Autonomous Vehicle Development

Problem & Its Solutions

Traffic Congestion

➤ **Problem:** More people in cities means more traffic.

➤ **Solution:** Our system watches traffic in real time to control lights better and reduce traffic lights.

Accidents Due to Human Error

➤ **Problem:** Many accidents happen because people are tired, not paying attention, or make bad decisions.

➤ **Solution:** Automated detection of lights, signs, lanes, and pedestrians reduces human reliance and improves road safety.

Lack of Real-Time Traffic Analysis

➤ **Problem:** Current systems struggle to quickly react to traffic changes.

➤ **Solution:** Our real-time deep learning models provide immediate detection and alert capabilities.

Difficulty in Detecting Multiple Traffic Elements

➤ **Problem:** Identifying various traffic components in complex scenes is challenging.

➤ **Solution:** Our multi-task model simultaneously detects traffic lights, signs, vehicles, lanes, and pedestrians for full scene understanding.

Project Goals

- Traffic sign detection and classification
- Traffic light detection and state determination
- Road and lane segmentation
- Pedestrian and vehicle detection
- Real-time decision-making mechanism
- Video processing and result visualization

Project Summary

This project aims to develop an AI-based image processing and decision-making system for autonomous vehicle systems.
The system includes fundamental features such as traffic sign detection, traffic light detection, road and lane segmentation, and pedestrian and vehicle detection.

Technologies Used

- YOLOv8
- PyTorch
- Ultralytics YOLO
- Google Colab for GPU
- Python
- OpenCV

Lane Segmentation

Dataset Information

Total images: 212

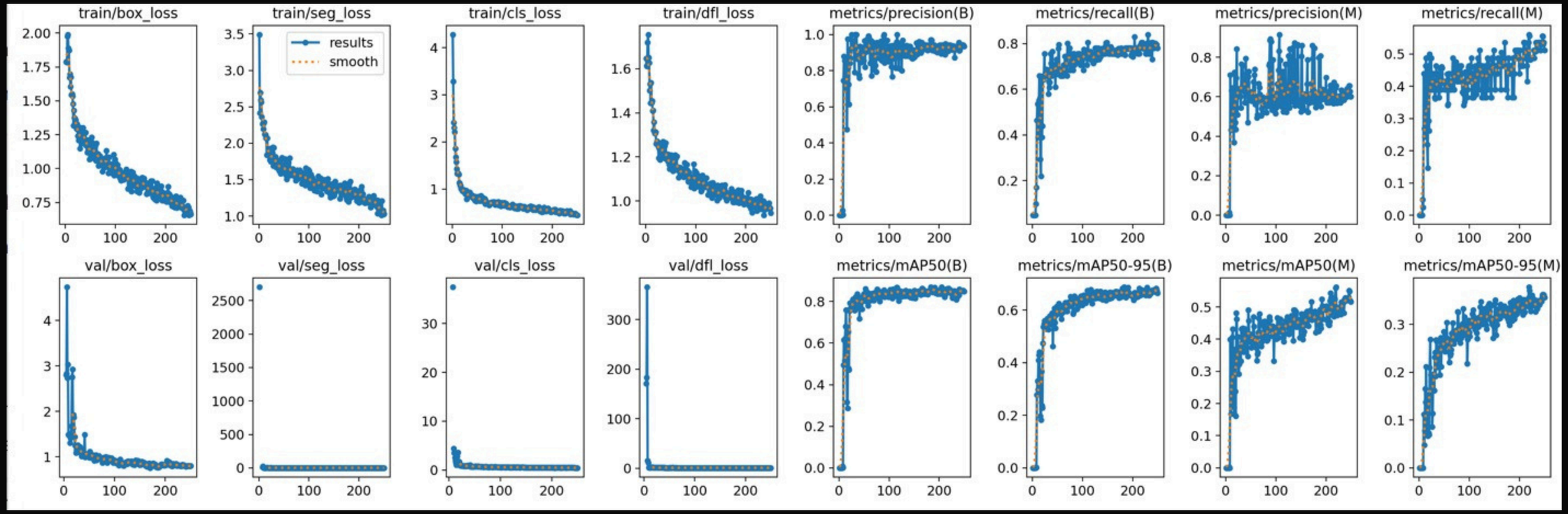
Train set: 190 images (90%)

Validation set: 22 images (10%)

Model Used

We used transfer learning with the yolov8l-seg.pt model and trained it on our own lane data to make it work for our task.

Results



Loss Functions

Box, segmentation, classification, and DFL losses consistently decreased during training. Validation losses also decreased in a similar trend indicates no overfitting occurred.

