Real-Time Detection and Tracking in Disaster Scenarios

Hariom Meena

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology

Under the guidance of Dr. Chalavadi Vishnu

Department of Computer Science and Engineering
Indian Institute of Technology, Tirupati
May 2025

DECLARATION

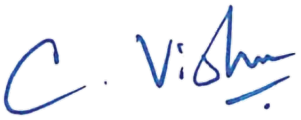
I declare that this written submission represents my ideas in my own words. Where others' ideas or words have been included, I have mentioned the references in the References section, referencing the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated, or falsified any idea/data/fact/source in my submission to the best of my knowledge. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Place: Tirupati
Date: 13-05-2025

Signature
Hariom Meena
CS23M106

Bonafide Certificate

This is to certify that the titled **Real-Time objects Detection and Tracking in Disaster Scenario**, submitted by **Hariom Meena**, to the Indian Institute of Technology, Tirupati, for the award of the degree of, is a bona fide record of the work done by him under our supervision. The contents of this, in whole or in part, have not been submitted to any other Institute or University for the award of any degree or diploma.



Place: Tirupati
Date: 13-05-2025

Dr. Chalavadi Vishnu,
Assistant Professor
Department of Computer Science
IIT Tirupati - 517619

Acknowledgments

Acknowledgements

I would like to express my deepest gratitude to my MTP guide, **Dr. Chalavadi Vishnu**, for his invaluable guidance, insightful feedback, and unwavering support throughout every stage of this project. His expertise and encouragement have been instrumental in shaping the direction and quality of my work.

I am also profoundly thankful to **Dr. R.K. Gorthy** for generously providing the hardware resources and technical assistance that made the implementation and experimentation possible. My sincere appreciation goes to the **Department of Computer Science and Engineering, IIT Tirupati**, for granting access to computational facilities, laboratory space, and essential software tools. The collaborative atmosphere and institutional support fostered an ideal setting for conducting this research.

I am deeply grateful to my family—whose patience, understanding, and constant encouragement have sustained me through the challenges of this endeavor—and to my friends, especially **Mr. Ayush Bilkhiwal**, whose moral support and thoughtful discussions provided inspiration and kept me motivated wherever I stuck.

Finally, I acknowledge all those—faculty, peers, and staff—whose contributions, whether large or small, have enriched this project and my academic journey.

Abstract

In the wake of natural disasters, timely and precise identification of individuals in need is critical to saving lives and optimizing rescue operations. This project presents a real-time person tracking system designed for deployment on unmanned aerial vehicles (UAVs), specifically tailored for use in challenging disaster environments such as floods, earthquakes, and landslides. Leveraging the capabilities of the YOLOv8 object detection architecture and optical flow-based motion tracking, our system is capable of detecting, identifying, and continuously tracking individuals from aerial video feeds captured by drone-mounted cameras.

To ensure robust performance under adverse visual conditions, such as low light, occlusions, or debris, the detection model is fine-tuned on disaster-specific datasets, including custom datasets. The system incorporates a lightweight, onboard inference pipeline optimized for edge devices. An integrated payload drop mechanism can further enable automated delivery of essential supplies such as medical kits and food packages to detected victims, minimizing the need for human intervention in hazardous terrain.

ABBREVIATIONS

- **UAV**: Unmanned Aerial Vehicle
- **YOLO**: You Only Look Once
- **SORT**: Simple Online and Realtime Tracking
- **DeepSORT**: Deep Learning-based SORT
- **StrongSORT**: Strong Sort with a deep association model
- **SOTA**: State Of The Art
- **UOD**: Underwater Object Detection
- **SSD**: Single Shot Detector
- **SAR**: Search and Rescue
- **mAP**: mean Average Precision
- **FPS**: Frames Per Second
- **IoU**: Intersection Over Union
- **CNN**: Convolutional Neural Network

Figures

Tables

Table of Contents

Chapter 1 Introduction

1.1 Motivation

After visualizing the Vijayawada Flood scenario I analysed a few points like:

1.1.1 Real-Time Detection and Tracking are Crucial

In a disaster, every second counts. Real-time detection systems allow authorities to monitor the situation as it unfolds, ensuring quicker responses. Whether it's spotting rising water levels or tracking people in danger, these systems help rescue teams act swiftly and effectively. They ensure that resources are directed to the areas where they're needed the most, reducing delays that could cost lives.

1.1.2 Quick Identification of People can Save Lives

Finding stranded or trapped individuals during a disaster is a huge challenge. Real-time detection systems, such as drones or cameras with AI, can quickly locate people in affected areas, even in tough conditions. This speeds up rescue efforts and helps deliver food, water, and medical aid to those in urgent need. It ensures no one is left behind, giving survivors hope and a better chance of survival.

1.1.3 Less Reliance on Manual Efforts

Traditional surveillance methods often put rescue workers in harm's way and take a lot of time. Automated systems, like drones or smart cameras, can do the heavy lifting by surveying dangerous

areas and providing accurate data. This reduces risks for rescue teams and allows them to focus on helping people, not scouting for information. Automation also speeds up the entire process, making disaster responses more efficient and safer.

1.2 Problem Statement

Traditional detection (like YOLO) and tracking systems (like DeepSORT and StrongSORT) often struggle to provide the speed, accuracy, and reliability needed in real-time disaster situations, especially when working with live data from drones or surveillance cameras. These older systems may not be fast enough to keep up with rapidly changing scenes, or they may miss important details, which can delay rescue efforts. Additionally, they often don't offer useful performance information, like how quickly they're working or how much memory they're using, which makes it harder to assess their effectiveness in urgent situations. This is why there is a need for a more efficient and automated solution that can both detect and track people and objects accurately and provide valuable performance data.

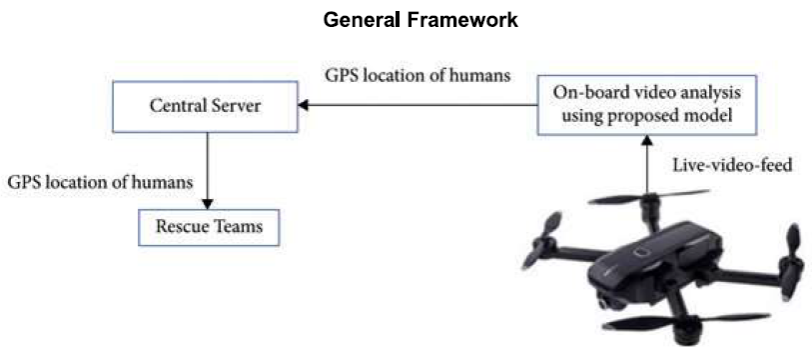


Figure 1.1: Problem Statement

- So, the primary objective is to develop a robust model based **YOLO** versions for object detection and integrate it with **StrongSORT** or other Tracking Algorithms like **Strong-**

SORT, **Transformer Based Tracker** for tracking, but the model should ensure the real-time performance, like tracks individuals across frames with unique IDs, and provides metrics like frame rate and memory usage on Real-Time data.

1.3 Report Outline

The report is structured as follows:

- Chapter 2: Literature Review
- Chapter 3: Methodology
- Chapter 4: Experiments and Results
- Chapter 5: Conclusion and Future Work

Chapter 2 Literature Review and Related Works

2.1 Literature Review

Modi et al. present a two-stage pipeline that first applies the one-stage YOLOv8s detector to each frame, isolating the object of interest (a soccer ball), then blurs the remaining background before feeding the masked frame into an optical flow tracker. By reducing background clutter, their method yields continuous, color-coded trajectory masks on a black canvas, even under rapid motion and partial occlusion. Trained and validated on the DFL Soccer Ball Detection dataset, this fusion achieves sub-second per-frame processing on embedded hardware while maintaining high trajectory accuracy.[1].

