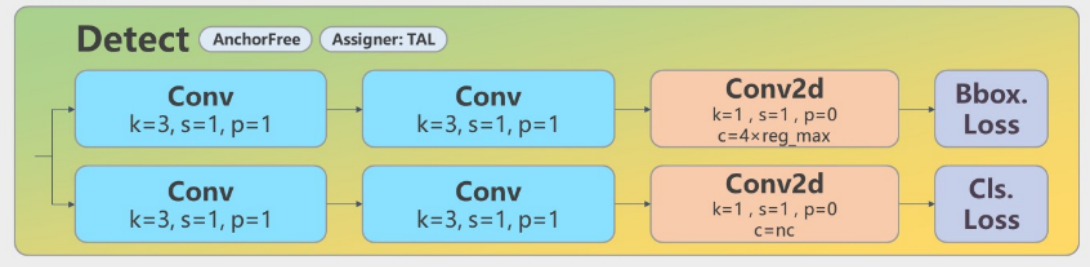

Figure 3.8: Detection head in YOLOv8 architecture

The head in YOLOv8 is responsible for performing the final object detection, transforming the refined feature maps from the neck into predictions of bounding boxes, class labels, and confidence scores. Figure 3.8 illustrates the architecture of the head section. This component plays a central role in identifying and localizing objects within the image with high precision and speed. The head predicts bounding box coordinates for each object in the image, typically represented as (x, y, w, h), where (x, y) denotes the center of the bounding box relative to the image dimensions, and w, h represent its width and height. These predictions are computed directly from the feature maps, allowing for efficient and anchor-free detection. Furthermore, it uses Task-Aligned Learning (TAL) as the assignment strategy, which helps in matching predictions to ground truth objects effectively.

The final predictions are supervised by bounding box and classification loss functions, ensuring precise object detection.

3.2.2 Object tracking

ByteTrack presents a method for Multi-Object Tracking that associates every detection box. It was introduced at ECCV2022 by Yifu Zhang et al[19]. and has been widely used by other researchers for their MOT trackers like Bot-SORT and SMILEtrack due to its versatile framework and simplicity.

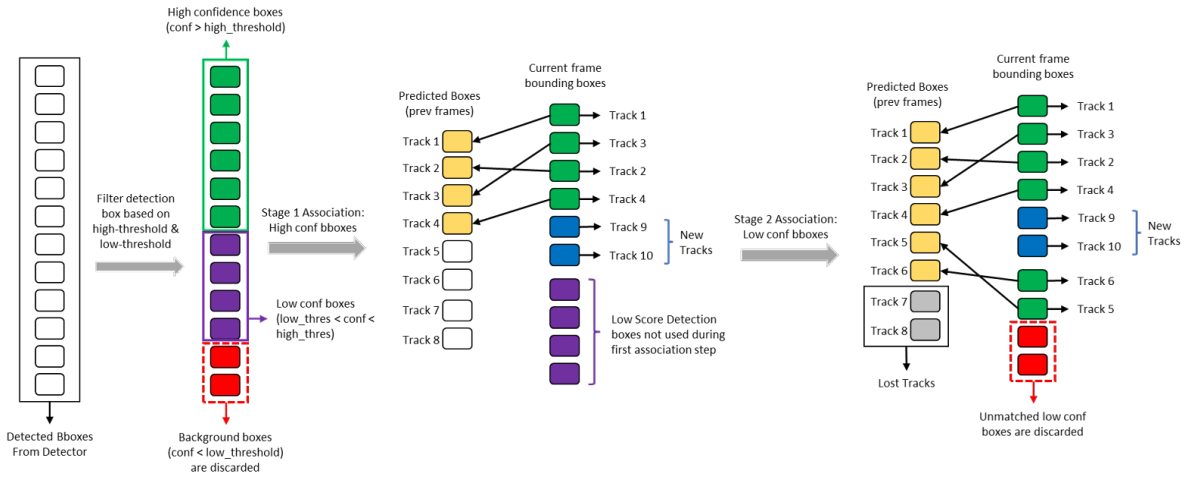

Figure 3.9: ByteTrack algorithm[2]

Figure 3.9 shows how ByteTrack algorithm works. It starts by detecting objects in each frame using an object detector - in our system this is YOLOv8. Each detected object is assigned a confidence score, which indicates how sure the detector is about the object's presence. Based on this score, detections are categorized into high-confidence detections and low-confidence detections. Once the detections are categorized, ByteTrack uses a Kalman filter to predict where previously tracked objects should appear in the new frame. This helps account for movement between frames and ensures smooth tracking. After prediction, the algorithm performs the first matching step, where it tries to match the predicted object locations with high-confidence detections using an overlap measure called Intersection over Union (IoU) or, in some cases, Re-ID features. The Hungarian algorithm is used to find the best matches between predicted objects and detected objects. Any unmatched detections and tracks are set aside for the next step.

