Traffic Safety: Helmet Detection System Documentation

Project Overview

This computer vision system detects motorcyclists and checks for helmet compliance using YOLOv8s. It processes images/videos, identifies violations (no helmet), and generates marked output files. The system is trained on custom datasets and handles real-world traffic scenarios.

Dataset Preparation

Data Source: Roboflow project "Helmate Detection" (3,053 images)

Classes: helmet, license\_plate, motorcyclist

Preprocessing:

- Auto-oriented all images

- Resized to 640x640 (stretched)

- Augmentations applied:

- 2 variants per original image

- Random crops (0-20% zoom)

- Brightness adjustment (±15%)

Dataset Split:

- Training: 2,137 images (70%)

- Validation: 611 images (20%)

- Testing: 305 images (10%)

Model Training

Training Command:

from ultralytics import YOLO

model = YOLO("yolov8s.pt")

model.train(

data="helmet\_dataset.yaml",

epochs=100,

imgsz=640,

batch=16,

lr0=0.01,

augment=True

)

Training Details:

- Base model: YOLOv8s

- Input size: 640x640

- Augmentation: Enabled

- Training time: ~4 hours (NVIDIA T4 GPU)

Performance Metrics:

- mAP@0.5: 0.892

- Precision: 0.87

- Recall: 0.85

Detection Logic

Processing Steps:

1. Detect motorcyclists (confidence threshold: 0.6)

2. Check helmet presence in upper body region (IoU threshold: 0.4)

3. Mark violations:

- Red box + "NO HELMET" label for offenders

- Green box for compliant riders

4. Optional license plate detection (blue boxes)

System Usage

Installation:

pip install ultralytics opencv-python numpy

Image Processing:

detector.process\_image("input.jpg", "output.jpg")

Video Processing:

detector.process\_video("input.mp4", "output.mp4", fps=30)

Hardware Requirements:

- Development: CPU i5, 8GB RAM

- Real-time: NVIDIA T4 GPU

- Edge devices: Jetson Nano 4GB

Future Improvements

Planned Enhancements:

1. Advanced tracking with unique IDs

2. Traffic system integration for e-challans

3. Model optimization for edge devices

4. Dashboard for monitoring violations

Troubleshooting

Common Issues:

1. Low accuracy: Retrain with more data or adjust thresholds

2. Slow performance: Reduce resolution or enable GPU

3. False positives: Increase IoU threshold or add negative samples

This documentation provides complete implementation details from dataset preparation to deployment. The system is ready for integration with traffic monitoring infrastructure and can be adapted for various use cases. For additional support, refer to the source code comments or dataset documentation on Roboflow.