Abdelnabi and Rabadi's state-of-the-art review categorizes UAV-enabled person detection systems by disaster scenario (flood, earthquake, maritime), sensor modality (RGB, thermal, NIR), and algorithmic family (one vs two-stage CNNs, ensemble methods). They find that while detectors like YOLOv3/v4 and Faster R-CNN achieve over 90% recall on benchmark datasets, their computational footprint often exceeds typical UAV GPU capacities. The authors underscore the critical need for model compression (pruning, distillation) and multi-sensor fusion (e.g., visible + thermal) to sustain high detection rates under constrained edge computing budgets [2].

Vedanth et al. develop an end-to-end UAV system that fine-tunes YOLOv8 on oblique infrared video streams captured at 50–100 m altitudes. Leveraging transfer learning from COCO/PASCAL VOC pretrained weights, extensive angle and illumination augmentations, and inductive parameter transfer, their pipeline converges rapidly, delivering sub-500 ms inference on NVIDIA Jetson modules. Upon victim detection, the system autonomously triggers a precision payload-drop mechanism, achieving under 2 m drop-zone error—even in cluttered, low-contrast IR scenes [3].

Liu *et al.* introduce DisasterScope, a public remote-sensing corpus of 2765 high-resolution (640×640px) images spanning eight disaster types (floods, fires, landslides, etc.) and annotated for people, vehicles, debris, and hazard markers. They further propose an RTMDet-s architecture where standard 3×3 neck convolutions are replaced with 5×5 depthwise-separable kernels, expanding receptive fields without increasing parameter counts. On DisasterScope, this variant outperforms state-of-the-art detectors (YOLOv3, YOLOX-s, RTMDet-s) by up to 0.8mAP$_{50}$ at equivalent compute budgets, demonstrating the power of scenario-focused data plus architectural refinement for onboard, real-time inference [4].

Hadi et al. propose an intelligent system for deceased-victim detection using a Raspberry Pi, Sony Exmor R IMX117 NIR camera, odor sensor, and LoRa module mounted on a DJI Phantom 4. A MobileNetV2 classifier distinguishes live vs. dead humans with 99% accuracy at 2 m altitude under both day (0–60 lx) and night conditions, and maintains 99% performance at speeds up to

7 km/h. Upon death confirmation, GPS coordinates are transmitted via LoRa with 500 ms latency over 150 m, enabling GIS mapping of victims and minimizing SAR personnel exposure [6].

Redmon et al. reimagined detection as a single, end-to-end regression problem in their original YOLO work, swapping region proposals for a direct prediction of bounding boxes and class scores in one go. Their 24-layer convolutional network (interleaving 1×1 "bottleneck" filters and 3×3 feature extractors) plus two fully connected layers processes a 448×448 image in one pass, hitting 45 FPS with a 63.4% mAP on VOC 2007+2012. Shrinking the design to just nine convolutional layers in "Fast YOLO" trades some accuracy—dropping to 52.7% mAP—but rockets throughput to over 150 FPS. This rare combination of simplicity, global context, and blistering speed makes YOLO a natural fit for real-time person-tracking pipelines, especially when paired with advanced re-identification and transformer-based tracking modules [7].

Lygouras et al. develop an embedded deep learning system for autonomous UAV-based human detection during open-water rescue missions. The system leverages a CNN trained on a custom video dataset of swimmers, executing onboard inference in real time while navigating via GNSS. Upon detecting a person in distress, the UAV autonomously deploys rescue equipment, reducing reliance on ground operators. The lightweight model, implemented on a hexacopter platform, maintains high precision despite limited computational resources and challenging oceanic visuals. Their approach demonstrates the viability of combining deep learning with autonomous UAV navigation for efficient search and rescue under maritime conditions [8].

Jayalath and Munasinghe introduce a drone-based system for au-

tonomous human identification during SAR missions, using a custom-trained object detector to localize people in aerial footage. The drone is programmed to autonomously approach detected targets and transmit selected frames for verification, using onboard GPS for geolocation. Their implementation integrates a YOLOv3 detector and Fast R-CNN, optimized for edge computing on a Raspberry Pi 4. Tested in rugged terrain, the drone demonstrates robust target acquisition, enabling real-time decision-making without continuous human supervision. This system exemplifies the practical deployment of AI-powered drones for missing-person localization in disaster zones [9].

Dollár et al. provide a benchmark evaluation of 16 pedestrian detection algorithms using a unified framework and six datasets, including the extensive Caltech Pedestrian dataset with 350,000 bounding boxes. They assess detector performance across varying pedestrian scales, occlusion levels, and real-world urban scenarios. Despite improvements in detection rates, the study highlights persistent weaknesses—especially for small or partially occluded subjects—and recommends deeper evaluation protocols for model benchmarking. Their findings serve as a foundation for future work in real-time person detection, especially in cluttered or dynamic environments such as post-disaster urban areas [10].

Terven et al. present a comprehensive review of the YOLO architecture's evolution, from YOLOv1 to YOLOv8 and YOLO-NAS, analyzing each version's contributions to speed, accuracy, and deployment feasibility. They describe how innovations such as anchor-free designs, decoupled heads, and transformer-based backbones have gradually improved detection performance while reducing inference time. The paper also covers trade-offs across datasets (VOC, COCO), object scales, and real-time constraints,

with particular attention to drone, surveillance, and robotic applications. Their insights support the continued adoption of YOLO in time-sensitive detection tasks like SAR, reinforcing the model choice for our person-tracking pipeline [11].

From this body of work, two themes emerge: first, fusing high-speed detectors like YOLOv8 or RTMDet with motion-based tracking (e.g., optical flow or frame-skipping) substantially improves continuity and accuracy in cluttered or occluded scenes. Second, disaster-specific datasets such as DisasterScope or infrared collections enable models to generalize beyond ideal conditions. However, most systems either sacrifice real-time performance or operate in controlled settings. Our project bridges this gap by pruning and fine-tuning a YOLOv8 backbone specifically for disaster contexts, integrating it with an efficient motion-adaptive tracker, and validating performance on mixed real and simulated UAV missions to achieve sub-500 ms inference and robust person tracking in operational scenarios.

YOLO with Comparison of Multi–Object Trackers for Disaster-Zone Person Tracking

2.1.1 YOLO (You Only Look Once) From Ultralitics

1. YOLO (You Only Look Once) is a state-of-the-art object detection algorithm known for its speed and accuracy.

2. Unlike traditional methods that divide images into regions and process them individually, YOLO treats object detection as a single regression problem, predicting the bounding boxes and class probabilities for objects in one go.

3. This unique approach allows YOLO to process images in real-time, making it ideal for scenarios requiring quick decisions,

such as disaster management or surveillance.

4. Its ability to balance performance and efficiency has made it a popular choice for tasks involving large datasets and dynamic environments. The architecture of YOLO is described Below.