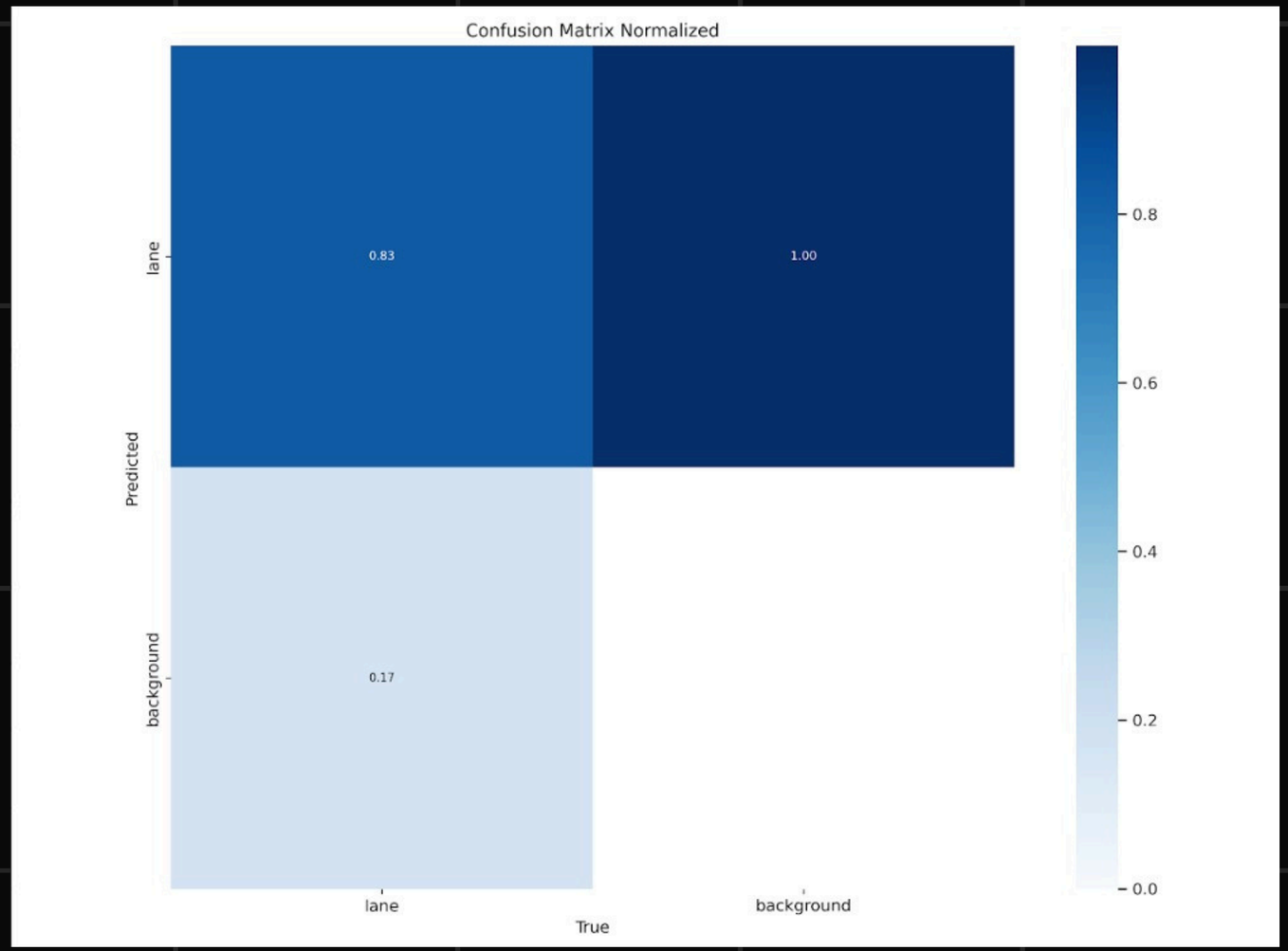
Performance Metrics

Precision and Recall got better over time, showing the model makes correct and accurate predictions. The higher mAP50 and mAP50-95 scores mean the model works well at different confidence levels.

Interpretation

The results show that our model was successfully trained using transfer learning and performs well on our task.

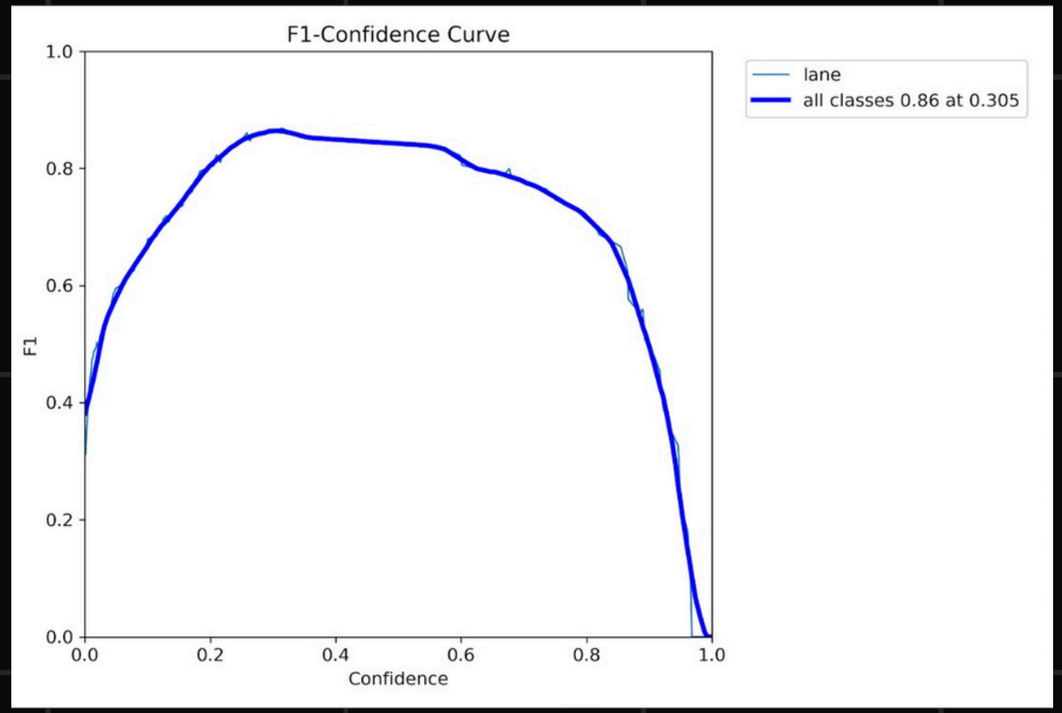
Confusion Matrix Normalized



- The model correctly classified 83% of lane pixels.
- It achieved 100% accuracy on background pixels.
- 17% of lane pixels were misclassified as background.

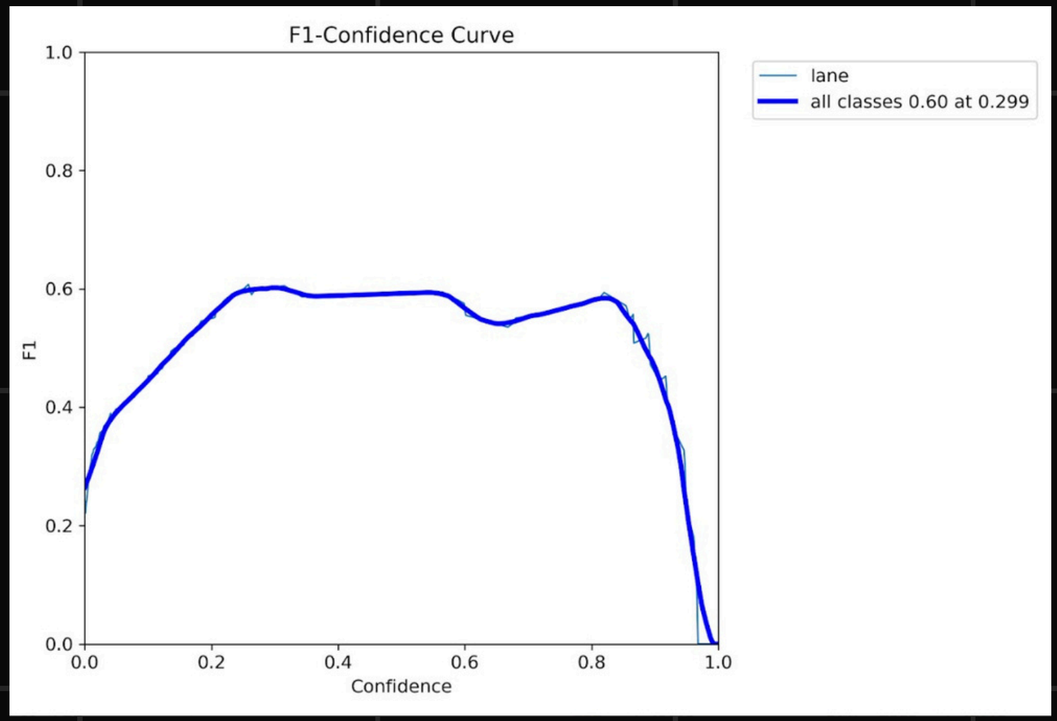
The model detects the background well, but lane segmentation still needs some improvement.