Next, ByteTrack performs the second matching step, where it tries to match the remaining, unmatched tracked objects with low-confidence detections. This

step is important because it helps keep track of objects that may be partially occluded or hard to detect. However, in this stage, only IoU-based matching is used (without Re-ID features) since low-confidence detections may not be reliable enough for appearance-based matching. Any objects that still cannot be matched are temporarily stored in a "lost" buffer, where they remain for a certain number of frames before being discarded. This allows ByteTrack to recover objects that were briefly missing due to occlusion. Finally, any remaining unmatched high-confidence detections are initialized as new tracks. This ensures that newly appearing objects are tracked properly. By maintaining a balance between tracking existing objects, recovering lost objects and adding new objects, ByteTrack achieves high accuracy in multi-object tracking scenarios.

CHAPTER 4. EXPERIMENT

4.1 Data collection

The dataset used for this study consists of a total of 18,250 image samples, all of the images are mainly human to help focus on recognizing and tracking people. This dataset is a combination of 8000 human images from COCO[10] dataset, 6000 images of surveillance camera images from author's workplace, and 4000 images of people in various environments sourced from public datasets on Roboflow and other sources. While the images from COCO come with pre-existing labels, the rest of the images are manually labeled using LabelImg[3] to ensure accurate bounding boxes for human detection and tracking. In this dataset, 15,550 images are used for training purposes, providing a diverse set of scenarios to enhance the model's learning capabilities. The remaining 2,700 images are implemented for validation, ensuring that the model's performance is carefully tested against unseen data. These images provide a wide range of environments and conditions, including different lighting scenarios, crowded scenes, and diverse perspectives, making the dataset highly reflective of real-world scenarios. Figure 4.1 shows the example of our training images.

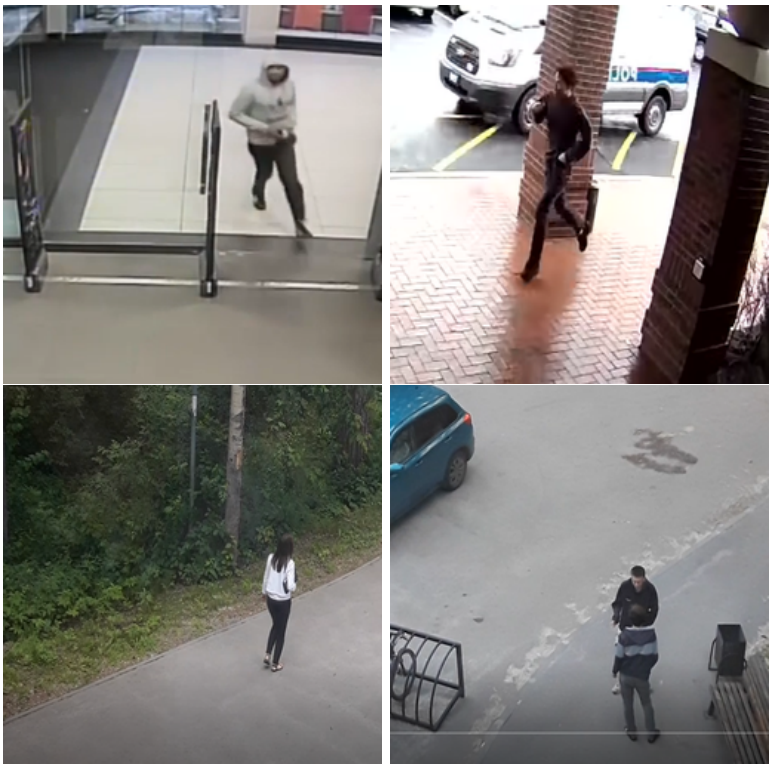

Figure 4.1: Training image example

4.2 Data preprocessing

To enhance the dataset diversity and improve the model's ability to generalize effectively to various real-world scenarios, a range of data augmentation techniques is employed:

Hue: This technique adjusts the hue values of an image, effectively shifting its color tones. By simulating changes in lighting conditions or imaging environments, hue adjustment helps the model handle variations in color perception.

Saturation: Saturation augmentation modifies the intensity of colors in an image. It can make colors appear more intense by increasing saturation or neutral by reduced saturation. This technique is particularly useful for preparing the model to handle over-saturated or under-saturated images that might arise due to imaging devices or lighting inconsistencies.

Brightness: Brightness alteration changes the luminance levels of an image, simulating different lighting environments such as bright daylight or dim indoor settings. By increasing or decreasing the brightness, this technique helps the model learn to interpret images accurately under diverse illumination conditions.

