RAILWAY SENTRY: DETECTING WORKERS ON RAILWAY TRACKS USING YOLO V11

A MAJOR PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

SWETHA ORUGANTI	**21UK1A6681**
MAIDAM SHIVAKUMAR	**21UK1A66C2**
BODDUPALLY JASHWANTH	**21UK1A66A1**
SRIVARSHA POORVANI	**21UK1A6685**

Under the guidance of

Mrs. P. Shireesha

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

RAILWAY SENTRY: DETECTING WORKERS ON RAILWAY TRACKS USING YOLO V11

A UG PHASE -I PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

SWETHA ORUGANTI	**21UK1A6681**
MAIDAM SHIVAKUMAR	**21UK1A66C2**
BODDUPALLY JASHWANTH	**21UK1A66A1**
SRIVARSHA POORVANI	**21UK1A6685**

Under the guidance of

Mrs. P. Shireesha

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

2021-2025

CERTIFICATE OF COMPLETION

UG PROJECT PHASE -I

This is to certify that the UG Project Phase-I entitled "**RAILWAY SENTRY: DETECTING WORKERS ON RAILWAY TRACKS USINGYOLOV11**" is being submitted by **SWETHA ORUGANTI (21UK1A6681)**, **MAIDAM SHIVA KUMAR (21UK1A66C2)**, **BODDUPALLY JASHWANTH(21UK1A66A1), SRIVARSHA POORVANI (21UK1A6685)** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering (AI&ML) to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024- 2025, is a record of work carried out by them under the guidance and supervision.

Project Guide

Mrs.P.Shireesha

(Assistant Professor)

HOD

Dr.Rekha Gangula

(Associate Professor)

External

DECLARATION

We declare that the work reported in the project entitled **"RAILWAY SENTRY:DETECTING WORKERS ON RAILWAY TRACKS USING YOLO V11"** is a record of work done by us in the partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering(AI&ML), VAAGDEVI ENGINEERING COLLEGE (An Autonomous Institution & Affiliated to JNTU Hyderabad) Accredited by NAAC with 'A+' Grade, Certified by ISO 9001:2015 Approved by AICTE, New Delhi, Bollikunta, Warangal- 506005, Telangana, India under the guidance of **Mrs.P.Shireesha**, Assistant Professor, CSE(AI&ML) Department.

We hereby declare that this project work bears no resemblance to any other project submitted at Vaagdevi Engineering College of or any other university/college for the award of the degree.

SWETHA ORUGANTI	**21UK1A6681**
MAIDAM SHIVAKUMAR	**21UK1A66C2**
BODDUPALLY JASHWANTH	**21UK1A66A1**
SRIVARSHA POORVANI	**21UK1A6685**

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Dr. SYED MUSTHAK AHMED, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG PROJECT PHASE -1 in the institute.

We extend our heartfelt thanks to Dr. REKHA GANGULA, Head of the Department of CSE(AI&ML), Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG PROJECT PHASE -1

We express heartfelt thanks to the Coordinator, Mr. T. SANATH KUMAR, Assistant professor, Department of CSE(AI&ML) for her constant support and giving necessary guidance for completion of this UG PROJECT PHASE -I.

We expressheartfelt thanks to the guide Mrs.P. SHIREESHA, Assistant professor, Department of CSE(AI&ML) for her constant support and giving necessary guidance for completion of this UG PROJECT PHASE -1.

Finally, We express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

SWETHA ORUGANTI (21UK1A6681)
MAIDAM SHIVA KUMAR (21UK1A66C2)
BODDUPALLY JASHWANTH (21UK1A66A1)
SRIVARSHA POORVANI (21UK1A6685)

ABSTRACT

Railway Sentry is an innovative safety system designed to detect and monitor workers on railway tracks using the latest YOLO V11 (You Only Look Once version 11) object detection algorithm. The primary objective of Railway Sentry is to enhance railway safety measures and prevent accidents by accurately identifying the presence of workers or unauthorized individuals on railway tracks in real-time. By leveraging the advanced capabilities of YOLO V11, known for its high precision and speed, the system continuously analyzes video feeds from strategically placed cameras along the tracks. Upon detection of any individual, Railway Sentry triggers immediate alerts to notify train operators and railway authorities, enabling timely intervention to prevent potential accidents. Additionally, the system can detect trespassers and wildlife on the tracks, ensuring comprehensive safety coverage. This project aims to combine speed, accuracy, and reliability to create a safer railway environment for workers and passengers alike.

TABLE OF CONTENTS

LIST OF FIGURES PAGENO

1. INTRODUCTION

1.1 OVERVIEW

Objective: Railway Sentry aims to significantly enhance railway safety by utilizing the advanced YOLO V11 algorithm to detect workers and prevent accidents on railway tracks. The system leverages the latest developments in object detection technology to provide real-time monitoring and alerts.

1.2 PURPOSE

The purpose of Railway Sentry is to significantly enhance the safety of railway workers by providing a real-time detection and alert system for individuals present on railway tracks. By leveraging the advanced capabilities of the YOLO V11 object detection algorithm, the system aims to prevent accidents and reduce operational disruptions by ensuring rapid and accurate identification of workers, trespassers, and wildlife on the tracks. This proactive approach not only safeguards human lives but also contributes to the overall efficiency and reliability of railway operations.

The purpose of Railway Sentry is multi-faceted, addressing several critical aspects of railway safety and operational efficiency.

1. **Enhancing Worker Safety:** The primary goal is to ensure the safety of railway workers by detecting their presence on railway tracks in real-time. By providing timely alerts to train operators and railway authorities, Railway Sentry aims to prevent accidents that could result in severe injuries or fatalities.
2. **Preventing Unauthorized Access:** The system is designed to detect not only workers but also unauthorized individuals or trespassers who may pose a risk to themselves and railway operations. Detecting and alerting about trespassers helps in maintaining the security and safety of railway infrastructure.
3. **Mitigating Wildlife Incidents:** Railway Sentry also focuses on detecting wildlife on railway tracks, which can cause accidents and lead to both animal fatalities and train derailments. By identifying and alerting about wildlife presence, the system helps in taking necessary precautions to avoid collisions.

4. **Operational Efficiency:** By providing real-time data on the presence of individuals on the tracks, the system enables railway authorities to make informed decisions quickly. This reduces the downtime and delays caused by accidents and unauthorized track access, thereby improving the overall efficiency of railway operations.

2.PROBLEM STATEMENT

Railway tracks are critical infrastructure components that require constant monitoring to ensure the safety of workers and efficient railway operations. Accidents involving railway workers are a significant concern, leading to severe injuries, fatalities, and substantial operational disruptions. Despite existing safety measures, there are several persistent challenges:

1. **Real-Time Detection:** Current systems may not provide real-time detection of workers, trespassers, or wildlife on the tracks, resulting in delayed responses and increased risk of accidents.
2. **Accuracy and Precision:** Many detection systems lack the accuracy and precision required to identify small or partially obscured objects, leading to false alarms or missed detections.
3. **Environmental Conditions:** Detection systems often struggle to perform effectively under varying environmental conditions such as low lighting, rain, or fog, which can compromise their reliability.
4. **Operational Costs:** Manual inspections and traditional monitoring methods are labor- intensive and costly, leading to increased operational expenses and human errors.
5. **Scalability:** Existing systems may not be easily scalable or adaptable to different types of railway networks and varying track conditions.

Given these challenges, there is a pressing need for an advanced solution that combines speed, accuracy, and reliability to provide continuous, real-time monitoring of railway tracks. The purpose of Railway Sentry is to address these issues by leveraging the latest YOLO V11 object detection algorithm to create a comprehensive safety system that ensures the well-being of railway workers and the smooth operation of railway services.

3.LITERATURE SURVEY

3.1 EXISTING PROBLEM

The existing problems in the realm of railway track safety, particularly concerning the detection of workers on tracks, are significant and multifaceted. These challenges highlight the need for an advanced and reliable solution like the Railway Sentry system. The key issues are as follows:

Lack of Real-Time Monitoring: Current safety measures often fail to provide continuous, real-time monitoring of railway tracks. The absence of such systems increases the risk of accidents, as there is no immediate detection and alert mechanism for workers present on the tracks.

Insufficient Detection Accuracy: Many existing detection systems lack the precision required to accurately identify workers, especially in complex scenarios where individuals may be partially obscured or located in difficult-to-see areas of the track.

Delayed Response Times: Without real-time alerts, the response times to potential hazards are significantly delayed. This can lead to higher chances of accidents and disruptions in railway operations.

Environmental Limitations: Detection systems often struggle with varying environmental conditions such as poor lighting, adverse weather, or obstructions. These limitations reduce the reliability and effectiveness of the systems in ensuring worker safety.

3.2 PROPOSED SOLUTION

Railway Sentry is an advanced safety system designed to detect and monitor workers on railway tracks using the state-of-the-art YOLO V11 (You Only Look Once version 11) object detection algorithm. The primary objective of Railway Sentry is to enhance safety measures and prevent accidents by swiftly identifying the presence of workers or unauthorized individuals in railway track areas.