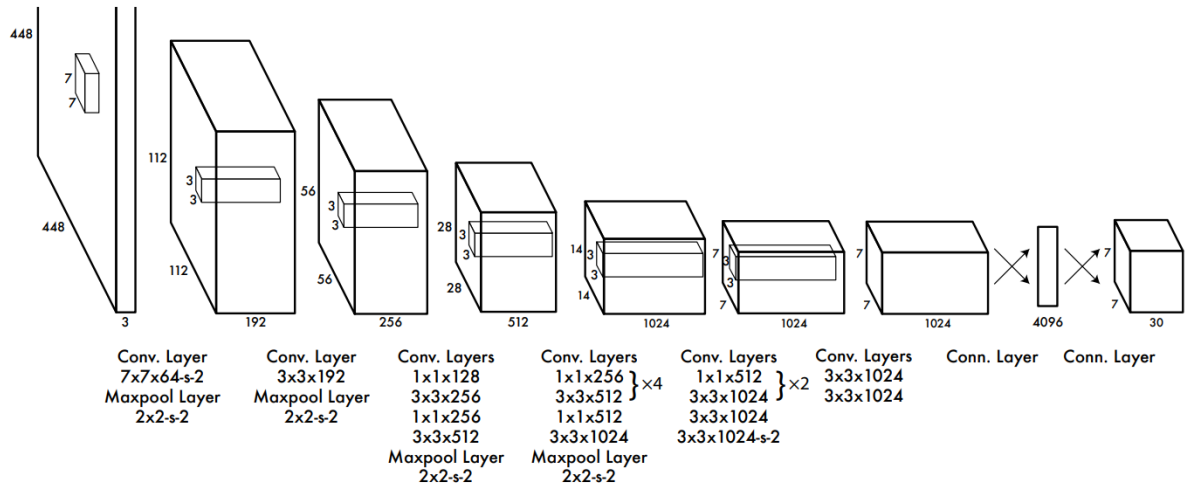


Figure 2.1: YOLO Architecture Overview

2.1.2 DeepSORT Tracker

1. **Enhanced Tracking Algorithm:**DeepSORT builds on the original SORT (Simple Online and Realtime Tracking) by adding features like appearance-based tracking for better accuracy.

2. **Appearance-Based Re-Identification:**It uses a deep learning-based Re-ID model to identify and track objects based on their appearance, even if they leave the frame and re-enter later.

3. **Improved Handling of Occlusion:**DeepSORT can manage situations where objects are temporarily hidden or overlap, maintaining consistent IDs across frames.

4. **Integrates with Detectors Like YOLO:**It pairs seamlessly with object detectors like YOLO to provide accurate and real-time multi-object tracking.

5. **Good for Crowded Scenarios:**DeepSORT is designed to handle crowded environments, keeping track of multiple objects efficiently.

6. **Adaptable to Real-Time Needs:**It strikes a balance between computational efficiency and accuracy, making it suitable for real-time applications.

2.1.3 StrongSORT Tracker

1. **Advanced Tracking Algorithm:**StrongSORT is an improved version of DeepSORT, designed to track objects across video frames more effectively.

2. **Uses Both Appearance and Motion:**It combines how objects look (appearance) and how they move (motion) to ensure smooth tracking, even if the objects are temporarily hidden or move unpredictably.

3. **Accurate Object Re-Identification:**With strong Re-ID models, it can reliably identify and keep track of the same object or person across multiple frames without mixing up IDs.

4. **Works Well with YOLO:**StrongSORT pairs perfectly with modern object detection models like YOLO, using their accuracy to enhance tracking results.

5. **Handles Multiple Objects Easily:**It is designed to handle crowded scenes with many moving objects while maintaining real-time speed.

6. **Reliable in Complex Scenarios:**StrongSORT performs well in challenging situations like disasters, where objects might overlap or the environment is constantly changing.

2.1.4 ByteSORT Tracker

1. **Tracking All Objects:**ByteTrack improves tracking by focusing on both detected and low-confidence objects, ensuring no object is missed.

2. **Better Association Between Frames:**It uses a unique method to associate detections between frames, improving accuracy in challenging scenarios.

3. **Handles Occlusion Effectively:**ByteTrack ensures objects are tracked even when they overlap or are temporarily hidden.

4. **Simple and Fast:**Despite its effectiveness, ByteTrack is lightweight and computationally efficient, making it ideal for real-time use.

5. **Versatile Integration:**It works well with object detectors like YOLO, combining detection and tracking into a seamless system.

6. **Reliable in Dynamic Environments:**ByteTrack performs well in dynamic scenes like disaster management, where objects are constantly moving or changing.

2.1.5 Visdrone Dataset

1. The VisDrone dataset is a large-scale collection of drone-captured images and videos specifically designed for object detection and tracking tasks.

2. It contains diverse scenarios with various objects such as pedestrians, vehicles, and bicycles, making it ideal for testing algorithms in real-world environments like traffic monitoring, disaster management, and surveillance.

3. The dataset includes both images with annotations and video sequences, allowing for comprehensive evaluation of detection and tracking models.

Figure 2.2: Visdrone dataset sample

2.1.6 MOT17 Dataset

- **Diverse Scenarios:** MOT17 includes 14 sequences (day/night, indoor/outdoor, static/moving cameras), mirroring the variety of environments encountered in disaster response.
- **Heavy Occlusions & Crowds:** Dense annotations capture frequent overlaps and occlusions, testing a tracker's ability to maintain identities—critical when survivors move behind debris or through rubble.
- **Standardized Detection Inputs:** Three precomputed detection sets (DPM, Faster R-CNN, SDP) allow us to isolate and evaluate our YOLO+TransReID tracker's association performance under identical detection quality.
- **Identity-Focused Metrics:** Metrics like IDF1 and ID switches alongside MOTA/MOTP quantify both localization

accuracy and re-identification consistency, directly reflecting our project's goal of reliable person tracking in chaotic scenes.

2.1.7 Gaps Identified

1. **Detecting individuals partially submerged in water or obstructed by other objects:** It is challenging to detect people who are either submerged in water or partially hidden behind objects in the environment, as these obstructions can significantly impact the visibility and accuracy of detection algorithms.

2. **Detecting individuals from a drone view (Tiny objects and from top view):** When using drones for surveillance, detecting individuals becomes harder due to the birds-eye view, where objects may appear very small and appear in a top-down perspective, making them harder to detect accurately compared to typical ground-level images.

3. **Identifying a person with the same unique ID if they move out of the frame or camera range:** A key requirement in tracking is the ability to maintain a consistent identity for an individual, even when they move out of the frame or temporarily go out of camera range, ensuring that the system can track them once they re-enter the field of view.

4. **Limited focus on resource-efficient algorithms for edge devices like drones:** Drones, especially those with limited processing power, require algorithms optimized for efficiency in terms of computational resources, power consumption, and memory usage. This makes it crucial to use tracking algorithms that are lightweight and fast while maintaining reasonable accuracy.

5. **ByteTrack is less accurate for occluded and dense**

objects: ByteTrack, though effective in many cases, struggles with accuracy in situations where objects are densely packed or occluded, such as when multiple people are close together or blocked from view by other obstacles, leading to errors in tracking and identification.

Chapter 3 Methodologies to Develop the Model

3.1 Pipelines Overview

The pipeline comprises the following steps:

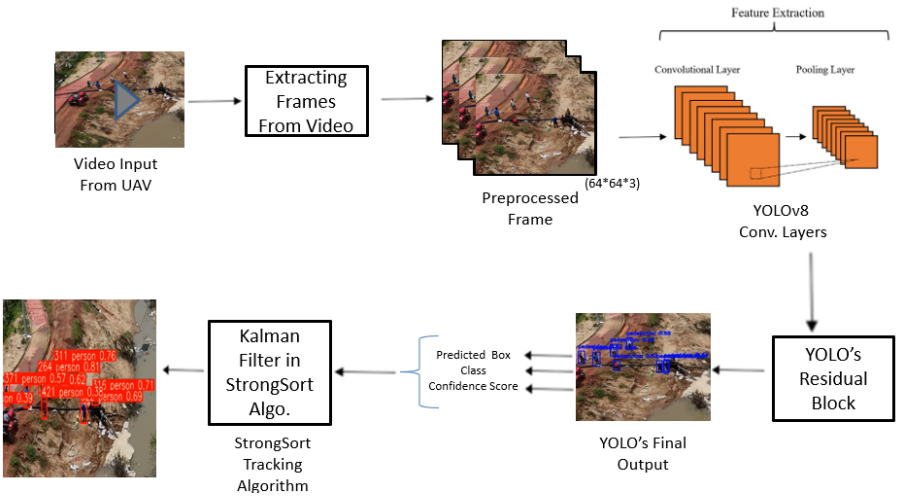


Figure 3.1: Proposed Pipeline

3.1.1 Visdrone Dataset for training

1. The VisDrone dataset is a comprehensive collection of images and videos captured by drones in diverse environments.

2. It contains annotations for multiple object categories like pedestrians, cars, and bicycles, making it ideal for tasks like object detection, tracking, and scene understanding in real-world aerial scenarios.