BoxF1_curve



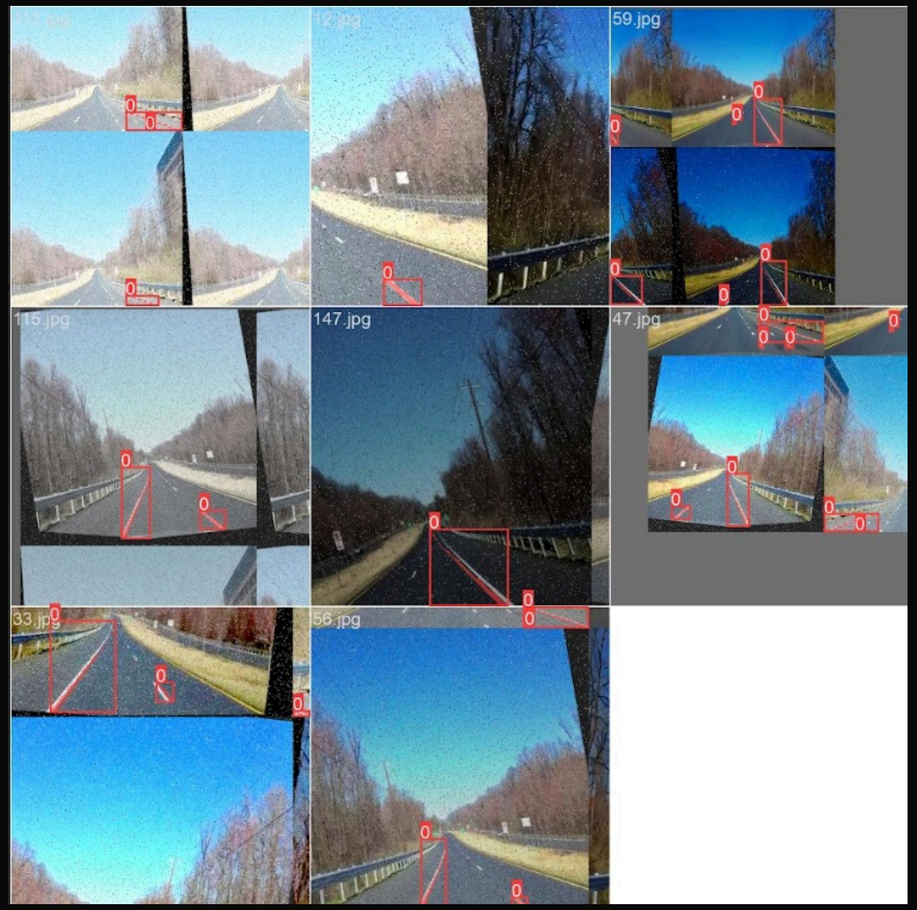
F1-score for object detection using bounding boxes.
The highest F1 score is 0.86, which was reached at around 30% confidence level.
This means our model performs very well in detecting lane objects.

MaskF1_curve

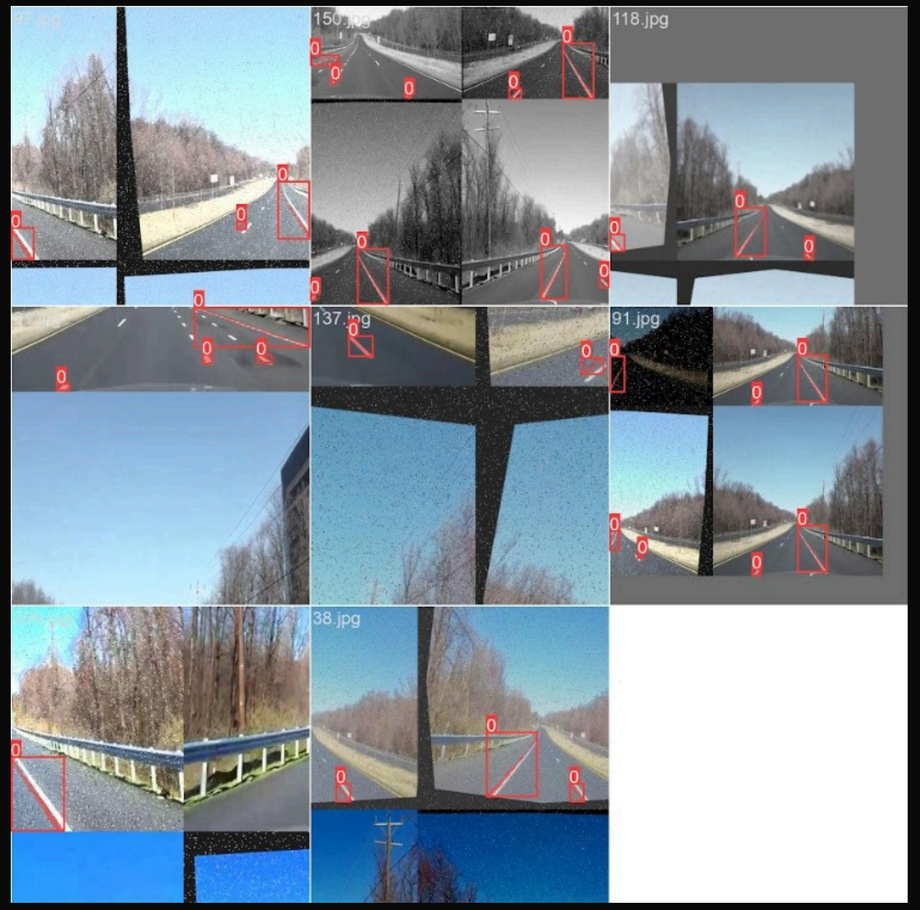


F1-score for segmentation masks.
The best F1 score here is 0.60, also around 30% confidence.
Segmentation is a more difficult task, but the result is still acceptable.

train_batch0



train_batch1



Examples from the first two batches (batch0 and batch1) used during model training.
To improve model generalization, we applied several data augmentation techniques.
These include rotation, cropping, converting to grayscale, and adding noise.
The model learns to handle different lighting, views, and noisy images.