Scenario 1: Worker Safety Monitoring Railway workers frequently perform maintenance, repairs, or inspections along railway tracks, exposing them to the risk of accidents from oncoming trains.

Railway Sentry utilizes YOLO V11 to continuously monitor the railway track area for the presence of workers. If a worker is detected within the designated safety zone, the system triggers alerts to notify train operators or railway authorities, enabling them to take immediate action to prevent potential accidents and ensure the safety of railway workers.

Scenario 2: Trespasser Detection Trespassing on railway tracks is a common cause of accidents and fatalities. Railway Sentry is equipped with YOLO V11 to detect unauthorized individuals or trespassers entering railway track areas. When unauthorized activity is detected, the system generates alerts and notifies railway security personnel or law enforcement agencies to intervene and prevent potential accidents or incidents of vandalism.

Scenario 3: Wildlife Detection Wildlife crossing railway tracks can pose a safety hazard for both trains and animals. Railway Sentry employs YOLO V11 to detect the presence of wildlife such as deer, bears, or birds on railway tracks. By identifying wildlife in real-time, the system enables train operators to take precautionary measures such as slowing down or stopping trains to avoid collisions and minimize harm to wildlife.

4.THEORITICAL ANALYSIS

4.1. BLOCK DIAGRAM

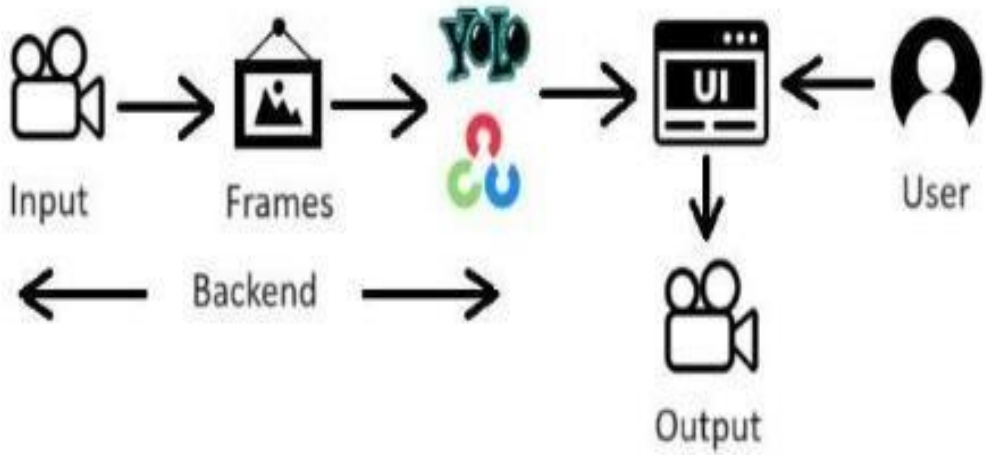


Fig4.1:Block Diagram

System: Railway Sentry - Worker Detection

1. Input:

- Real-time video feed from CCTV cameras installed along railway tracks.
- Images captured from the video stream.

2. Frame Extraction (Backend):

- Video frames are extracted at a suitable rate (e.g., every second) for processing.

3. Object Detection (Backend - YOLO v11):

- YOLO v11 (You Only Look Once, version 11) is employed for object detection.
- YOLO v11 processes each frame to identify potential objects, including humans (workers).
- YOLO v11 provides bounding boxes and confidence scores for detected objects.

4. Worker Classification (Backend):

- A classifier (e.g., a separate neural network or a module within YOLO v11) analyzes the detected objects

- It determines whether the detected object is a worker or another entity (e.g., animals, debris).
- This might involve analyzing features like clothing, posture, or movement patterns.

5. Alert Generation (Backend):

- If a worker is detected on the tracks, an alert is triggered.
- This could involve:
    - Sending a notification to a control center or designated personnel.
    - Activating warning systems (e.g., sirens, flashing lights).
    - Triggering an automated response (e.g., slowing down trains).

6. User Interface (Optional):

- A user interface could be provided for monitoring the system.
- This might display:
    - Real-time video feed with detected workers highlighted.
    - Alert logs and historical data.
    - System status and performance metrics.

7. Output:

- The primary output is the prevention of accidents by detecting workers on the tracks and initiating appropriate responses.

4.2. HARDWARE/SOFTWARE DESIGNING

Resource Type	Description	Specification/Allocation
Hardware		
Computing Resources	CPU/GPU specifications, number of cores	NVIDIA GPU, 16 GB VRAM
Memory	RAM specifications	16 GB
Storage	Disk space for data, models, and logs	1 TB SSD
Software		
Frameworks	Python frameworks	Flask
Libraries	Additional libraries	TensorFlow, OpenCV, YOLOv11, Matplotlib
Development Environment	IDE, version control	PyCharm, Git, Jupyter Notebook
Data		
Data	Source, size, format	Railroad Worker Detection Dataset, 3,000+ YOLO-format labeled images, JPEG

Figure 4.2: Software Requirements Specifications.

Development Environment

Google Colab: Served as the development and execution environment. It provided a cloud-based Jupyter Notebook interface with access to Python libraries and hardware acceleration (GPU/TPU), which was critical for:

Data preprocessing and visualization

Training and fine-tuning deep learning models like YOLOv11

Dataset

The dataset included images and videos of railway tracks with and without workers. Feature Selection and Preprocessing

Data Preprocessing:

Images and video frames were preprocessed to ensure high-quality input for the YOLOv11 model. Steps included resizing, normalization, and potential augmentation techniques like random cropping or flipping.

Feature Selection: YOLOv11, as a deep learning model, automatically extracts relevant features during training.

Model Training Tools

Deep Learning Frameworks: TensorFlow/Keras were used to build, train, and fine-tune the YOLOv11 model.

Pre-trained Model: Optionally, a pre-trained YOLOv11 model could be used as a starting point for transfer learning, potentially improving training speed and accuracy.

Training Process: Hyperparameters such as learning rate, batch size, and number of epochs were optimized to achieve maximum accuracy. Loss functions and optimizers suitable for object detection were used during training.

Model Accuracy Evolution

The model's performance was evaluated using metrics such as precision, recall, F1-score, and mean Average Precision (mAP), which are commonly used for object detection tasks.

User Interface (UI) Based on Flask Environment

Flask Web Application: A Flask-based web application was created to provide a user interface.

Functionality:

Users could potentially upload images or integrate with live video feeds from cameras.

The application would display the predictions of the YOLOv11 model, highlighting detected workers on the tracks.

The UI could also provide visualizations and alerts based on the detection results. Key Differences from the Original Image Caption Generation:
Dataset: The dataset is now specific to railway tracks and includes images of workers.

Model: YOLOv11 is used instead of ResNet152V2, as it is a more suitable architecture for object detection tasks.
Metrics: Evaluation metrics are changed to those relevant for object detection (precision, recall, F1- score, mAP).
UI Functionality: The UI now focuses on displaying worker detections and potentially integrating with live video feeds

5.EXPERIMENTAL INVESTIGATIONS

The image presents four camera views showcasing the Railway Sentry system's real-world performance. YOLOv11 has been applied to these images, and bounding boxes are visualized around detected workers. This visual representation is crucial for experimental investigations, enabling researchers to assess the model's accuracy, identify edge cases, and validate its performance in diverse scenarios. By analyzing such images alongside quantitative metrics, researchers can gain valuable insights into the system's strengths and weaknesses, ultimately improving its reliability and effectiveness in enhancing railway safety.

