# Model Training Report

## 1. Code Flow / Algorithm

1. Load YOLOv11m pretrained model (`yolo11m.pt`)

2. Load and parse dataset defined in `data.yaml`

3. Set training configurations: `epochs=50`, `batch size=8`, `device=GPU`

4. Train the model on 5048 images

5. Validate the model on 561 images

6. Evaluate model metrics and save best-performing weights

## 2. Exploratory Data Analysis (EDA)

Dataset Summary:

- Training Images: 5048

- Validation/Test Images: 561

Classes:

- ADVISORY_SPEED_MPH: 188 instances

- DIRECTIONAL_ARROW_AUXILIARY: 199 instances

- DO_NOT_ENTER: 220 instances

## 3. Training Configuration

- Model: YOLOv11m

- Epochs: 50

- Batch Size: 8

- Patience: 0 (no early stopping)

- Device: GPU (Google Colab)

## Why Only 50 Epochs?

The dataset consists of a substantial number of labeled images (5,609 in total). Training on such a large dataset requires high-performance hardware. Since training was conducted on Google Colab, which has time and resource limitations, we capped the training at 50 epochs. Despite this limitation, the model demonstrated exceptional performance, achieving high

accuracy and generalization. Therefore, extending the epochs further was not necessary for this task.

## 4. Training Results Analysis

Validation Metrics:

- mAP@0.5: 0.979

- mAP@0.5:0.95: 0.871

- Precision: 0.947

- Recall: 0.959

Class-wise Performance:

| Class | Precision | Recall | mAP@0.5 | mAP@0.5:0.95 |
|---|---|---|---|---|
| ADVISORY_SPEED_MPH | 0.975 | 0.995 | 0.994 | 0.915 |
| DIRECTIONAL_ARROW_AUXILIARY | 0.930 | 0.905 | 0.958 | 0.775 |
| DO_NOT_ENTER | 0.936 | 0.977 | 0.986 | 0.923 |

Inference Speed:

- Preprocess: 0.3 ms/image

- Inference: 7.5 ms/image

- Postprocess: 3.1 ms/image

## 5. Confusion Matrix Analysis

- ADVISORY_SPEED_MPH: 187 correct, 9 missed

- DIRECTIONAL_ARROW_AUXILIARY: 188 correct, 29 missed

- DO_NOT_ENTER: 218 correct, 31 missed

- 14 false positives from background

## 6. Detection on Unseen Data

Performs reliably across various sizes and lighting. Cluttered background is the major error source.

## 7. Tracking Overview

Integrated real-time object tracking using YOLOv11 with selectable trackers:

- **ByteTrack**: Fast and accurate, ideal for simple scenes. Doesn't handle occlusion or re-ID.
- **BoT-SORT**: Supports re-identification and occlusion handling using Kalman Filter + appearance features.

**False Tracking (ID Switching)**

Occurs when:

- Objects cross paths or overlap.
- Rapid motion or occlusion occurs.

**Solution:** Use **BoT-SORT** for better ID persistence or fine-tune tracker's appearance model.

## 8. Tracker Comparison

| Feature | ByteTrack | BoT-SORT |
|---|---|---|
| Accuracy | High in sparse scenes | Higher in dense, crowded scenes |
| Speed | Faster, more lightweight | Slightly slower due to re-ID and Kalman |
| Re-identification | ❌ Not supported | ✅ Supported |
| Motion Model | Simple IoU-based matching | Kalman Filter + Deep Re-ID |
| Occlusion Handling | Basic | Advanced (better ID persistence) |
| Best Use Case | Simple, low-occlusion videos | Crowded scenes with occlusion and reappearance |
| Config Complexity | Very low | Moderate |

## Conclusion

The high-precision, low-latency YOLOv11m model was trained on our custom dataset, making it suitable for real-time object detection and tracking.