Road Segmentation

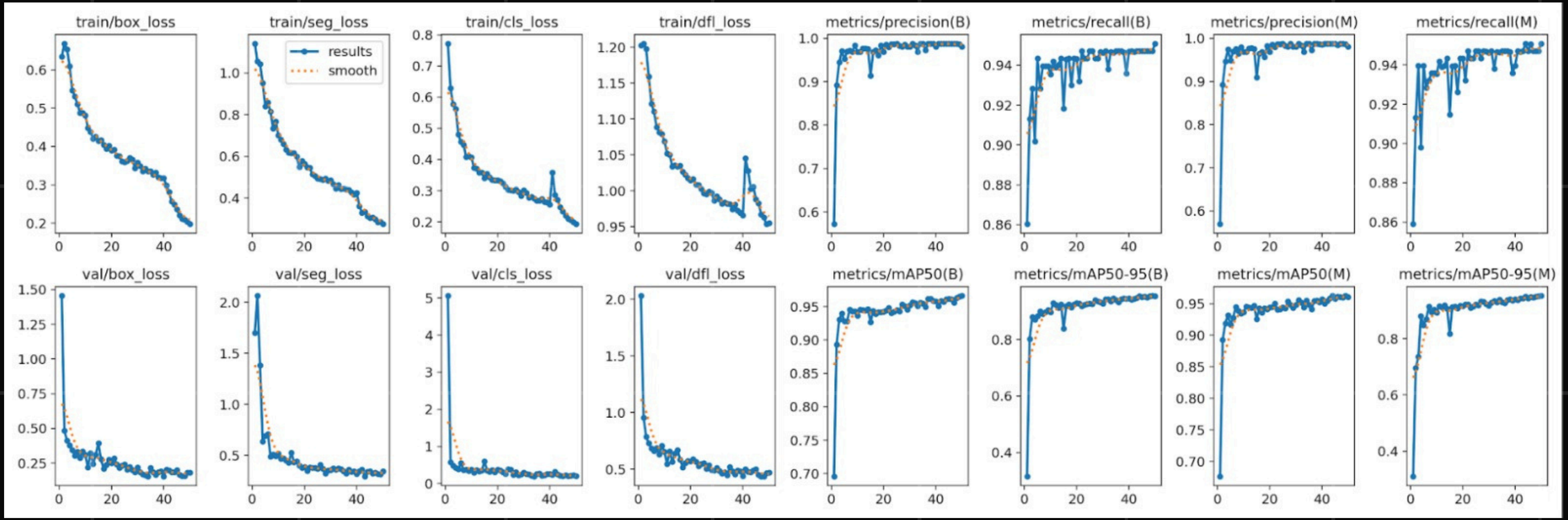
Dataset Information

- Total images: 2806
- Training set: 2525 images (90%)
- Validation set: 281 images (10%)

Model Used

We applied transfer learning using the yolov8l-seg.pt model and trained it on our custom road segmentation dataset to adapt the model to our specific task.

Results



Loss Curves

Losses for box, segmentation, classification, and DFL steadily decreased.
Validation losses followed a similar trend indicates no overfitting.
The model successfully learned to generalize well.

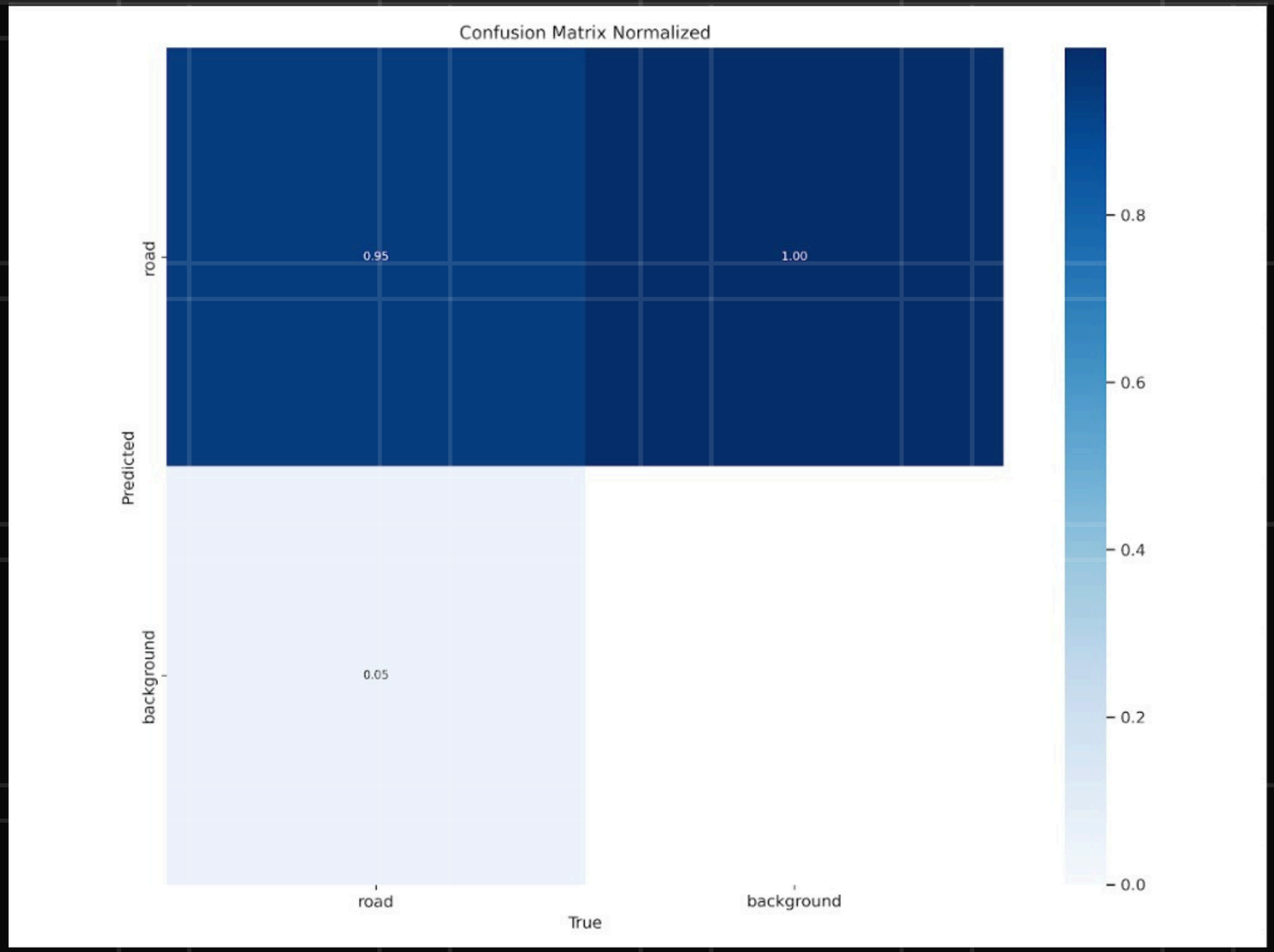
Performance Metrics

Precision and Recall reached above 95% for both bounding boxes and masks.
mAP50 and mAP50-95 values are also high, showing strong prediction quality.
Segmentation performance is particularly reliable.

Interpretation

The model performs very well in both object localization and segmentation. Training is stable and validation confirms high model accuracy.