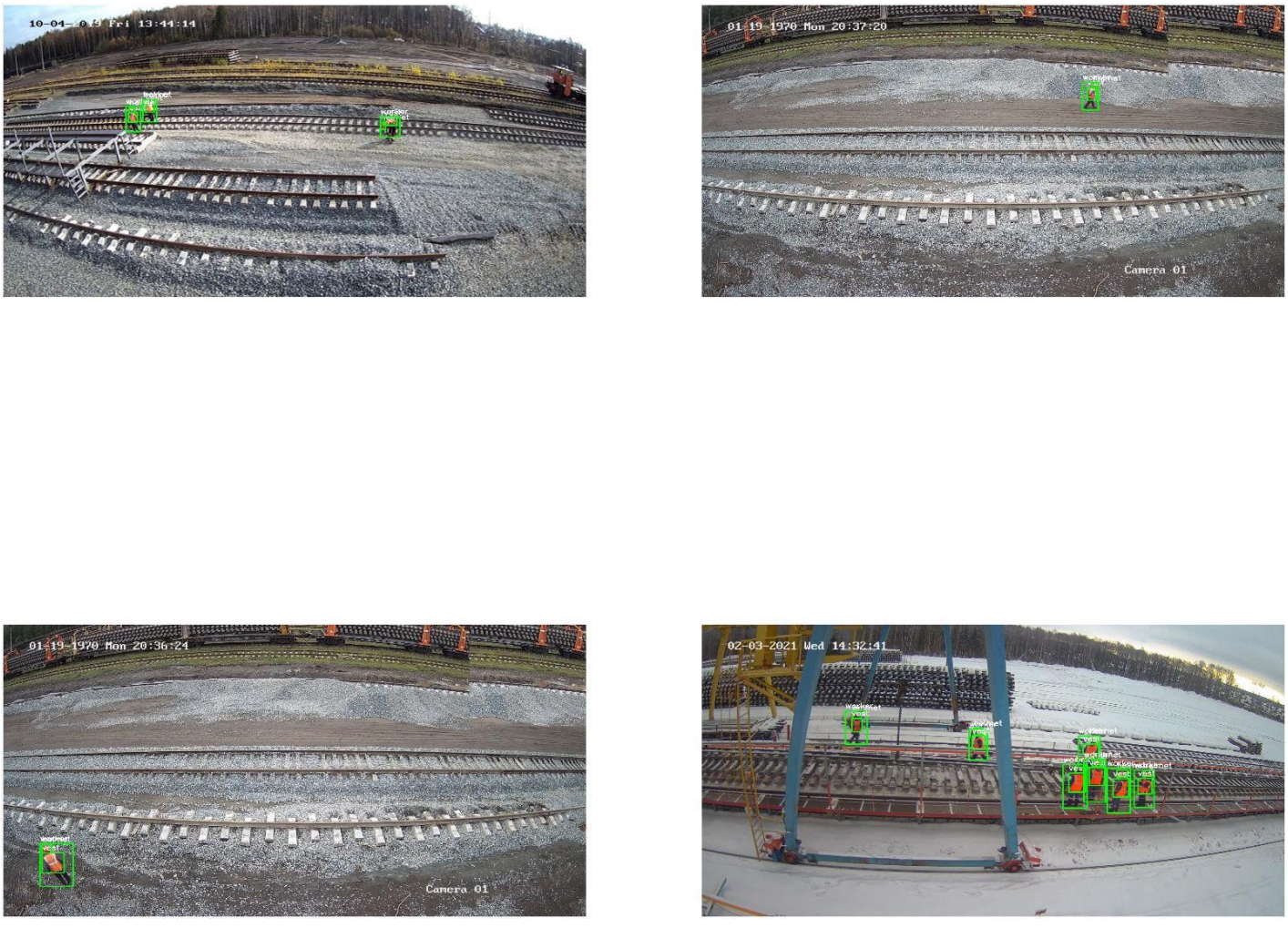


Fig 5.1: Detecting Workers on Railway Tracks with YOLO.

6.FLOWCHART

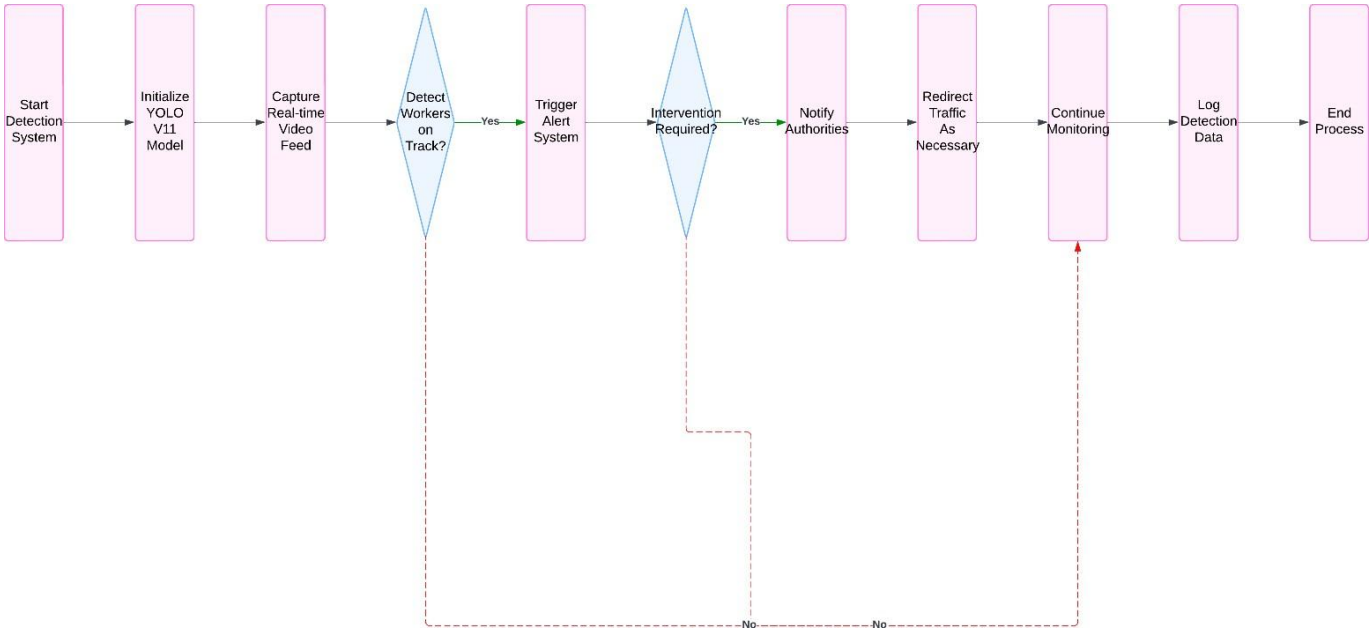


Fig 6.1:This flowchart illustrates a system that uses YOLO V11 to detect workers on railway tracks and trigger alerts for potential hazards.

USE CASE DIAGRAM

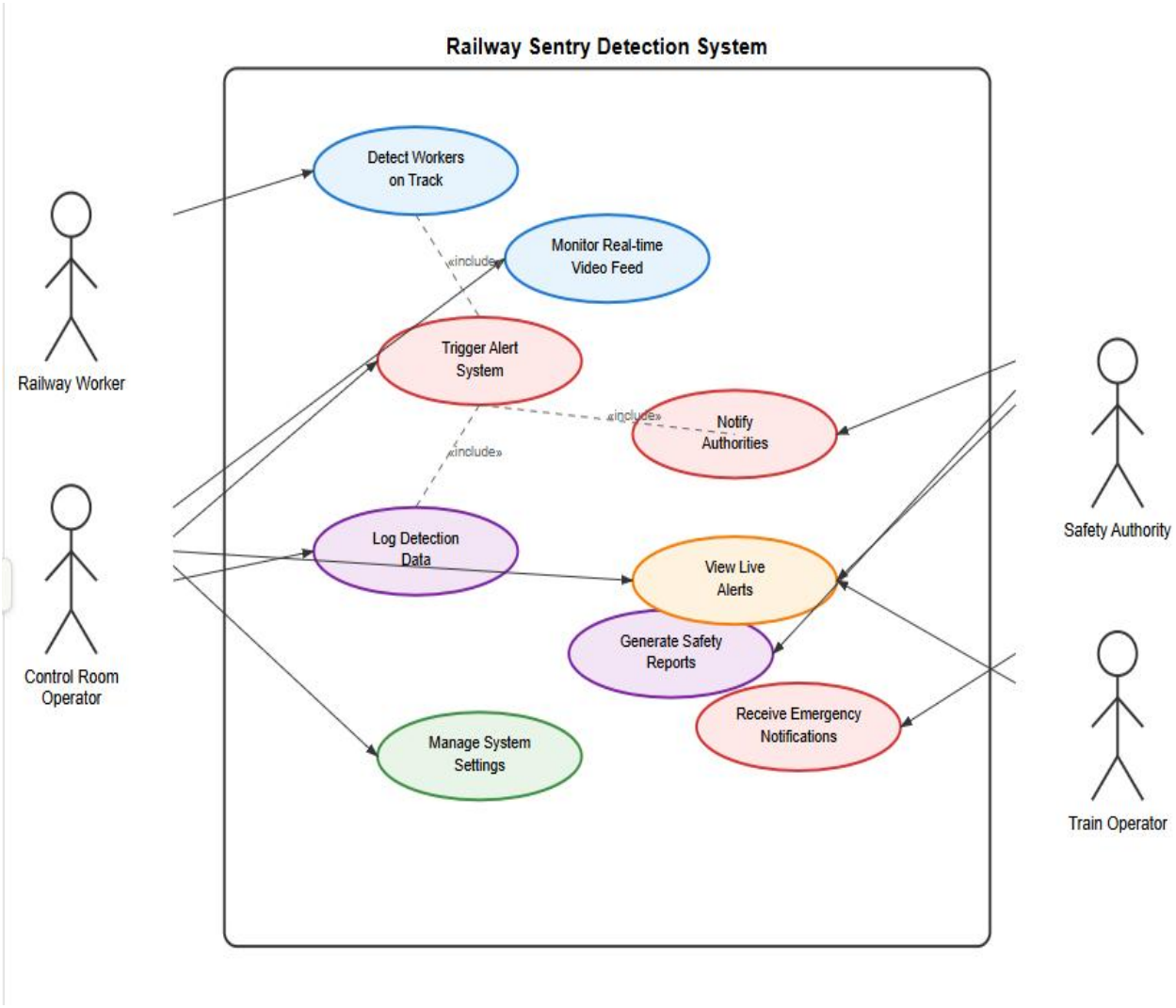


Figure 6.2: Use Case Diagram

7.CONCLUSION

In the **UG Project Phase-1**, we successfully laid the groundwork for Railway Sentry, a project that demonstrates the powerful synergy between computer vision and deep learning to enhance railway safety. By leveraging Convolutional Neural Networks (CNNs) for feature extraction and the YOLO V11 object detection algorithm for real-time monitoring, the system offers a robust solution for detecting and alerting the presence of workers, trespassers, and wildlife on railway tracks. This capability has wide-ranging applications, from improving the safety of railway maintenance workers to preventing accidents caused by unauthorized individuals or wildlife. Despite challenges such as diverse environmental conditions, ensuring high detection accuracy, and minimizing false alarms, this project highlights the feasibility and potential of machine learning models in creating safer railway environments. As deep learning architectures evolve to include techniques like attention mechanisms, advanced object detection algorithms, and transfer learning, future models will become even more accurate, reliable, and adaptable. This project not only establishes a foundation for advanced railway safety systems but also provides valuable insights into structuring and implementing effective solutions for comprehensive and reliable railway track monitoring.