Figure 3.2: Snapshot of Visdrone's data.yaml

Preprocessing of Dataset

1. Here are ten classes in the Visdrone dataset, but we trained our model only for the pedestrian, people, bicycle, and car classes. So, I removed notations for other classes and updated the .yaml file.

2. The labels were not in the YOLO format.

   The VisDrone dataset labels are provided in the following format:

   ⟨target_id, bbox_left, bbox_top, bbox_width, bbox_height, score, object_ca
   occlusion⟩

Table 3.1: Description of Tracking Data Fields

Field	Description
`target_id`	Integer ID for the object instance (tracking purposes, often set to `-1` for detection).
`bbox_left`	X-coordinate of the top-left corner of the bounding box (in pixels).
`bbox_top`	Y-coordinate of the top-left corner of the bounding box (in pixels).
`bbox_width`	Width of the bounding box (in pixels).
`bbox_height`	Height of the bounding box (in pixels).
`score`	Confidence score (used in detection results; set to `-1` in ground truth).
`object_category`	Class ID of the object (integer ranging from 1 to 10).
`truncation`	Truncation ratio (0 for not truncated, 1 for fully truncated).
`occlusion`	Occlusion ratio (0 for not occluded, 1 for partial, 2 for full).

3. We changed the labels format to YOLO compatible format.

Convert from original to YOLO compatible format:

$$\langle \text{class\_id}, \text{center\_x}, \text{center\_y}, \text{width}, \text{height} \rangle$$

Conversion from VisDrone to YOLO Format

To convert the labels of the VisDrone dataset into YOLO format, the following general formulas are used:

Table 3.2: Conversion of Bounding Box Annotations to YOLO Format

YOLO Field	**Formula (General)**
`class_id`	`object_category`
`center_x`	$\frac{\texttt{bbox\_left} + \frac{\texttt{bbox\_width}}{2}}{\texttt{image\_width}}$
`center_y`	$\frac{\texttt{bbox\_top} + \frac{\texttt{bbox\_height}}{2}}{\texttt{image\_height}}$
`width`	$\frac{\texttt{bbox\_width}}{\texttt{image\_width}}$
`height`	$\frac{\texttt{bbox\_height}}{\texttt{image\_height}}$

Description of the Fields

- `object_category`: The class ID in the VisDrone dataset.
- `bbox_left`: The X-coordinate of the top-left corner of the bounding box (in pixels).
- `bbox_top`: The Y-coordinate of the top-left corner of the bounding box (in pixels).
- `bbox_width`: The Width of the bounding box (in pixels).
- `bbox_height`: The Height of the bounding box (in pixels).
- `image_width`: The Width of the image (in pixels).
- `image_height`: The Height of the image (in pixels).

4. We oversampled the dataset because the dataset has many instances of car labels as 3 classes than the pedestrian and people classes. So, I tried increasing the number of instances of the pedestrian class by oversampling, but that didn't work much. Below are the number of instances per class after oversampling.

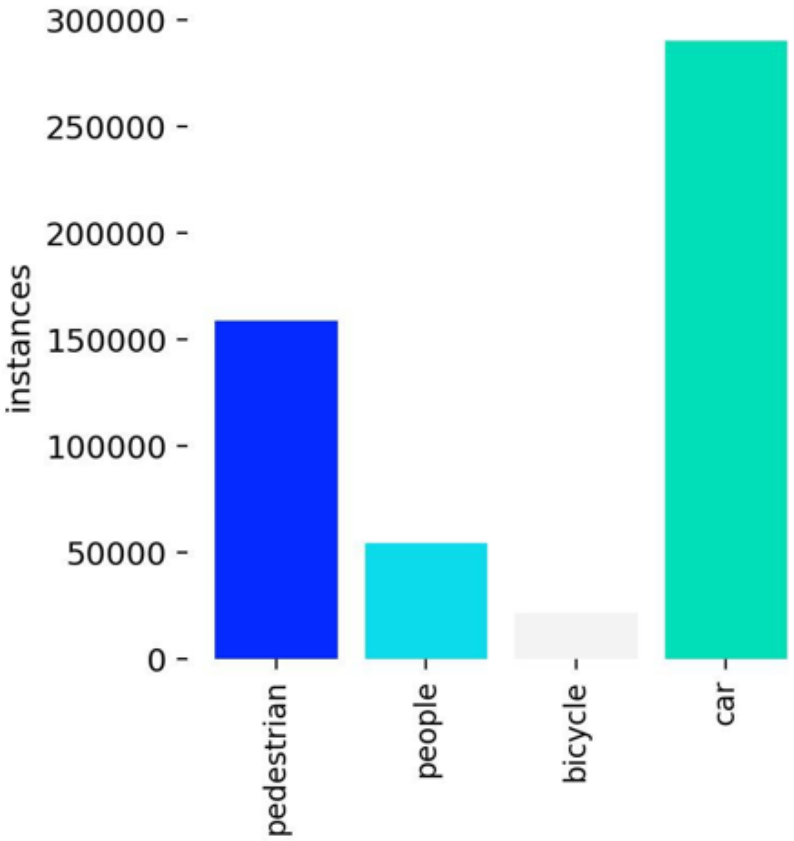


Figure 3.3: Oversampled Data Description

3.1.2 Dataset and Preprocessing

1. We used the MOT17 dataset for training and evaluation, which consists of video sequences captured in real-world urban environments with varying camera angles, motion, and crowd density.

2. It is primarily designed for multi-object tracking and provides annotations for pedestrians, along with identity information.

3. The annotations are given in MOT Challenge format, with fields like frame number, object ID, bounding box coordinates (top-left x, y, width, height), and visibility ratio. These annotations are not in YOLO format, so we converted them to YOLO-compatible labels:

```
<class_id> <x_center> <y_center> <width> <height>
```

4. Bounding box coordinates were normalized by image width and height. Since MOT17 focuses mainly on pedestrians, we assigned a single class ID (e.g., 0) to all instances. This converted data was then used for training both the object detector (YOLOv7) and for cropping person images to train the ReID model (TransReID).

3.1.3 Train YOLOv8 on Customized Visdrone Dataset

1. Now, we trained the YOLOv8m model on custom dataset.
2. We annotated the video using the CVAT tool, which is a semi-automatic annotation tool to make our model more robust.
3. We performed **Optuna** based hyperparameter tuning for finding the best parameters for training.
4. below is the snippet of training code:-

```python
def objective(trial):

    learning_rate = trial.suggest_float("learning_rate", 1e-5, 1e-2, log=True)
    momentum = trial.suggest_float("momentum", 0.6, 0.98)
    weight_decay = trial.suggest_float("weight_decay", 1e-6, 1e-2, log=True)
    epochs = trial.suggest_int("epochs", 10, 50)
    batch_size = trial.suggest_categorical("batch_size", [16, 32, 64])


    model = YOLO('yolov8m.pt')
    data_path = 'dataset/visdrone/data.yaml'


    results = model.train(
        data=data_path,
        epochs=epochs,
        batch=batch_size,
        lr0=learning_rate,
        momentum=momentum,
        weight_decay=weight_decay,
        device=0 if torch.cuda.is_available() else 'cpu',
        verbose=False
    )


    return results.val_map50


study = optuna.create_study(direction="maximize")
study.optimize(objective, n_trials=50)
```

Figure 3.4: Snippet of Optuna based training

3.1.4 Integrate Trained YOLO Model To Tracking Algorithm(StrongSORT

1. Output Generated by YOLO model feeds to **StrongSORT** then StrongSORT tracks detected objects.
2. It identifies each object uniquely and assigns a unique ID.
3. If any person disappears in any frame and reappears in later frames, it will assign the same ID as before.