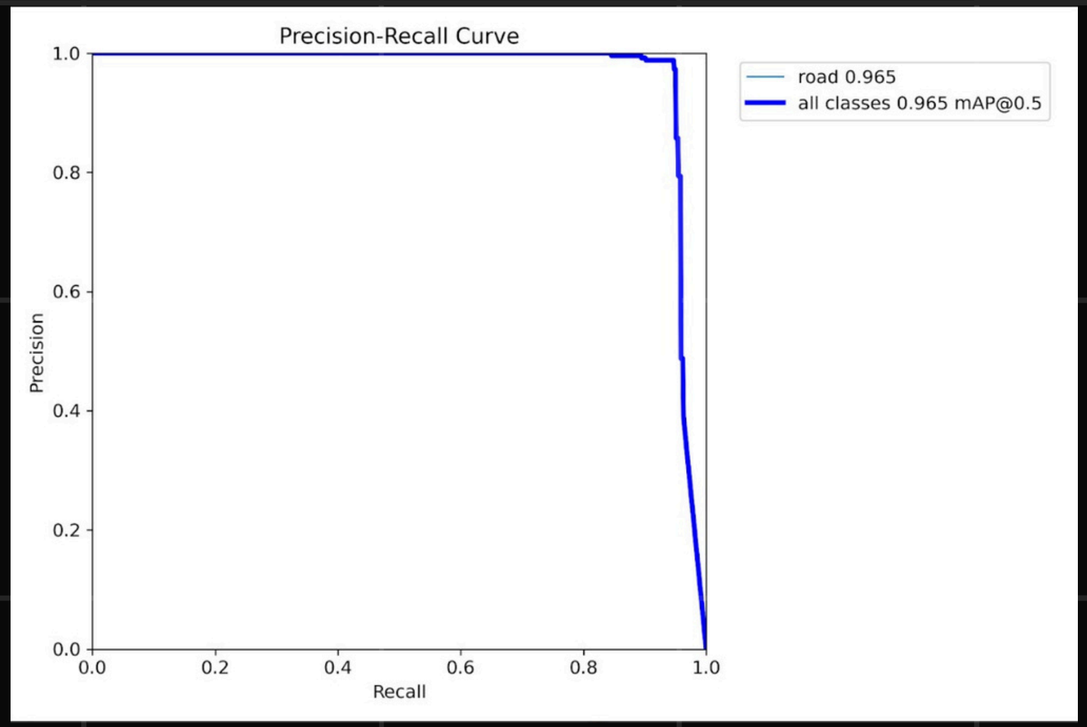
Confusion Matrix Normalized



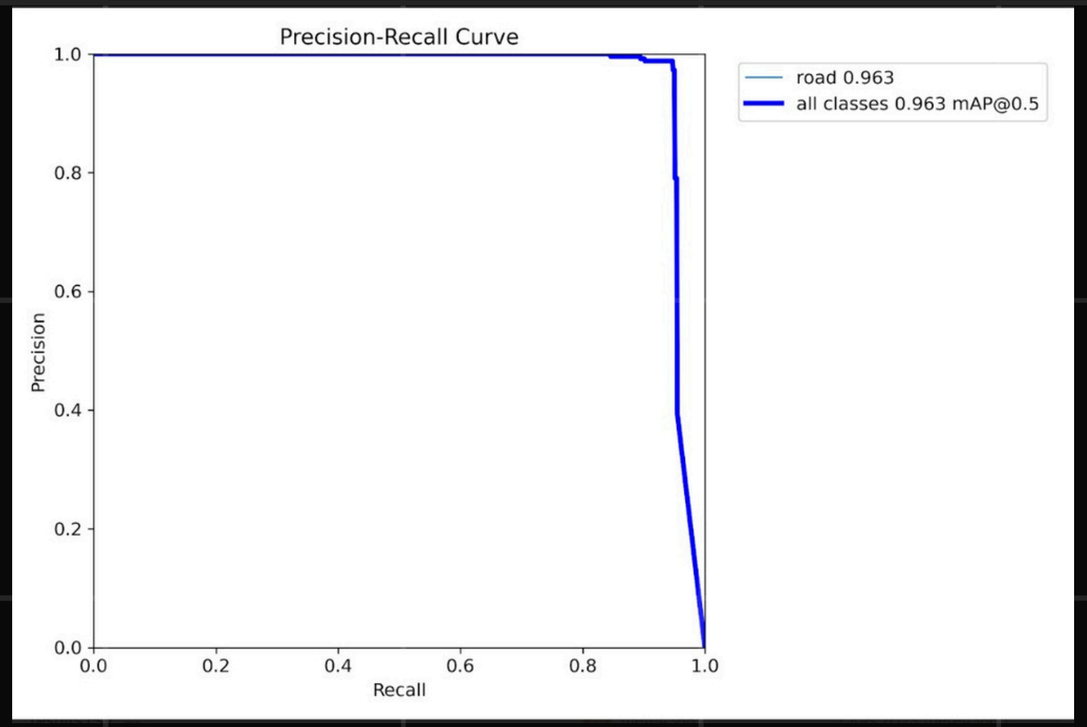
- The model correctly classified 95% of road pixels.
- 100% accuracy was achieved for background pixels.
- The error rate is very low (about 5%).

This means the model can tell the difference between road and background very well, even when the image is hard to understand.

BoxPR_curve



MaskPR_curve



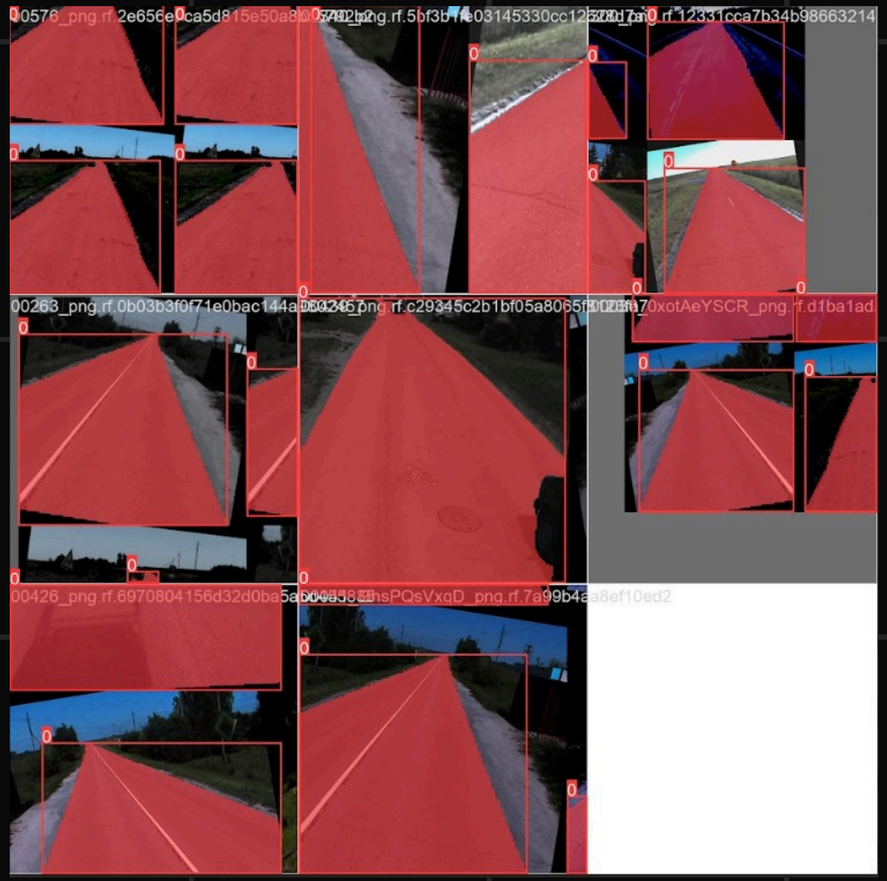
The model reached a mAP@0.5 score of 0.965, which shows very high accuracy in detecting road regions using boxes.

We see the PR curve for segmentation masks.