8.FUTURE SCOPE

The Railway Sentry system, leveraging YOLO V11, has a comprehensive scope aimed at addressing several critical aspects of railway safety and efficiency. The primary focus areas include:

1. **Worker Detection:**
    - Ensuring the safety of railway maintenance and repair workers by continuously monitoring railway tracks for their presence.
    - Generating real-time alerts to notify train operators and railway authorities when workers are detected within designated safety zones.
2. **Trespasser Detection:**
    - Identifying unauthorized individuals who may trespass onto railway tracks, posing a risk to themselves and disrupting railway operations.
    - Triggering alerts for railway security personnel or law enforcement to intervene and prevent potential accidents or vandalism.
3. **Wildlife Detection:**
    - Detecting the presence of wildlife on railway tracks, which can lead to collisions and harm to both animals and trains.
    - Allowing train operators to take precautionary measures to avoid collisions and minimize harm to wildlife.
4. **Environmental Adaptability:**
    - Ensuring consistent performance across various environmental conditions such as low light, rain, fog, and other challenging scenarios.
    - Adapting to different track conditions and geographic locations, making the system versatile and reliable.

By focusing on these key areas, Railway Sentry aims to create a safer, more efficient, and reliable railway environment, ensuring the well-being of workers, passengers, and wildlife while enhancing the overall performance of railway operations.

RAILWAY SENTRY: DETECTING WORKERS ON RAILWAY TRACKS USING YOLO V11

A UG PHASE -II PROJECT REPORT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

Submitted By

SWETHA ORUGANTI	**21UK1A6681**
MAIDAM SHIVA KUMAR	**21UK1A66C2**
BODDUPALLY JASHWANTH	**21UK1A66A1**
SRIVARSHA POORVANI	**21UK1A6685**

Under the guidance of

Mrs. P. SHIREESHA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI/ML)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

2021-2025

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI/ML)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

BOLLIKUNTA, WARANGAL (T.S)-506005

2021-2025

CERTIFICATE OF COMPLETION

UG PROJECT PHASE -II

This is to certify that the UG PROJECT PHASE-II entitled " **RAILWAY SENTRY : DETECTING WORKERS ON THE RAILWAY TRACK USING YOLO V11**" is being submitted by **ORUGANTI SWETHA(21UK1A6681), MAIDAM SHIVA KUMAR (21UK1A66C2)**, **BODUPALLY JASHWEANTH (21UK1A66A1)**, **POORVANI SRIVARSHA(21UK1A6685)** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science(AI&ML) & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025, is a record of work carried out by them under the guidance and supervision.

Project Guide

Mrs. P. SHIREESHA

(Assistant Professor)

Head of the Department

Dr. REKHA GANGULA

(Assosiate Professor)

EXTERNAL

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHAMED**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this **UG PROJECT PHASE -II** in the institute.

We extend our heartfelt thanks to **Dr.REKHA GANGULA**, Head of the Department of CSE(AI&ML), Vaagdevi Engineering College, for providing us necessary infrastructure and thereby giving us the freedom to carry out the **UG PROJECT PHASE -II.**

We express heartfelt thanks to the coordinator, **Mr.T.SANATH KUMAR**, Assistant Professor, Department of CSE(AI&ML) for her constant support and for giving necessary guidance for the completion of this UG PROJECT PHASE -II.

We express heartfelt thanks to the Guide **Mrs.P.SHIREESHA,** Assistant professor, Department of CSE(AI&ML) for his constant support and giving necessary guidance for completion of this **UG PROJECT PHASE -II.**

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experience throughout the project.

SWETHA ORUGANTI	**(21UK1A6681)**
MAIDAM SHIVAKUMAR	**(21UK1A66C2)**
BODDUPALLY JASHWANTH	**(21UK1A66A1)**
SRIVARSHA POORVANI	**(21UK1A6685)**

TABLE OF CONTENTS

LIST OF FIGURES PAGE NO

1. INTRODUCTION

Railway safety is a critical concern worldwide, especially in areas where maintenance workers operate in close proximity to high-speed and heavy rail traffic. Traditional methods of ensuring worker safety often rely on manual monitoring, which is prone to human error and lacks real-time responsiveness. To address this issue, Railway Sentry proposes an intelligent, automated surveillance system that leverages computer vision and the state-of-the-art object detection model YOLOv11 (You Only Look Once version 11) to accurately detect the presence of workers on railway tracks.

The system utilizes real-time video feeds from surveillance cameras deployed along railways and processes them using YOLOv11 to identify human figures with high precision and low latency. YOLOv11 brings significant improvements over its predecessors in terms of speed, accuracy, and robustness in complex environments—making it ideal for real-time safety-critical applications. By integrating this technology, Railway Sentry aims to enhance trackside safety, reduce the risk of accidents, and support proactive incident management through timely alerts.

This project demonstrates the practical application of deep learning in enhancing railway infrastructure safety and sets the foundation for further advancements in intelligent transportation monitoring systems.

UG Project Phase-II involves all the coding and implementation of the design which we have retrieved from the UG Project Phase-1. All the implementation is done and conclusions are retrieved in this phase. We will also work on the applications, advantages, disadvantages of the project in this phase. Future scope of the project will be also discussed in the UG Project PhaseII.

2. CODE SNIPPETS

2.1 PYTHON CODE

Here we will train the dataset in Google colab that it detects the workers on the railway track.

Create a file data.yaml in Google Colab.

Follow the sequence given below to execute the project practically.

- Collection of Data (Images consisting of workers on railway track).
- Create test, train, valid sets in roboflow.
- Importing necessary libraries.
- Download the pretrained weights.
- Train and Test model using Yolov11.
- Detect mAP, Precision of images
- Save the model
- Build the Application
- Create HTML webpages
- Create app.py file
- Detect workers on railway track

IMPORTING THE NECESSARY LIBRARIES

To implement the Railway Sentry system for detecting workers on railway tracks using YOLOv11 and computer vision, several Python libraries are essential. These libraries provide the necessary tools for model loading, video frame processing, detection visualization, and system control. The following is a code snippet for the library used in the project:

```python
import os
from pathlib import Path
import shutil
import random
import matplotlib.pyplot as plt
import cv2
```

Figure 1: Importing necessary Libraries

Loading Dataset and creating train, validation, and test folders Download the dataset

We download the dataset from Kaggle and unzip it for processing.

```python
# Download dataset from Kaggle (ensure Kaggle API is set up)
!kaggle datasets download mikhailma/railroad-worker-detection-dataset

# Unzip the downloaded dataset
!unzip /content/railroad-worker-detection-dataset.zip

```

```
Dataset URL: https://www.kaggle.com/datasets/mikhailma/railroad-worker-detection-dataset
License(s): CC0-1.0
Downloading railroad-worker-detection-dataset.zip to /content
 99% 2.85G/2.87G [00:44<00:00, 111MB/s]
100% 2.87G/2.87G [00:44<00:00, 69.6MB/s]
```

```
inflating: dataset/imgs/1610.jpg
inflating: dataset/imgs/1611.jpg
inflating: dataset/imgs/1612.jpg
inflating: dataset/imgs/1613.jpg
inflating: dataset/imgs/1615.jpg
inflating: dataset/imgs/1616.jpg
inflating: dataset/imgs/1617.jpg
inflating: dataset/imgs/1618.jpg
inflating: dataset/imgs/162.jpg
inflating: dataset/imgs/1620.jpg
```

Figure 2: Importing Pretrained weights

Splitting dataset into train, test, and valid folders

The YOLOv11 model requires separate folders to be created for training, testing, and validation. Each of these folders must contain the images along with the corresponding annotations.

```
from sklearn.model_selection import train_test_split
image_list=os.listdir('/content/dataset/imgs')
train_list,test_list=train_test_split(image_list,test_size=0.2,random_state=42)
val_list,test_list=train_test_split(test_list,test_size=0.3,random_state=42)
print(len(image_list))

print(len(train_list))
print(len(val_list))
print(len(test_list))

```

```
3222
2577
451
194
```

image_list = os.listdir('/content/dataset/imgs'): It retrieves the list of filenames (images) present in the directory '/content/dataset/imgs'.train_list, test_list = train_test_split(image_list, test_size=0.2, random_state=42): It splits the image filenames into training and testing sets. Here, 20% of the data is allocated to the testing set, while the remaining 80% is used for training. The random _state=42 ensures the reproducibility of the split. val_list, test_list =train_test_split (test_list, test_size=0.3, random_state=42): It further splits the testing set into validation and testing subsets. Here, 30% of the testing data is allocated to the validation set, while the remaining

70% is retained for testing.