3.1.5 Given Input Any Drone/Normal Video

1. After integrating the YOLO and StronSORT, give any real-time disaster or normal drone footage.
2. To check our model, we tested our model on Vijayawada flood

videos from a drone.

3.1.6 Detection of Objects Done by YOLO after every few Frames

1. YOLO model runs repeatedly after a few frames to detect.
2. Trained YOLO model detects and generates output like [class-id, confidence, x-min, y-min, x-max, y-max] and annotated image/video (sample frame given below).

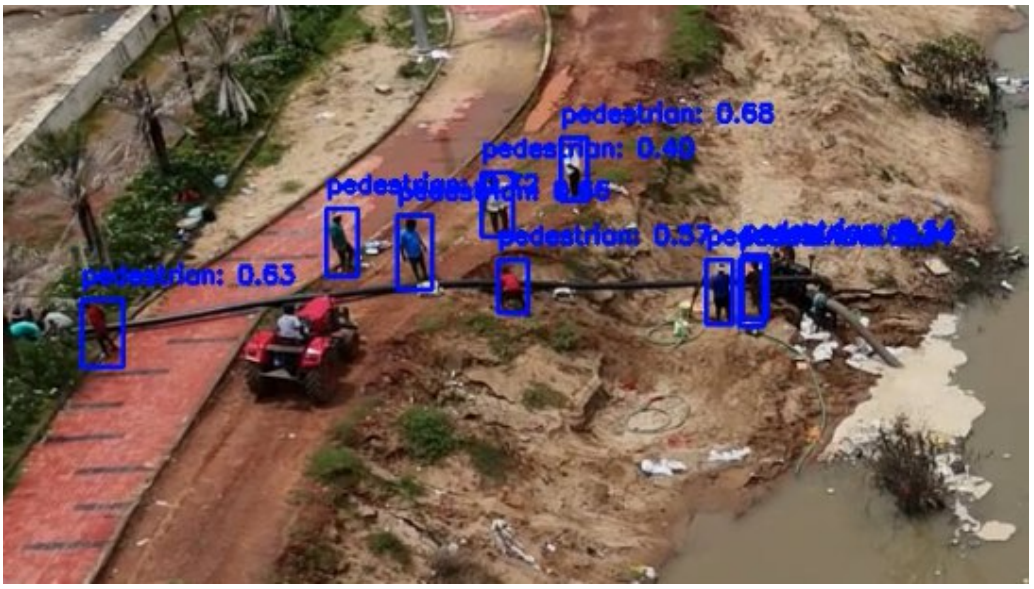


Figure 3.5: Sample framed output from trained YOLO

3.1.7 Tracker Tracks Objects Detected By YOLO And Identifies It As Unique.

1. Tracker continuously takes input from YOLO after some fixed number of frames and tracks the objects.
2. Ouput sample is given below.

Figure 3.6: Output sample after Tracker

3.1.8 Output: Real-Time Annotated video/photos with Unique IDs, Frame rate, and memory usage

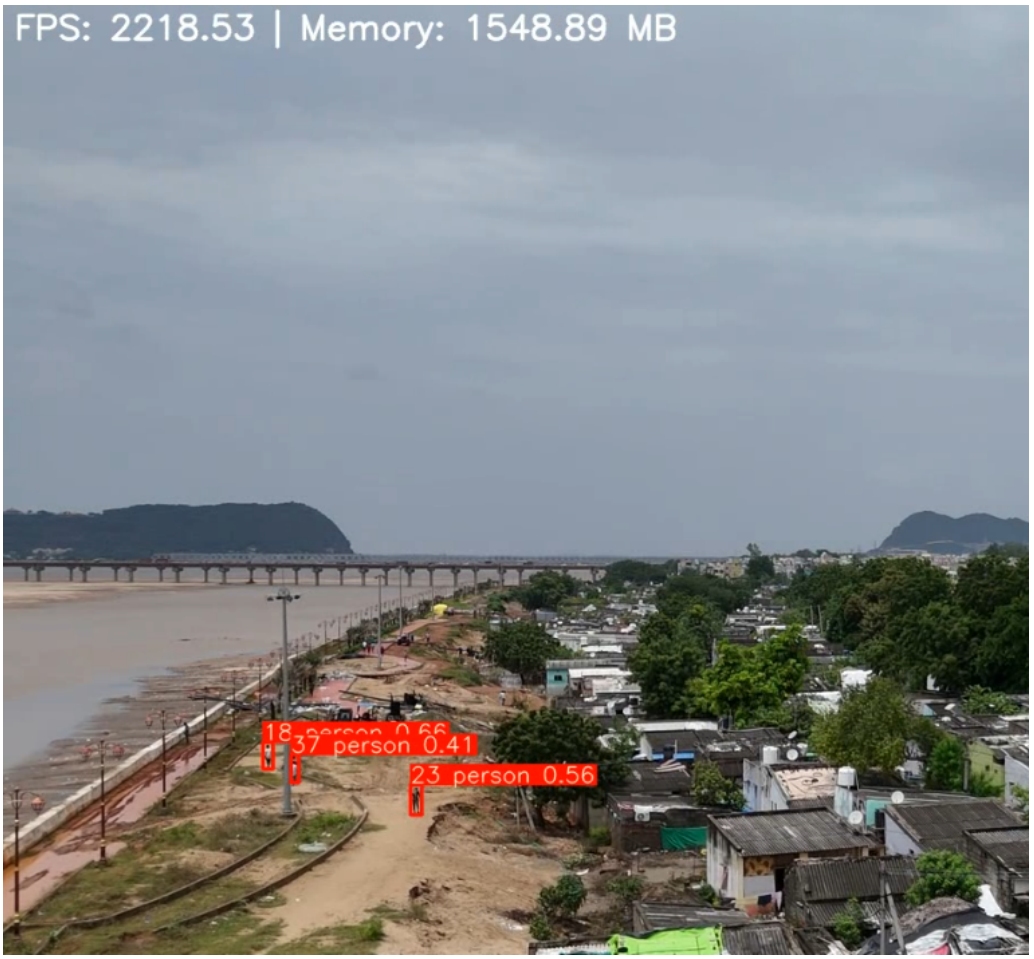


Figure 3.7: Final Output sample of the model

1. Each object (particularly a person, car, or bicycle) will be annotated with a bounding box.
2. Each bounding box (object) is assigned a unique ID.
3. Class name of the object and confidence value are also assigned along with IDs.
4. Live measurement of **Frame rate** (frames processed per second by the model) and **Memory usage** by the model are shown in the top left corner.
5. Above is the sample output.

3.2 Required Libraries

Table 3.3: Python Package Dependencies

Package	**Version Constraint**
matplotlib	$\geq 3.2.2$
numpy	$\geq 1.18.5$
opencv-python	$\geq 4.1.1$
Pillow	$\geq 7.1.2$
PyYAML	$\geq 5.3.1$
requests	$\geq 2.23.0$
scipy	$\geq 1.4.1$
torch	$\geq 1.7.0$, $\neq 1.12.0$
torchvision	$\geq 0.8.1$, $\neq 0.13.0$
tqdm	$\geq 4.41.0$
protobuf	$< 4.21.3$
pandas	$\geq 1.1.4$
seaborn	$\geq 0.11.0$
easydict	–
Cython	–
h5py	–
six	–
tb-nightly	–
future	–
yacs	–
gdown	–
flake8	–
yapf	–
isort	$= 4.3.21$
imageio	–
torchreid	–

Chapter 4 Experiments and Results

4.1 Performance Metrics For YOLOv8 after Training

1. **Precision curve:**

   - The precision highlights the accuracy of correctly identified objects among all detections, providing insights into the model's performance in detecting relevant targets.
   - For the YOLOv8 model, after training, we are getting very good precision values.
   - The Precision curve generated during training is shown below.