Translation: Translation shifts the image horizontally, vertically, or both, without altering its content. This technique ensures the model can recognize features regardless of their position in the frame, helping it generalize to scenarios where objects may not be perfectly centered or aligned.

Scale: Scaling involves resizing the image while maintaining its aspect ratio, simulating objects appearing at different distances or sizes. This augmentation technique enhances the model's ability to recognize patterns and features across varying scales, ensuring robustness to size variations.

Flip: Flipping creates mirror-image versions of the input data by horizontally, vertically, or diagonally flipping the image. This technique increases dataset diversity and helps the model generalize to features that may appear in mirrored orientations in real-world scenarios.

Erasing: Random erasing involves replacing a random portion of the image with a solid color or noise. By occluding certain areas, this technique forces the model to focus on the remaining visible features and improves its robustness to partial occlusions or missing data.

Crop: Cropping selects a random portion of the image and resizes it to the original dimensions. This technique helps the model focus on specific regions of interest while reducing reliance on global context, which is especially useful for analyzing localized features.

Blur: Blurring applies a smoothing effect to the image using techniques like Gaussian blur or motion blur. This augmentation simulates conditions where images may be slightly out of focus or affected by motion, training the model to perform well even under such distortions.

4.3 Model training results

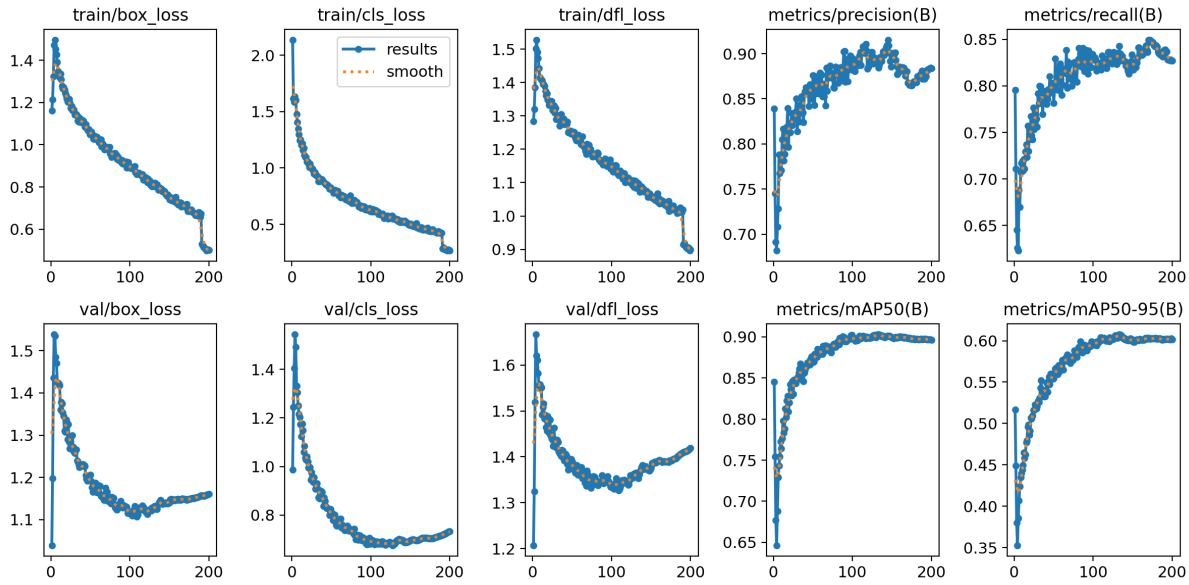

Figure 4.2: Training results

The model was trained for 200 epochs. We use batch size of 3 with an input image of 640x640. The GPU used for training was RTX2080Ti. Figure 4.2 shows the results of our training process. The model achieved the best result on epoch 145 with a precision of 91.53%, recall at 81.28%, mAP50 at 90.15%, mAP50-95 at 60.12% and the inference time is 17.52 ms. The precision score indicates that the model effectively avoids false positives, meaning it is highly accurate when identifying people in the frame. With a precision of 91.53%, the model successfully differentiates people from the background or other objects in more than 91 out of 100 cases. This high precision is an crucial for applications where false alarms need to be minimized, such as intrusion detection systems or prohibited zone monitoring. With the inference time of 17.52 ms, this indicates that the model can quickly process and analyze video. This is essential for security applications, where delayed detections could lead to missed intrusions or slower response times.

The recall score measures the model's ability to detect all actual instances of people in the scene, and at 81.28%, it is reasonably good. This indicates that the model successfully detects about 81% of the people present. However, the remaining 19% of missed detections could pose a problem in scenarios where complete coverage is crucial such as crowd management. Improving recall would involve ensuring the model is exposed to a wider range of challenging scenarios during training, such

as crowded or occluded scenes. Figure 4.3 illustrates the model's detection results.