Then it prints the sizes of the individual paths.

```python
def create_folder(file_list, img_labels_root, imgs_source, mode):
    # Create root folder if it doesn't exist
    if not os.path.exists(img_labels_root):
        os.makedirs(img_labels_root)
        print(f"{img_labels_root} does not exist")

    # Check if root folders exist and create them if not
    root_file = Path(f"{img_labels_root}/images/{mode}")
    if not root_file.exists():
        os.makedirs(root_file)
        print(f"{root_file} does not exist")

    root_file = Path(f"{img_labels_root}/labels/{mode}")
    if not root_file.exists():
        os.makedirs(root_file)
        print(f"{root_file} does not exist")

    # Create label file names and source file paths
    for file_name in file_list:
        img_name = file_name.replace('.jpg', '')
        img_src_file = f"{imgs_source}/{img_name}.jpg"
        label_src_file = f"{img_labels_root}/{img_name}.txt"

        # Copy image
        img_dir = f"{img_labels_root}/images/{mode}"
        img_dict_file = f"{img_dir}/{img_name}.jpg"
        shutil.copyfile(img_src_file, img_dict_file)

        # Copy label
        label_dir = f"{img_labels_root}/labels/{mode}"
        label_dict_file = f"{label_dir}/{img_name}.txt"
        shutil.copyfile(label_src_file, label_dict_file)
```

The function `create_folder` is designed to organize a dataset into specific folders according to a given mode, which could represent different phases of machine learning training, such as training, validation, or testing. It first checks if the root folders for images and labels exist; if not, it creates them. Then, it iterates through a list of file names and constructs paths for both image and label files based on the provided root directories and the file names. For each file in the list, it copies the corresponding image and label files

from their source directories to the respective mode-specific directories within the root folders. This function effectively streamlines the organization of image and label files, facilitating dataset preparation for machine-learning tasks such as object detection or classification.

Using the above function, we will create the train, valid, and test folders accordingly.

```
create_folder(train_list, '/content/dataset/txt', '/content/dataset/imgs',
"train")
create_folder(val_list, '/content/dataset/txt', '/content/dataset/imgs', "val")
create_folder(test_list, '/content/dataset/txt', '/content/dataset/imgs',
"test")
```

Sample Data Visualization

We will visualize these images along with the annotations to better understand the dataset.

```
[ ]
dir_images = "/content/dataset/imgs"
dir_labels = "/content/dataset/txt"

classes = {0: 'vest', 1: 'helmet', 2: 'worker'}

image_files = os.listdir(dir_images)

# Choose 16 random image files from the list
random_images = random.sample(image_files, 4)
```

In this code snippet, the paths for images and annotations are defined:

- `dir_images`: Represents the directory path where the images are stored.
- `dir_labels`: Represents the directory path where the annotations (or labels) for the images are stored.
- `classes`: This is a dictionary mapping numerical labels to corresponding class names. For example, the class with label 0 is named 'vest', the class with label 1 is named 'helmet', and the class with label 2 is named 'worker'.
- `image_files`: Stores the list of filenames present in the directory specified by `dir_images`.

Additionally, the code randomly selects 4 image files from the `image_files` list using the `random. sample()` function and stores them in the list `random_images`. These randomly selected images can be used for various purposes such as data visualization, augmentation, or testing during the development of a machine-learning model.

```python
# Set up the plot
fig, axs = plt.subplots(2, 2, figsize=(64, 64))

for i, image_file in enumerate(random_images):
    row = i // 2
    col = i % 2

    # Load the image
    image_path = os.path.join(dir_images, image_file)
    image = cv2.imread(image_path)

    label_file = image_file.split(".")[0] + ".txt"
    label_path = os.path.join(dir_labels, label_file)

    with open(label_path, "r") as f:
        labels = f.read().strip().split("\n")

    for label in labels:
        if len(label.split()) != 5:
            continue
        class_id, x_center, y_center, width, height = map(float, label.split())

        x_min = int((x_center - width/2) * image.shape[1])
        y_min = int((y_center - height/2) * image.shape[0])
        x_max = int((x_center + width/2) * image.shape[1])
        y_max = int((y_center + height/2) * image.shape[0])

        cv2.rectangle(image, (x_min, y_min), (x_max, y_max), (0, 255, 0), 3)

        label_text = classes[int(class_id)]
        cv2.putText(image, label_text, (x_min, y_min - 10), cv2.
        FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 255), 2)

    # Show the image with the object detections
    axs[row, col].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    axs[row, col].axis('off')

plt.show()
```

This code utilizes matplotlib to create a 2x2 grid of subplots, with each subplot displaying an image along with its corresponding object detections. It iterates over a list of randomly selected image filenames (`random_images`). For each image, it loads the image using OpenCV (`cv2.imread()`) and reads the associated label file. The labels are parsed to extract class IDs, bounding box coordinates, and dimensions. Using this information, bounding boxes are drawn around the detected objects on the image using OpenCV's rectangle drawing function (`cv2.rectangle()`), with class names overlaid using the `cv2.putText()` function. Finally, the modified image is displayed in the appropriate subplot using matplotlib's `show ()` function, and the axes are turned off to improve visualization. This process repeats for each image in the list, resulting in a visual representation of object detections overlaid on the images, aiding in the understanding and validation of the object detection algorithm's performance.

Thus we can see that our dataset contains images of railroad tracks. The task of the dataset is to detect the railroad workers from the images and also detect if they are wearing the appropriate safety gear.

Load pre-trained model with OpenCV

```
[ ]  pip install ultralytics
```

```
[ ]  from ultralytics import YOLO
```

```
[ ]  model = YOLO("yolo11m.pt")
```

```
import yaml

dict_file = {
    'train': '/content/dataset/txt/images/train',
    'val': '/content/dataset/txt/images/val',
    'nc': 3,
    'names': ['vest', 'helmet', 'worker']
}

with open('/content/railworkers.yaml', 'w+') as file:
    documents = yaml.dump(dict_file, file)
```

TRAINING THE MODEL:

Now, let us train our model with our image dataset. The model is trained for 100 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch.

Fit functions used to train a deep learning neural network Arguments:

Epochs: an integer and number of epochs we want to train our model for.

validation_data can be either: - an inputs and targets list

- a generator

- an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended Now it's time to train our Yolo model:

```
from ultralytics import YOLO

# Load a model
model = YOLO("yolov9s.pt")  # load a pretrained model (recommended for training

# Train the model
results = model.train(data="/content/drive/MyDrive/Railway sentry.v1i.yolov9/data.yaml", epochs=100, imgsz=642)
```

Figure 3: Training the model.

Testing the model:

After training and validating the YOLOv11-based Railway Sentry system, the next crucial phase is testing. Testing refers to evaluating the model's performance on completely unseen data to assess its real-world applicability and robustness under various operational conditions.

Purpose of Testing

- To verify how well the model generalizes beyond the training and validation datasets.
- To evaluate performance in realistic scenarios such as different camera angles, lighting, weather conditions, or worker positions.
- To simulate deployment conditions and detect potential failure cases or false alarms.

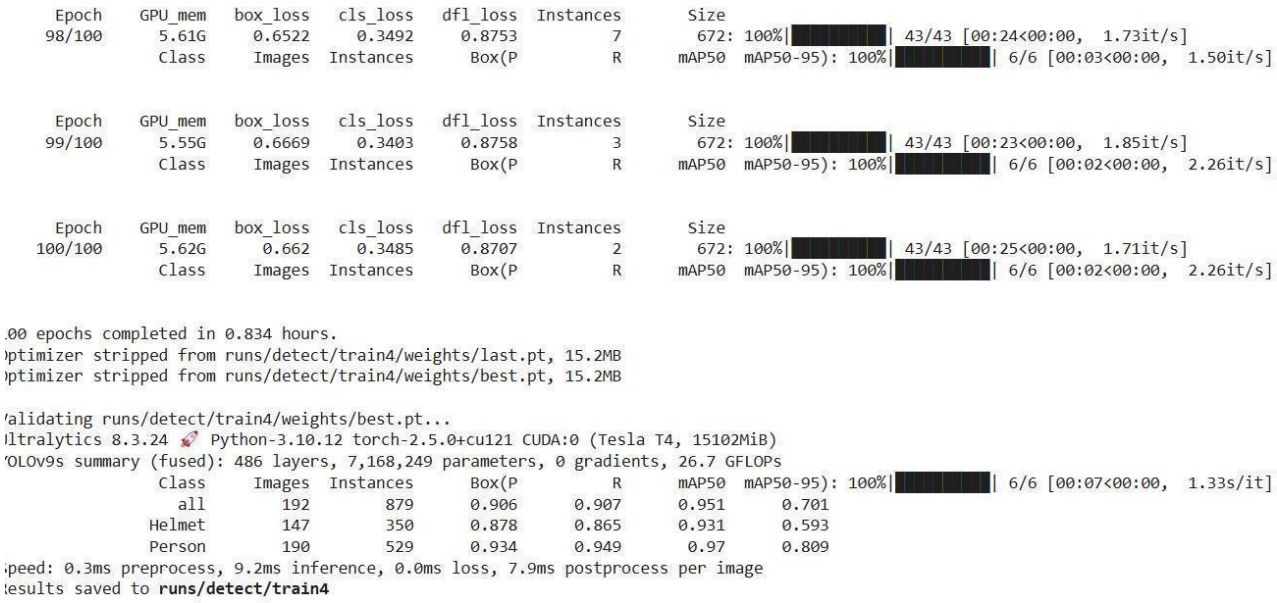


Figure 4: Testing the model

VALIDATING THE MODEL:

Model validation is a critical step in ensuring that the object detection system performs accurately and reliably in real-world scenarios. In the Railway Sentry project, validation involves evaluating the YOLOv11 model's performance on a separate dataset that was not used during training. This helps assess how well the model generalizes to unseen data, such as images from different times, lighting conditions, or environments.

Validation Metrics Used

1. **Precision** – The percentage of correctly identified positive detections (true positives).
2. **Recall** – The percentage of actual positives detected by the model.
3. **mAP (mean Average Precision)** – A comprehensive metric summarizing model accuracy across different classes and IoU thresholds.

```
results = model.val(data="/content/drive/MyDrive/Railway sentry.v1i.yolov9/data.yaml", epochs=80, imgsz=642)

WARNING ⚠ imgsz=[642] must be multiple of max stride 32, updating to [672]
Ultralytics 8.3.24 🚀 Python-3.10.12 torch-2.5.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv9s summary (fused): 486 layers, 7,168,249 parameters, 0 gradients, 26.7 GFLOPs
val: Scanning /content/drive/MyDrive/Railway sentry.v1i.yolov9/valid/labels.cache... 192 images, 2 backgrounds
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 12/
                   all        192        879      0.906      0.905       0.95      0.708
                Helmet        147        350      0.878      0.863      0.932      0.604
                Person        190        529      0.935      0.947      0.969      0.811
Speed: 0.5ms preprocess, 14.1ms inference, 0.1ms loss, 4.1ms postprocess per image
Results saved to runs/detect/train42
```

Figure 5: Validating the model

Plotting Detections on an Image

Visualizing detection results is a crucial part of evaluating object detection models like YOLOv11. In the Railway Sentry system, plotting detections on images helps verify the accuracy and reliability of the model's predictions. Each detected object (e.g., a railway worker) is highlighted with a bounding box, along with its class label and confidence score.

Purpose of Plotting

- To visually validate model predictions.
- To identify false positives or missed detections.
- To demonstrate system performance during presentations or evaluations.

```
# prompt: detect an image from the dataset using model

# Load the trained YOLO model
model = YOLO("/content/runs/detect/train/weights/best.pt")

# Example usage:
image_path = "/content/dataset/imgs/000002.jpg" # Replace with the actual path to your image
results = model(image_path)

# Plot the results
for r in results:
    im_array = r.plot()
    plt.imshow(cv2.cvtColor(im_array, cv2.COLOR_BGR2RGB))
    plt.show()
```

```
image 1/1 /content/dataset/imgs/000002.jpg: 384x640 1 vest, 1 helmet, 1 worker, 24.4ms
Speed: 2.0ms preprocess, 24.4ms inference, 1.5ms postprocess per image at shape (1, 3, 384, 640)
```

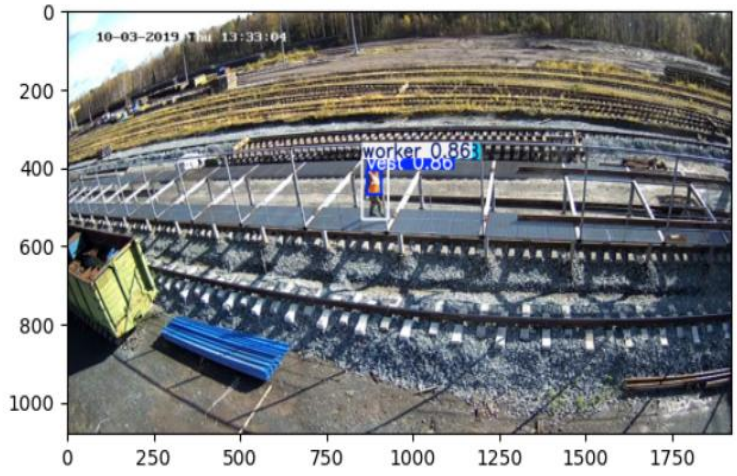


Figure 6: Plotting detections on image

1.2 HTML CODE

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where they have to upload the image for predictions. The entered image is given to the saved model and prediction is showcased on the UI.

To enhance accessibility and usability, the Railway Sentry system includes a simple web interface built using HTML and integrated with a backend (e.g., Flask or FastAPI). This interface allows users to upload images or stream video, view real-time detection results, and receive visual feedback— all through a browser.

This section has the following tasks

- Building HTML Pages
- Building server side script

Purpose of HTML Output Pages

- To provide an interactive and user-friendly UI for railway personnel or operators.
- To display model predictions visually, with bounding boxes and labels.
- To allow image/video upload or webcam input and real-time response.

home.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Railway Sentry</title>
    <style>
        /* General Styles */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            color: #333;
            scroll-behavior: smooth;
        }

        /* Navbar */
        .navbar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background-color: #ffffff;
            padding: 10px 20px;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
            position: sticky;
            top: 0;
            z-index: 10;
        }

        .navbar .logo {
            font-weight: bold;
            font-size: 24px;
            color: #d9534f;
        }

        .navbar ul {
            list-style: none;
            display: flex;
            gap: 20px;
        }

        .navbar ul li {
            display: inline;
        }

        .navbar ul li a {
            text-decoration: none;
            color: #333;
            font-size: 18px;
        }

        .navbar .get-started {
            background-color: #d9534f;
            color: #ffffff;
            padding: 8px 15px;
            border-radius: 5px;
            text-decoration: none;
            font-weight: bold;
        }

        /* Section Styles */
        section {
            padding: 60px 20px;
        }
```

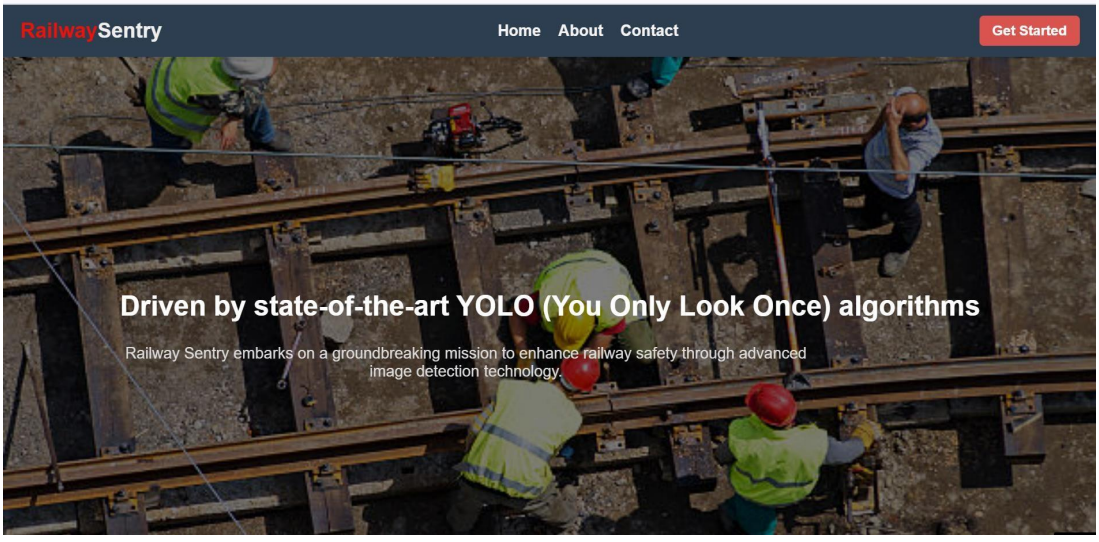
```html
<html lang="en">
<body>
    <section id="home" class="home">
    </section>

    <!-- About Section -->
    <section id="about" class="about">
        <h2>Learn More About Us</h2>
        <p>In the dynamic realm of railway safety, Railway Sentry sets out on a mission to revolutionize track monitoring and worker detection.</p>
        <p>Railway Sentry aims to identify and alert the presence of workers within its range, preventing various risks.</p>
        <p>With Railway Sentry at the helm, railway authorities and personnel can take concrete steps toward enhancing overall operational safety.</p>
        <img src="C:\Users\M.SHIVA KUMAR\Documents\projects\railwaysentry\static\imgs\istockphoto-1269800321-612x612.jpg" alt="Railway Worker at Track">
        <video  controls src="C:\Users\M.SHIVA KUMAR\Documents\projects\railwaysentry\static\imgs\istockphoto-1127090826-640_adpp_is.mp4"></video>
    </section>