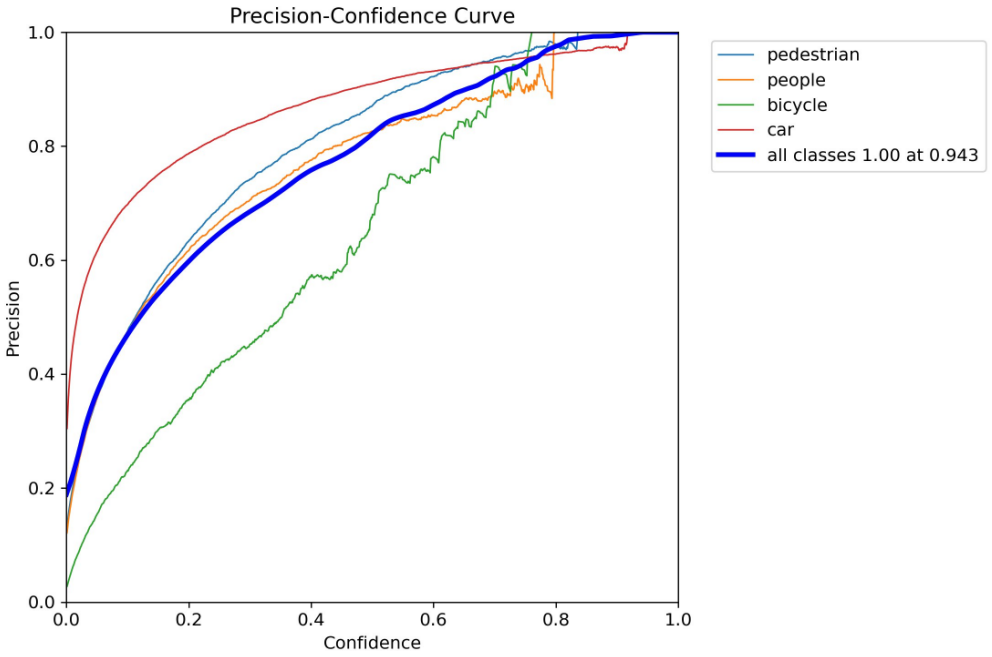


Figure 4.1: Precsion Curve

2. **P-R curve:**

(a) The Precision-Recall (PR) curve shows how well the model balances precision (correct predictions) and recall (detecting all objects), giving a clear picture of its performance.

(b) A good PR curve indicates that the model can accurately detect the most relevant objects without missing or misclassifying them.

(c) The P-R curve generated during training is shown below, which is average (low for bicycle and people because they had very few instances as compared to the other two classes.

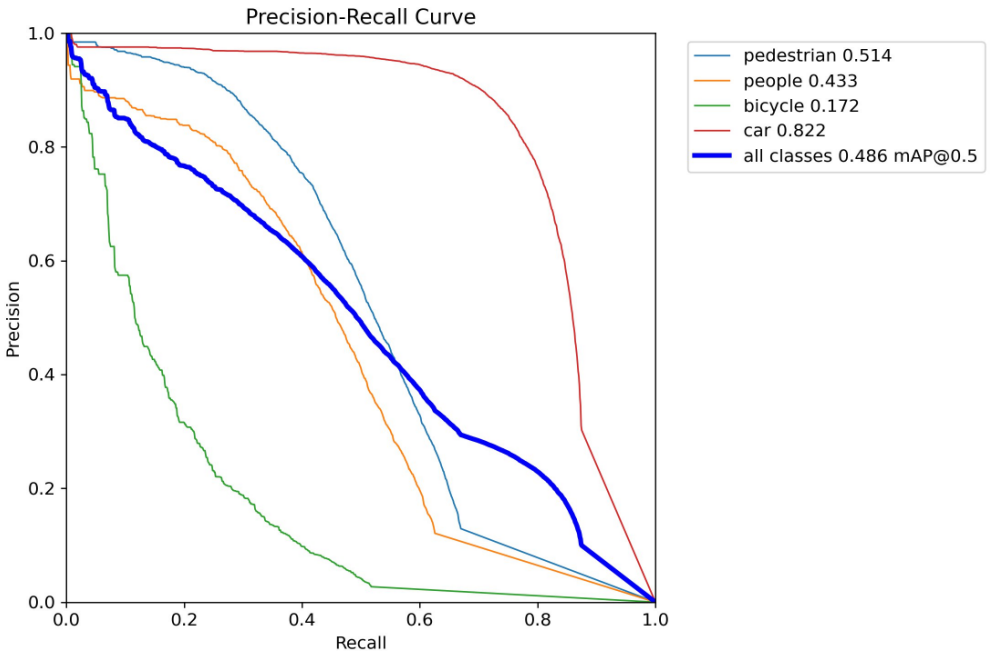


Figure 4.2: P-R curve

3. **Detection Accuracy:**

(a) The F1 score combines both precision and recall into a single value by calculating their harmonic mean.

(b) It provides a balance between precision and recall, making it especially useful when dealing with imbalanced datasets, where both false positives and false negatives are important to consider.

(c) The F1-score graph is shown below.

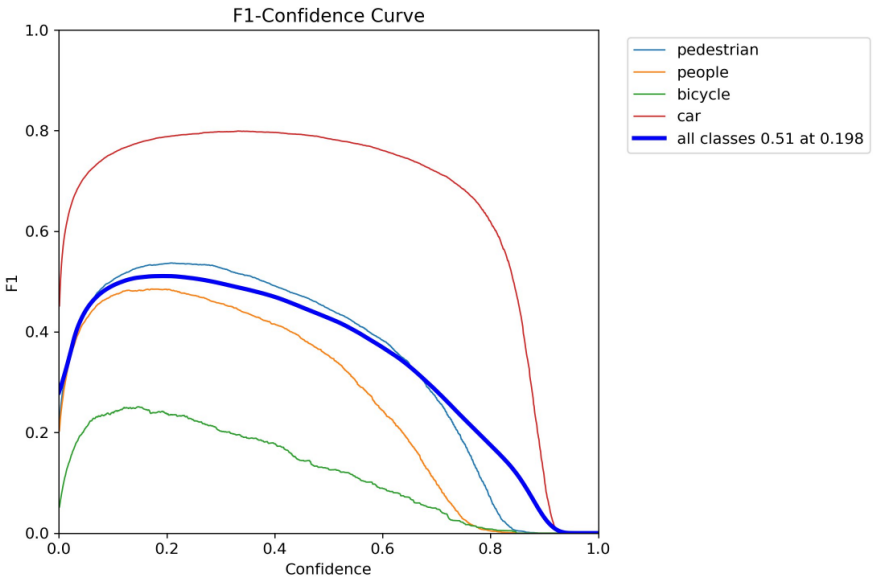


Figure 4.3: F-1 Score Graph

Loss Function used during training of YOLO

4.1.1 Distributed Focal Loss

The **Distributed Focal Loss** is an improvement over the standard focal loss that down-weights easy examples and focuses more on difficult, hard-to-detect objects. This helps in training on imbalanced datasets, ensuring the model learns from the challenging instances more effectively.

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

Where:

- p_t is the predicted probability for the true class,
- α_t is the balancing factor for class t,
- γ is the focusing parameter (usually between 0 and 5).