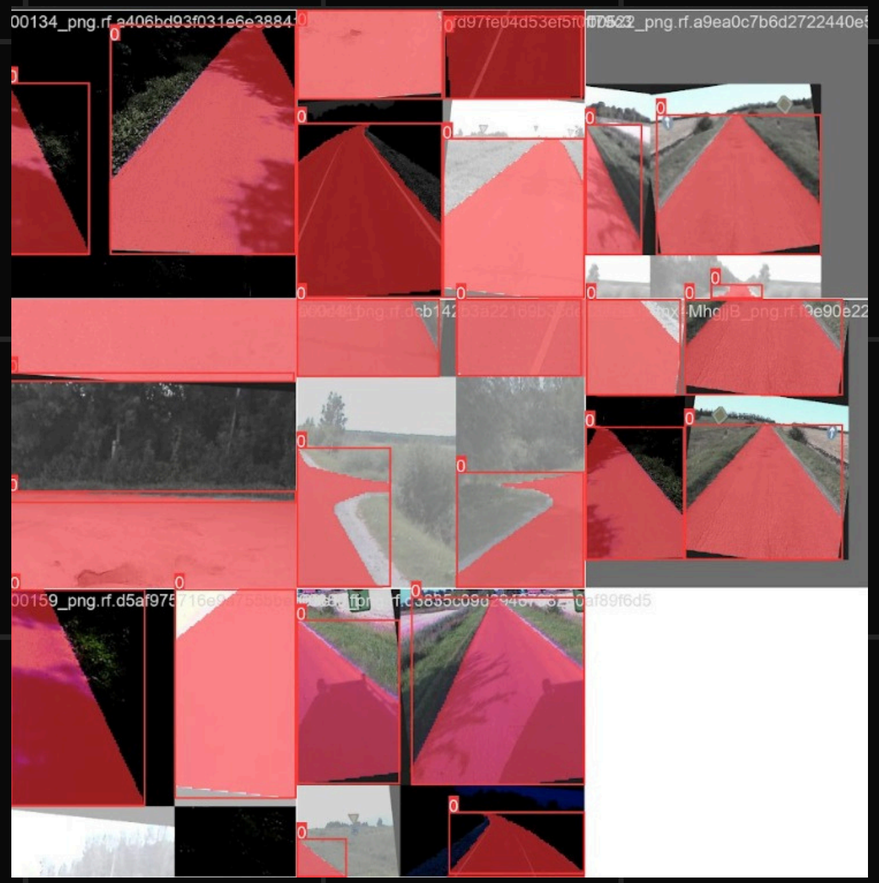
Here too, the model achieved a mAP@0.5 score of 0.963, indicating strong performance in mask-based road segmentation.

The curves are very close to the top-right corner, which means the model gives both precise and confident predictions.

train_batch0



train_batch1



These images show the first two training batches (batch0, batch1) used during training. The red areas are the ground truth masks used for road segmentation. These masks tell the model which parts of the image are road during training. You can also see that the images have different colors, rotations, and styles. These are caused by data augmentation techniques like rotation, brightness change, and grayscale conversion, which help improve model generalization.In short, this step ensures that the model learns to detect roads in many different visual conditions.

Traffic Light

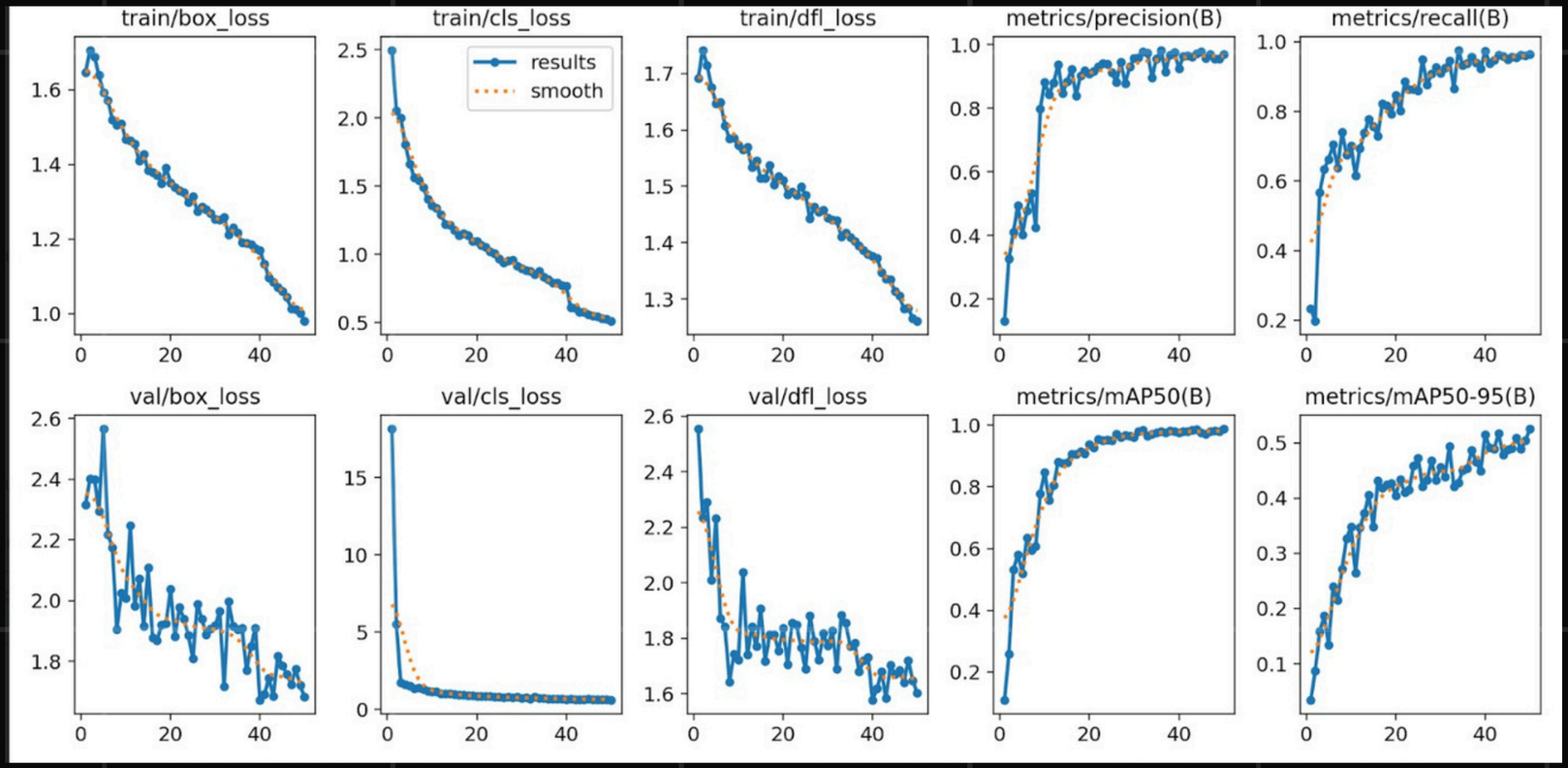
Dataset Info

- Total images: 2397
- Training set: 2157 images (90%)
- Validation set: 240 images (10%)