Figure 4.3: The model's detecting results

The mean Average Precision (mAP) at a 50% Intersection over Union (IoU) threshold highlights the model's ability to both detect and localize people accurately. With a score of 90.15%, the model performs well in balancing detection and localization, even when the IoU threshold allows for some overlap between predicted and ground truth bounding boxes. This indicates strong overall model performance for general detection tasks, showing that the model can consistently identify and localize people within an acceptable margin of error. The mAP50-95 metric evaluates the model's performance across a range of IoU thresholds, requiring stricter alignment between predicted and ground truth bounding boxes. The lower score of 60.12% suggests that while the model is good at detecting people, its bounding box predictions are less precise when stricter localization accuracy is required. Table 4.1 shows the performance of our model. Overall, the model demonstrates strong capability in accurately detecting people, showcasing its effectiveness in real-world scenarios.

Table 4.1: Our model's performance

Precision(%)	Recall(%)	mAP50(%)	mAP50-95(%)	Inference time(ms)
91.53	81.28	90.15	60.12	17.52

CHAPTER 5. CONCLUSION

This thesis integrates YOLOv8 and ByteTrack to improve accuracy and processing speed in monitoring restricted areas. The project's practical applications are noticeable in secure public surveillance and industrial safety, where reliable and cost-effective object detection is essential. The system can be used to detect intrusions or enhancing the security of public spaces. By demonstrating that high-precision detection can be accomplished with fast processing speed, this work provides a practical solution for real-world applications, making advanced object detection more accessible for security and surveillance systems. It can be used in the deployment of robust detection systems in diverse environments such as urban infrastructure, smart homes, and autonomous systems.

In this paper, we implement a method for monitoring and tracking people, designed to address the limitations of previous approaches and improve detection latency in real-world scenarios. By fine-tuning the YOLOv8 model combined with ByteTrack, our system provides effective intrusion alert. Experimental results show significant results with a precision of 91.53% and an inference time of 17.52 ms, proving its performance in tracking speed and accuracy.

However, there are limitations that need to be solved in the future. Firstly, the intrusion alert system still have some drawbacks. This function only works properly when the camera angle is set in a executable place. If the camera is set in an environment that is hard to draw intrusion zone, this function might not work that well. Furthermore, the intrusion zone required the target to stand in the zone for 3 seconds for it to work. If the target quickly run in and run out, the system can only detect but might not be able to show alerts to user. In addition, a function capable of sending alerts to users is essential. Further evaluation is needed on smaller computers such as Raspberry Pi or real-life surveillance hardware like webcams to further prove its applicability. Testing on these devices would allow us to validate the model's versatility and real-world application.

References

[1] https://github.com/ultralytics/ultralytics/issues/189.

[2] https://www.datature.io/blog/introduction-to-bytetrack-multi-object-tracking-by-associating-every-detection-box.

[3] https://github.com/HumanSignal/labelImg.

[4] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[8] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*, 25:8725–8737, 2023.

[9] Sean Gillies et al. Shapely: manipulation and analysis of geometric objects, 2007–.

[10] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[11] Hamam Mokayed, Tee Zhen Quan, Lama Alkhaled, and V Sivakumar. Real-time human detection and counting system using deep learning computer vision techniques. In *Artificial Intelligence and Applications*, volume 1, pages 221–229, 2023.

[12] Seyed Yahya Nikouei, Yu Chen, Sejun Song, Ronghua Xu, Baek-Young Choi, and Timothy R Faughnan. Real-time human detection as an edge service enabled by a lightweight cnn. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 125–129. IEEE, 2018.

[13] Rajat Singh, Sarvesh Vishwakarma, Anupam Agrawal, and MD Tiwari. Unusual activity detection for video surveillance. In *Proceedings of the First International Conference on Intelligent Interactive Technologies and Multimedia*, pages 297–305, 2010.

[14] Mupparaju Sohan, Thotakura Sai Ram, Rami Reddy, and Ch Venkata. A review on yolov8 and its advancements. In *International Conference on Data Intelligence and Cognitive Informatics*, pages 529–545. Springer, 2024.

[15] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[16] Chien-Yao Wang and Hong-Yuan Mark Liao. Yolov1 to yolov10: The fastest and most accurate real-time object detection systems. *arXiv preprint arXiv:2408.09332*, 2024.

[17] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[18] Apirak Worrakantapon, Wattana Pongsena, Kittisak Kerdprasop, and Nittaya Kerdprasop. Real-time human detection in a restricted area for safety in truck dumper control system using deep learning. 2021.

[19] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European conference on computer vision*, pages 1–21. Springer, 2022.