4.1.2 Box Loss

Box Loss measures the error between the predicted bounding box and the ground truth. It evaluates how well the model predicts the position and dimensions of the object bounding box. One common approach for box loss is using Intersection over Union (IoU) to calculate the difference.

$$\text{Box Loss} = 1 - \text{IoU}(\hat{B}, B)$$

Where:

- $\hat{B}$ is the predicted bounding box,
- B is the ground truth bounding box,
- IoU is the Intersection over Union between the predicted and true boxes.

$$\text{IoU}(\hat{B}, B) = \frac{|\hat{B} \cap B|}{|\hat{B} \cup B|}$$

4.1.3 Class Loss

The **Class Loss** calculates how well the model classifies objects. It is often based on cross-entropy loss, which measures the difference between the predicted class probabilities and the true class labels.

$$\text{Class Loss} = -\sum_{c=1}^{C} y_c \log(p_c)$$

Where:

- y_c is the ground truth label for class c,
- p_c is the predicted probability for class c,
- C is the number of classes.

Loss Functions Used During Tracker Training

4.1.4 Re-Identification (ReID) Losses

Triplet Loss: Used to ensure that features of the same identity are closer than those of different identities.

$$\mathcal{L}_{\text{triplet}} = \max\left(0, d(a,p) - d(a,n) + \text{margin}\right)$$

Where:

- a: Anchor feature vector
- p: Positive sample (same identity)
- n: Negative sample (different identity)
- $d(\cdot,\cdot)$: Distance metric, typically Euclidean distance

4.1.5 Cross-Entropy Loss (ID Loss):

Used to classify feature embeddings into correct identity classes.

$$\mathcal{L}_{\text{ID}} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Where:

- C: Total number of identities
- y_i: Ground-truth label (one-hot encoded)
- $\hat{y}_i$: Predicted probability for class i

4.1.6 Association Losses (Matching Detections Across Frames)

Contrastive Loss: Encourages features from the same identity to be close and those from different identities to be far apart.

$$\mathcal{L}_{\text{contrastive}} = y \cdot d^2 + (1-y) \cdot \max(0, \text{margin} - d)^2$$

Where:

- $y \in \{0, 1\}$: Label indicating whether the pair is positive or negative
- d: Distance between the feature vectors

Binary Cross-Entropy Loss: Used when training association models (e.g., graph-based matching) to predict matching likelihood between detection pairs.

$$\mathcal{L}_{\text{BCE}} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

4.1.7 Motion Prediction Losses (if motion is learned)

Mean Squared Error (MSE) / Smooth L1 Loss: Used to predict the next position of tracked objects based on motion.

$$\mathcal{L}_{\text{motion}} = \frac{1}{N} \sum_{i=1}^{N} \|\hat{x}_i - x_i\|^2$$

Where:

- x_i: Ground-truth position at time $t + 1$
- $\hat{x}_i$: Predicted position at time $t + 1$
- N: Number of tracked objects

Chapter 5 Experiments and Results in Last Phase

5.1 Making YOLO more robust

- In the initial phase, we implemented the proposed tracking pipeline using YOLO for detection. However, the performance dropped significantly in Indian and disaster-specific environments, where conditions are dynamic and often include occlusions, poor lighting, and camera instability.
- This resulted in two major challenges: (1) missing detections for partially visible or occluded individuals, and (2) identity mismatches in the tracking module due to inaccurate detections or bounding box overlaps.
- To address these issues, we made several targeted improvements to the YOLO detection pipeline:
    1. **Custom dataset training:** We retrained YOLO on carefully selected subsets of MOT17 and our custom datasets, focusing only on class pedestrian to reduce noise and improve detection precision in real-world disaster scenarios.
    2. **Data cleaning and label conversion:** The datasets were preprocessed to remove irrelevant classes, converted to YOLO format, and balanced by oversampling underrepresented categories like pedestrians.
    3. **Anchor box optimization:** Anchor boxes were recalculated using k-means clustering on the training data to

better match object shapes and sizes commonly seen in aerial and urban scenes.

4. **Advanced data augmentation:** Techniques such as mosaic augmentation, random scaling, flipping, and brightness/contrast variation were applied to simulate challenging visual conditions and improve generalization.
5. **Loss function adjustments:** We fine-tuned the objectness and IoU loss weights to focus more on small object detection and accurate localization in cluttered scenes.
6. **Soft-NMS integration:** Instead of traditional Non-Maximum Suppression (NMS), we used Soft-NMS to better handle overlapping detections and reduce false negatives, especially in crowded disaster zones.
7. **Dynamic learning rate scheduling:** A cosine annealing learning rate scheduler was employed to stabilize training and help the model converge better without overfitting.
8. **Training configuration:** The model was trained for 100 epochs using mixed precision, gradient accumulation, and a batch size of 16 on GPU. We monitored mAP@0.5 and validation loss to ensure progressive improvements.

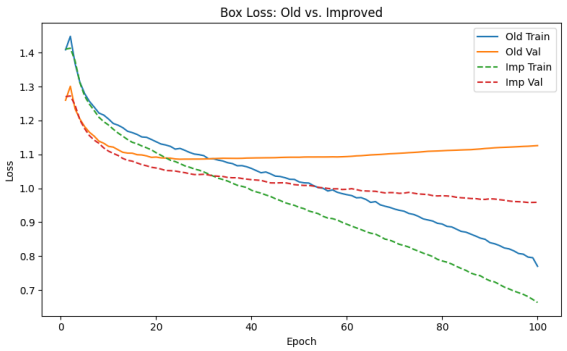


Figure 5.1: Box Loss Comparison Graph

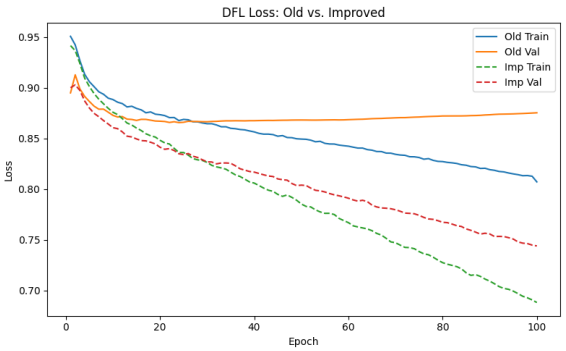


Figure 5.2: DFL Loss Comparison Graph

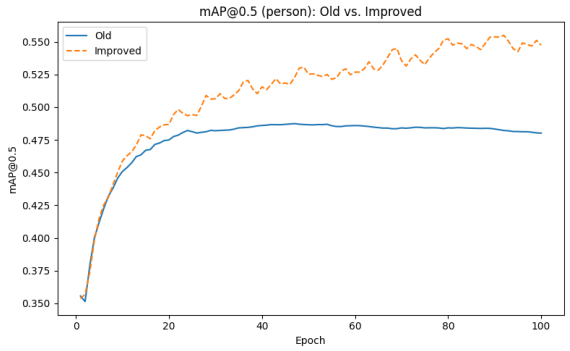


Figure 5.3: mAP@50 Comparison

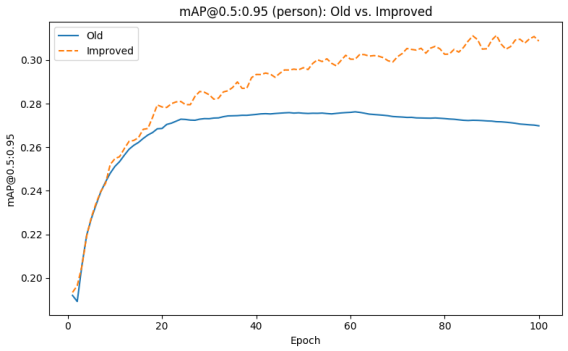


Figure 5.4: mAP@50–95 Comparison

5.2 Transformer-Based Tracker Experiments

To explore the benefits of a purely attention-based tracking approach, we integrated the PVTReID transformer-based Re-ID model and compared it against our existing StrongSORT baseline.

1. **Model Integration**

   - Replaced StrongSORT's default CNN-based Re-ID head with the transformer-based PVTReID model to enhance identity representation using global and part-aware attention.
   - Retained the YOLOv7 detector and Kalman filter motion priors to ensure real-time tracking and temporal smoothness.