Model Used

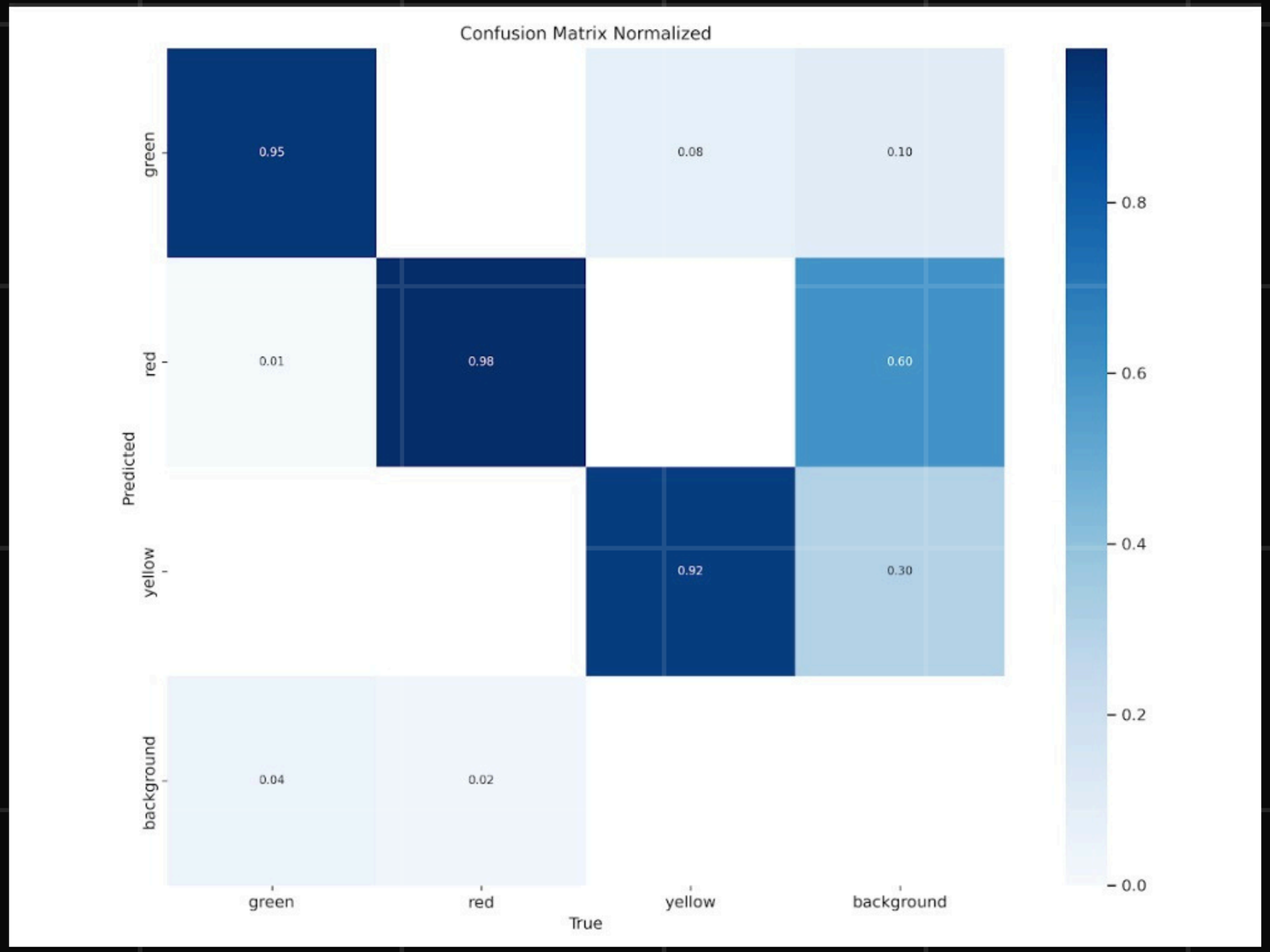
We applied transfer learning using the yolov8l.pt model and trained it on our custom Traffic Light dataset to adapt the model to our specific task.

Result



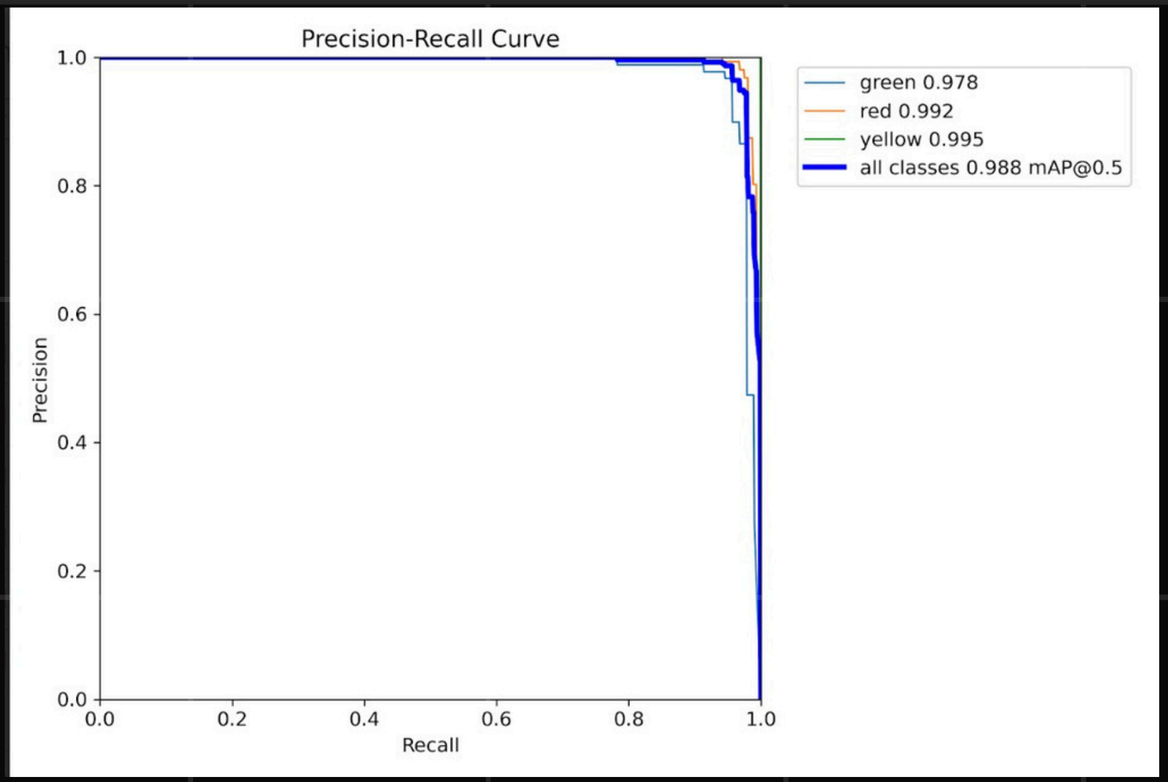
- During training, losses such as box_loss, cls_loss, and dfl_loss decreased steadily.
- Precision, Recall, and mAP metrics improved over time, showing accurate learning.
- No major gap between training and validation losses → no overfitting detected.