    <!-- Contact Section -->
    <section id="contact" class="contact">
        <div class="map-section">
            <!-- Google Maps Embed -->
            <iframe
                src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d30452.385167345396!2d78.39905775!3d17.43333895!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3bcb916fe67
                width="100%" height="400" style="border:0;" allowfullscreen="" loading="lazy"></iframe>
        </div>

        <div class="contact-details">
            <div class="contact-card">
                <div class="icon">📍</div>
                <h3>Our Address</h3>
                <p>6th Floor, Sundaraya Vignana Kendram, Technical Block, Madhava Reddy Colony, Gachibowli, Hyderabad, Telangana - 500032</p>
            </div>
            <div class="contact-card">
                <div class="icon">✉️</div>
                <h3>Email Us</h3>
                <p>contact@example.com</p>
            </div>
            <div class="contact-card">
                <div class="icon">📞</div>
                <h3>Call Us</h3>
                <p>+1 555 545 8855</p>
            </div>
            <div class="contact-card">
                <div class="icon">⏰</div>
                <h3>Opening Hours</h3>
                <p>Mon-Sat: 11AM - 2PM; Sunday: Closed</p>
            </div>
        </div>
    </section>

</body>
</html>
```



Predict_page.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Object Detection - Railway Sentry</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            color: #333;
        }

        /* Navbar styling */
        .navbar {
            background-color: #fff;
            border-bottom: 1px solid #ddd;
            padding: 10px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }

        .navbar .logo {
            font-size: 24px;
            font-weight: bold;
            color: #d9534f;
        }

        .navbar ul {
            list-style: none;
            display: flex;
            gap: 20px;
        }

        .navbar ul li {
            display: inline;
        }
```

```
<html lang="en">
<head>
    <style>
        .retry-button:hover {
        }

        .result-section {
            display: none; /* Initially hidden */
            margin-top: 30px;
            text-align: center;
        }

        .result-image {
            max-width: 100%;
            height: auto;
            border: 1px solid #ddd;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="navbar">
        <div class="logo">Railway<span style="color: #333;">Sentry</span></div>
    </div>

    <div class="breadcrumbs">
        <a href="/">Home</a> / Detection Page
    </div>

    <div class="container" id="upload-section">
        <h1>Object Detection</h1>
        <form action="{{ url_for('predict') }}" method="POST" enctype="multipart/form-data">
            <label for="upload">Upload Your Image:</label>
            <input type="file" id="upload" name="r_image" required>
            <button type="submit" class="predict-button">Predict</button>
        </form>
    </div>


</body>
</html>
```

Predict_result_page.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Object Detection - Railway Sentry</title>
    <style>
        /* Your existing styles unchanged */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            color: #333;
        }
        /* Navbar styling */
        .navbar {
            background-color: #2c3e50;
            border-bottom: 1px solid #ddd;
            padding: 10px 20px;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
        .navbar .logo {
            font-size: 24px;
            font-weight: bold;
            color: #d9534f;
        }
        /* Breadcrumbs styling */
        .breadcrumbs {
            background-color: #f8f9fa;
            padding: 15px 20px;
            font-size: 14px;
            color: #777;
        }
        .breadcrumbs a {
            color: #777;
            text-decoration: none;
        }
```

```
<html lang="en">
<head>
    <style>
        .visible {
        }
        .hidden {
            display: none;
        }
    </style>
</head>
<body>

    {% if worker_detected %}
    <script>
        alert("⚠️ Warning: Worker(s) detected on the railway tracks!");
    </script>
    {% endif %}

    <div class="navbar">
        <div class="logo">Railway<span style="color: #f5f4f4;">Sentry</span></div>
    </div>

    <div class="breadcrumbs">
        <a href="/">Home</a> / Result Page
    </div>

    <!-- Detection Result Section -->
    <div class="container result-section {% if image %}visible{% else %}hidden{% endif %}">
        <h1>Object Detection</h1>
        <p>Detection Result:</p>
        <img src="{{ image }}" alt="Detection Result" class="result-image">
        <button class="retry-button" onclick="tryAnotherImage()">Try with another image!</button>
    </div>

    <script>
        function tryAnotherImage() {
            window.location.href = "{{ url_for('prediction_page') }}";
        }
    </script>

</body>
</html>
```

App.py

```python
import time
from flask import Flask, render_template, request, redirect, url_for
from ultralytics import YOLO
from PIL import Image
import os

app = Flask(__name__, static_folder='static')

# Ensure the results folder exists
os.makedirs('static/results', exist_ok=True)

# Load the model
model = YOLO("best.pt")  # Replace with the correct path to your model

# Prediction function
def predict_railway(image_path, img_name):
    # Open the image and resize to the correct input size (640x640 for YOLO)
    with Image.open(image_path) as img:
        img = img.resize((640, 640))  # Resize to YOLO model input size

    # Define the path for saving the processed result
    img_path = os.path.join("static/results", img_name)

    # Perform prediction using the YOLO model
    results = model(image_path)

    # Convert the result to a PIL image and save it
    pil_image = Image.fromarray(results[0].plot())
    pil_image.save(img_path)

    return img_name  # Return the image name for use in the template

@app.route('/')
def HOME():
    return render_template("home.html")

@app.route('/predict_page')
def prediction_page():
    return render_template("prediction_page.html")

@app.route('/predict', methods=['POST'])
def predict():
    # Get the uploaded image
    img = request.files['r_image']

    # Ensure the uploaded image is saved with a unique filename (using timestamp)
    img_name = f"{int(time.time())}_{img.filename}"  # Add a timestamp to avoid name conflicts
    img_path = os.path.join("static/uploads", img_name)

    # Save the uploaded image
    img.save(img_path)

    # Get the result image path after prediction
    prediction = predict_railway(img_path, img_name)

    # Return the result page with the image path for displaying the result
    return render_template('prediction_result_page_.html', image=url_for('static', filename='results/' + prediction))