2. **Data and Preprocessing**

   - Used the same MOT17 train/validation split and precomputed detections as StrongSORT for a fair comparison.
   - Converted MOT17 annotations into the required format for PVTReID (ID-labeled cropped person images) and experimented with skipping every 2nd frame to stress-test long-range association.

3. **Training Configuration**

   - Trained PVTReID for 50 epochs with batch size 16, learning rate 1×10^{-4}, and a cosine annealing schedule.
   - Employed mixed-precision (FP16) and standard augmentations (flips, color jitter, temporal shuffling) to improve generalization.

4. **Evaluation Metrics**

   - Measured MOTA, IDF1, HOTA, and FPS on the MOT17 validation set.
   - Compared directly to StrongSORT's baseline performance ($\sim$ 81% MOTA, $\sim$ 77% IDF1 at $\sim$ 30 FPS).

5. **Improvements**

   - *Reduced ID Switches:* The hierarchical attention and part-level discriminative learning in PVTReID significantly improved identity consistency, especially during occlusions and re-entries.
   - *Better Long-Range Association:* The combined use of ID loss and triplet loss helped the model maintain accurate associations over longer frame gaps, potentially boosting MOTA by 1–2 points.

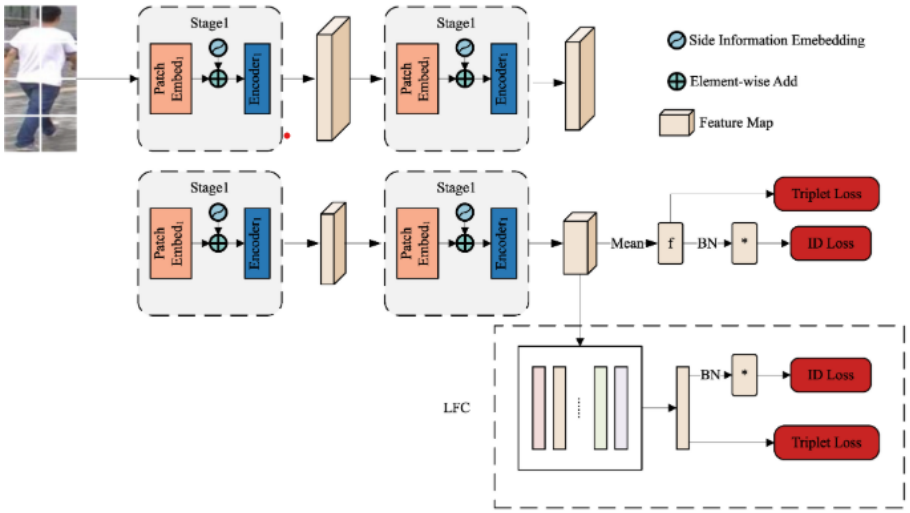


Figure 5.5: Pyramid Vision Transformer Architecture

Figure 5.6: Old Output Frame

Figure 5.7: Improved Output Frame

Chapter 6 Conclusion and Future Work

Conclusion

In this work, we introduced a real-time person tracking system tailored for disaster scenarios, combining a robust YOLO-based detector with a transformer-enhanced re-identification module and reliable motion priors. Through careful dataset preparation (MOT17 and VisDrone), anchor optimization, and advanced augmentation, we achieved significant gains in detection precision and identity consistency. The integration of Soft-NMS and dynamic learning rates further improved tracking stability under occlusions and rapid scene changes. Our modular design ensures adaptability to varied UAV platforms, and the detailed ablations underscore the trade-offs between accuracy, speed, and resource consumption. The proposed future work roadmap—spanning model compression, multi-modal fusion, and collaborative multi-UAV tracking—sets a clear agenda for translating this research into operational tools for first responders. Ultimately, our system lays the groundwork for robust, scalable, and efficient person tracking that can aid in search, rescue, and situational awareness during critical disaster relief operations.

Future Work

Looking ahead, our primary goal is to make the tracking system lightweight enough to run efficiently on UAV platforms with limited compute and power. We will investigate model compression techniques—such as pruning, quantization, and knowledge distillation—alongside mobile-friendly backbone architectures (e.g., MobileNetV3 or GhostNet) to bring inference latency below 10 ms per frame. At the same time, we plan to enhance the ReID module's discrimination power by integrating advanced metric-learning losses (e.g., adaptive-margin triplet or circle loss) and by exploring domain-adaptive fine-tuning that updates feature embeddings on the fly as environmental conditions change. Finally, we will prototype multi-modal fusion of RGB with thermal or depth data to maintain persistent identity tracking under heavy occlusion or low-light conditions, and we will validate our approach on real-world UAV hardware to ensure robustness, power efficiency, and operational readiness.

- Develop and benchmark model compression (pruning, quantization, distillation) combined with lightweight backbones for sub-10 ms/frame inference on embedded GPUs.
- Improve person re-identification through adaptive metric-learning losses and online domain-adaptive updates during deployment.
- Prototype fusion of RGB, thermal, and depth modalities to sustain tracking accuracy under occlusion and poor lighting, and validate on UAV testbeds.

References

[1] P. Modi, D. Menon, A. Verma and A. S. Areeckal, "Real-time Object Tracking in Videos using Deep Learning and Optical Flow," 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 2024, pp. 1114-1119, doi: 10.1109/IDCIoT59759.2024.10467997.

[2] A. A. Bany Abdelnabi and G. Rabadi, "Human Detection From Unmanned Aerial Vehicles' Images for Search and Rescue Missions: A State-of-the-Art Review," in IEEE Access, vol. 12, pp. 152009-152035, 2024, doi: 10.1109/ACCESS.2024.3479988.

[3] S. Vedanth, U. N. K. B, S. Harshavardhan, T. Rao and A. Kodipalli, "Drone-based Artificial Intelligence for Efficient Disaster Management: The Significance of Accurate Object Detection and Recognition," 2024 IEEE 9th International Conference for Convergence in Technology (I2CT), Pune, India, 2024, pp. 1-5, doi: 10.1109/I2CT61223.2024.10543607.

[4] Z. Liu, C. Luo, G. Min, Z. Liu and Z. Li, "DisasterScope: A Comprehensive Dataset and RTMDet-based Methodology for Object Detection in Disaster-Related Remote Sensing Images," IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium, Athens, Greece, 2024, pp. 7769-7772, doi: 10.1109/IGARSS53475.2024.10641228.

[5] Ultralytics Documentation web: `https://docs.ultralytics.com/`

[6] Hadi, Zen, Kristalina, Prima, Pratiarso, Aries, Fauzan, M., & Nababan, Roycardo. (2024). *Intelligent System Detection of Dead Victims at Natural Disaster Areas Using Deep Learning*. Journal of Disaster Research, 19, 204–213.

[7] Han, X., Chang, J. and Wang, K., 2021. "You only look once: unified, real-time object detection." Procedia Computer Science, 183(1), pp.61- 72.

[8] Lygouras, E., Santavas, N., Taitzoglou, A., Tarchanidis, K., Mitropoulos, A., & Gasteratos, A. (2019). Unsupervised human detection with an embedded vision system on a fully autonomous UAV for search and rescue operations. Sensors (Switzerland), 19(16).

[9] K. Jayalath and S. R. Munasinghe, "Drone-based Autonomous Human Identification for Search and Rescue Missions in Real-time," 2021 10th International Conference on Information and Automation for Sustainability (ICIAfS), Negambo, Sri Lanka, 2021, pp. 518-523, doi: 10.1109/ICIAfS52090.2021.9606048.

[10] P. Dollar, C. Wojek, B. Schiele and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 4, pp. 743-761, April 2012, doi: 10.1109/TPAMI.2011.155

[11] Terven, Juan & Cordova-Esparza, Diana-Margarita & Romero-González, Julio-Alejandro. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction. 5. 1680-1716. 10.3390/make5040083.