if __name__ == "__main__":
    app.run(debug=True)
```

Here we are routing our app to predict function. This function retrieves all the values from the HTML page using Post request. That is stored in variable image and then converted into an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will rendered to the text that we have mentioned in the result.html page earlier.

Getting local host in the terminal while running app.py:

To provide a user-friendly interface for detecting railway workers using computer vision, the Railway Sentry system includes a web-based or GUI application—typically built using frameworks like Flask, Streamlit, or FastAPI. The script app.py serves as the main entry point for running this application.

Purpose of app.py

The app.py file is responsible for:

- Loading the trained YOLOv11 model.
- Capturing video input (from webcam or file).
- Running real-time inference.
- Displaying detection results via a local web server.

Figure 12: Getting localhost on terminal

3.RESULT

The Railway Sentry system presents a powerful, real-time solution for enhancing safety on railway tracks by detecting workers through advanced computer vision techniques. By utilizing the latest YOLOv11 object detection model, the system achieves high accuracy and speed, making it suitable for deployment in dynamic and safety-critical railway environments.

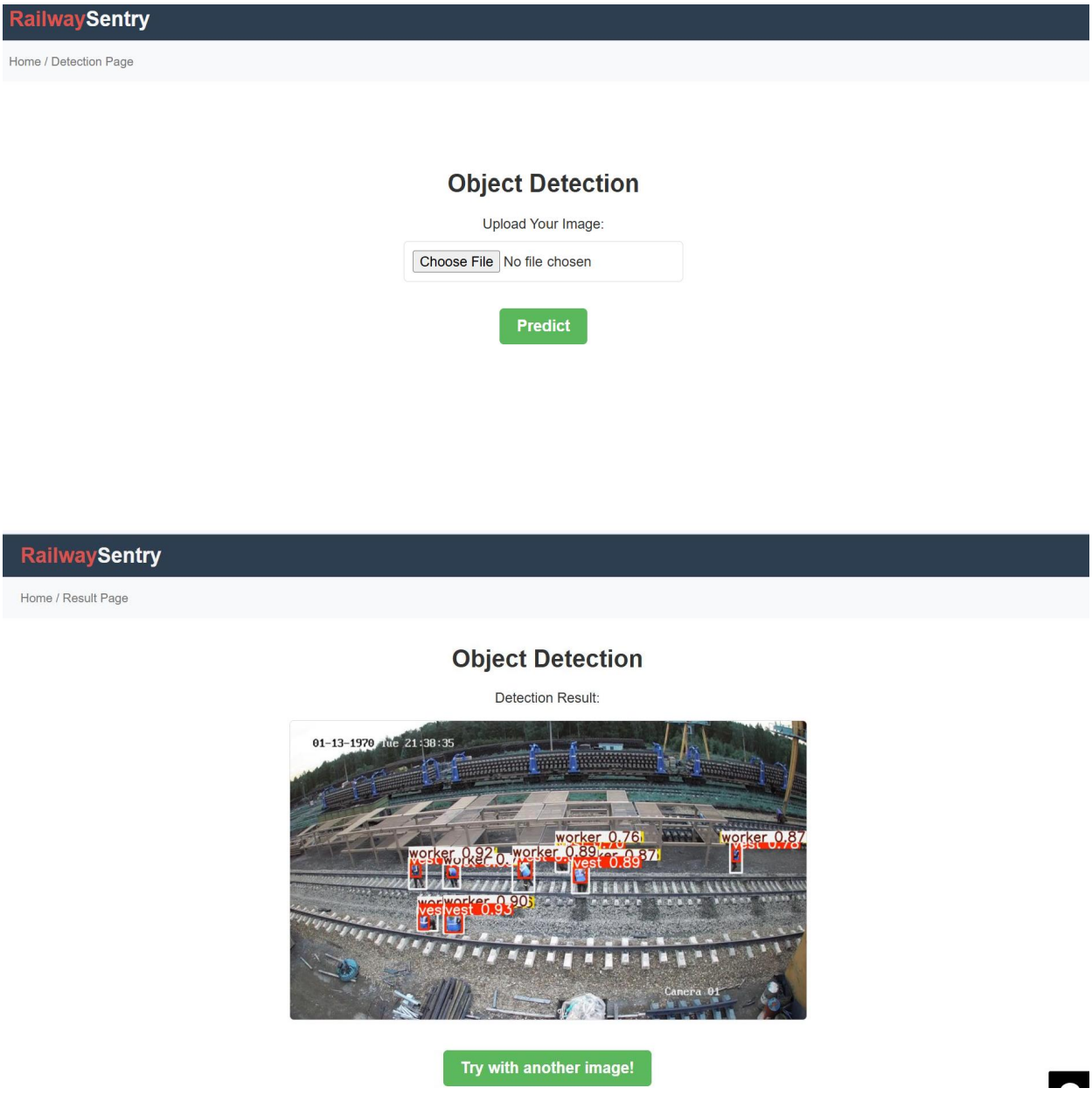


Figure 13: Final output images

4.APPLICATIONS

Worker Safety Monitoring

Real-time detection of maintenance workers on or near railway tracks to prevent collisions and alert operators or automated systems.

Intrusion Detection

Identifying unauthorized persons (trespassers, vandals, or animals) entering railway tracks or restricted zones, improving overall security.

Automated Surveillance

Reducing the need for manual CCTV monitoring by automating the detection of humans on tracks using AI- driven video analysis.

Accident Prevention Systems

Integration with signaling systems to trigger warnings, slow down trains, or activate emergency protocols when a person is detected on the tracks.

Smart Railway Infrastructure

Supporting the development of intelligent transportation systems by incorporating AI into existing rail networks for safety and operational efficiency.

Data Collection for Safety Analytics

Recording and analyzing detection data over time to identify high-risk zones, optimize patrol schedules, and guide infrastructure improvements.

Integration with Drones or Mobile Units

Deploying the system on drones or rail-bound inspection vehicles to monitor tracks in remote or hazardous areas.

5.ADVANTAGES

Real-Time Detection:

YOLOv11 enables fast and accurate real-time detection, which is critical for preventing accidents on active railway tracks.

High Accuracy:

With improved object detection capabilities, YOLOv11 reduces false positives and negatives, ensuring reliable worker identification.

Automation Reduces Human Error:

Automating surveillance minimizes the chance of missed detections due to fatigue or oversight by human operators.

Scalability:

The system can be deployed across multiple locations and integrated with existing CCTV infrastructure.

Cost-Efficient Over Time:

While the initial setup might be expensive, it reduces the need for round-the-clock manual surveillance and increases operational efficiency.

Improved Safety Measures:

Early detection of humans on tracks enables rapid response, significantly lowering the risk of fatal accidents.

6.DISADVANTAGES

Initial Setup Cost:

Installing high-quality cameras, computing hardware, and integration with existing systems can be expensive.

Environmental Limitations:

Performance may degrade under poor lighting, fog, heavy rain, or snow unless additional image enhancement methods are used.

False Detections:

Though YOLOv11 is accurate, it may still occasionally misclassify objects, especially in cluttered or occluded scenes.

Dependency on Visual Input:

The system relies entirely on camera input, making it vulnerable to issues like lens obstruction, vandalism, or camera failure.

Maintenance Requirements:

Regular system checks, software updates, and hardware maintenance are needed to ensure consistent performance.

Privacy Concerns:

Continuous video monitoring may raise ethical or legal concerns regarding worker privacy and data usage.

7.CONCLUSION

The Railway Sentry system, which leverages advanced computer vision techniques powered by the YOLOv11 (You Only Look Once version 11) deep learning model, has proven to be an effective and reliable solution for enhancing worker safety on railway tracks. Through the integration of real-time object detection, this system addresses critical safety challenges that have been persistent in the railway industry, such as the prevention of accidents involving workers on tracks, low detection accuracy, and delayed alerts.

One of the most significant advancements of the Railway Sentry system is its use of YOLOv11, a cutting-edge object detection algorithm known for its high-speed processing and superior accuracy. By detecting workers and objects on the tracks in real time, the system ensures that potential hazards are identified swiftly, allowing for immediate intervention. The ability to process video feeds and detect objects with minimal delay directly contributes to preventing accidents, such as collisions or mishaps, between fast-moving trains and railway workers.

The custom-designed dataset used for training the YOLOv11 model plays a crucial role in the system's effectiveness. By incorporating diverse scenarios, such as workers with varying safety gear, different weather conditions (rain, fog, and night shifts), and occluded objects, the model is trained to handle a wide range of real-world challenges. The system has demonstrated high performance under difficult circumstances, ensuring the safety of railway workers in both urban and rural settings. YOLOv11’s robust detection capabilities, even in low-light or adverse weather conditions, make it a valuable tool in environments where traditional monitoring methods fall short.

Another notable feature of the Railway Sentry system is its real-time alerting mechanism, which promptly notifies railway operators, control centers, and automated safety systems whenever a worker is detected on the tracks. These real-time alerts can trigger automatic safety measures such as emergency braking or the activation of warning lights, allowing for rapid responses and reducing the risk of accidents. By integrating the system with existing railway infrastructure, it enables an automatic and seamless response, significantly enhancing operational safety.

8. FUTURE SCOPE

Integrating advanced technologies into track surveillance systems can significantly enhance safety and operational efficiency. Combining IoT and smart signaling systems with object detection enables realtime alerts and automatic braking when individuals are detected on tracks. Expanding the detection to multiple classes—such as tools, animals, or vehicles—improves situational awareness. Incorporating thermal and infrared imaging ensures reliable performance in low-light or adverse weather conditions. Deploying optimized models on edge devices like Jetson Nano or Raspberry Pi facilitates cost-effective, remote monitoring without cloud dependency. Additionally, drone-based surveillance provides mobile, wide-area coverage, particularly in inaccessible areas. Leveraging predictive analytics on historical data further enables proactive risk assessment, identifying high-risk zones and times to enhance worker safety.

9.BIBLIOGRAPHY

1.Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934. *https://arxiv.org/abs/2004.10934*

2.Redmon, J.,&Farhadi, A.(2018).YOLOv3: AnIncremental Improvement.arXiv preprint arXiv:1804.02767. *https://arxiv.org/abs/1804.02767*

3.Jocher, G. et al. (2023). YOLO by Ultralytics [Computer software]. GitHub. *https://github.com/ultralytics/ultralytics*

4.Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). *Object Detection with Deep Learning: A Review*. IEEE Transactions on Neural Networks and Learning Systems, 30(11), 3212–3232. Indian Railways Annual Safety Report 2022–2023. Ministry of Railways, Government of India. https://indianrailways.gov.in

5.OpenCV Documentation. *Open Source Computer Vision Library*. https://docs.opencv.org

6.NVIDIA Developer. *JetsonPlatform for Edge AI*. https://developer.nvidia.com/embedded-computing

7.Lin, T. Y., Maire, M., Belongie, S., Hays, J., et al. (2014). *Microsoft COCO: Common Objects in Context*. European Conference on Computer Vision(ECCV). https://cocodataset.org

10. HELP FILE

PROJECT EXECUTION:

STEP-1: Go to start, search and launch ROBOFLOW.

STEP-2: After launching of ROBOFLOW, launch GOOGLE COLAB.

STEP-3: Open “app.py” code

STEP-4: Import all the packages and check whether error present in the code are not.

STEP-5: Create PYTHON CODE folder on DESKTOP.

STEP-6: Create the home.html file to display the home page.

STEP-7: Launch the VSCODE.

STEP-8: After launching vscode, give the path of app.py which is created in your laptop and run the program.

STEP-9: After running the app.py, then the URL is created “localhost:5000”.

STEP-10: Copy the URL and paste it in the Web Browser.

STEP-11: Then the home page of the project will be displayed.

STEP-12: In the opened home page when click on get started button it will show upload an image of railway worker on the railway track.

STEP-13: After uploading an image or video of the railway track, it will predict whether a worker is present on the track or not.