Confusion Matrix Normalized

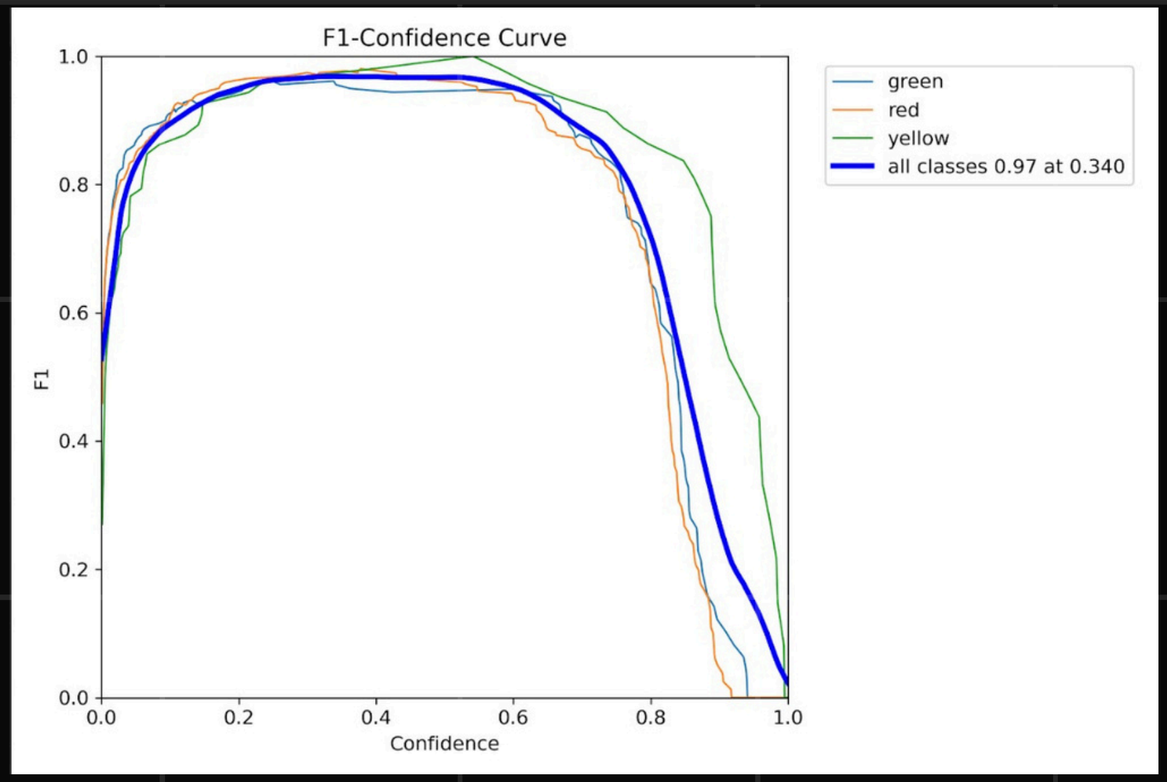


- The model classified traffic light colors (green, red, yellow) with high accuracy.
- Best performance was seen in red (0.98) and green (0.95) predictions.
- There is some confusion with yellow, often misclassified or detected as background.

PR_Curve



F1 Curve



- PR Curve shows high-precision predictions across all classes (mAP@0.5 ≈ 0.988).
- F1 Curve indicates a peak F1 score of ~0.97 around 34% confidence.
- Smooth and stable curves show strong model generalization and balanced prediction.

train_batch0

train_batch1

- Data augmentation techniques like rotation, color shift, and flips are applied.
- Red boxes show the correct labeling of traffic lights, ensuring supervised learning.

Traffic Sign Detection

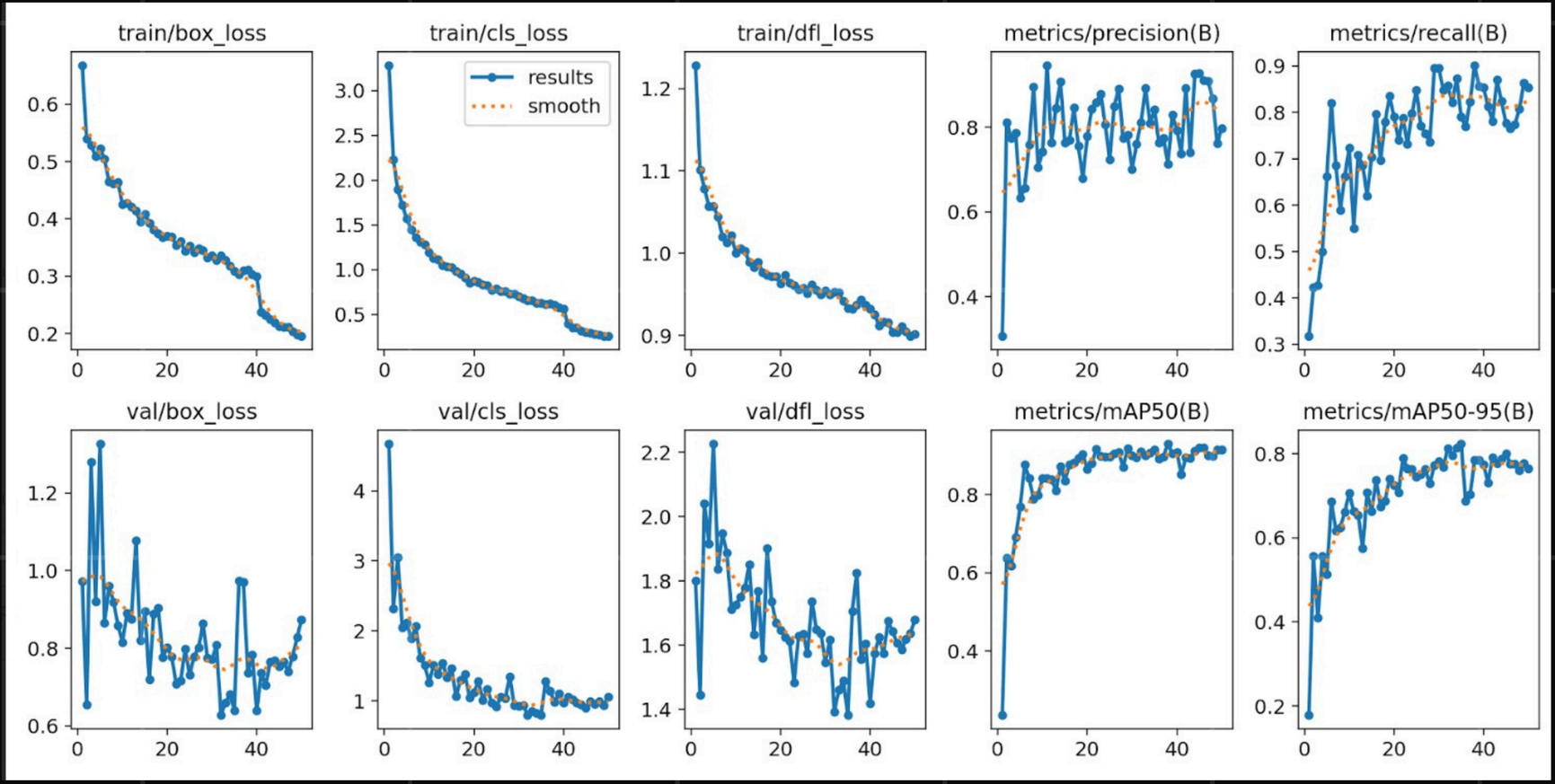
Dataset Information

- Total images: 763
- Training set: 686 images (90%)
- Validation set: 77 images (10%)

Model Used

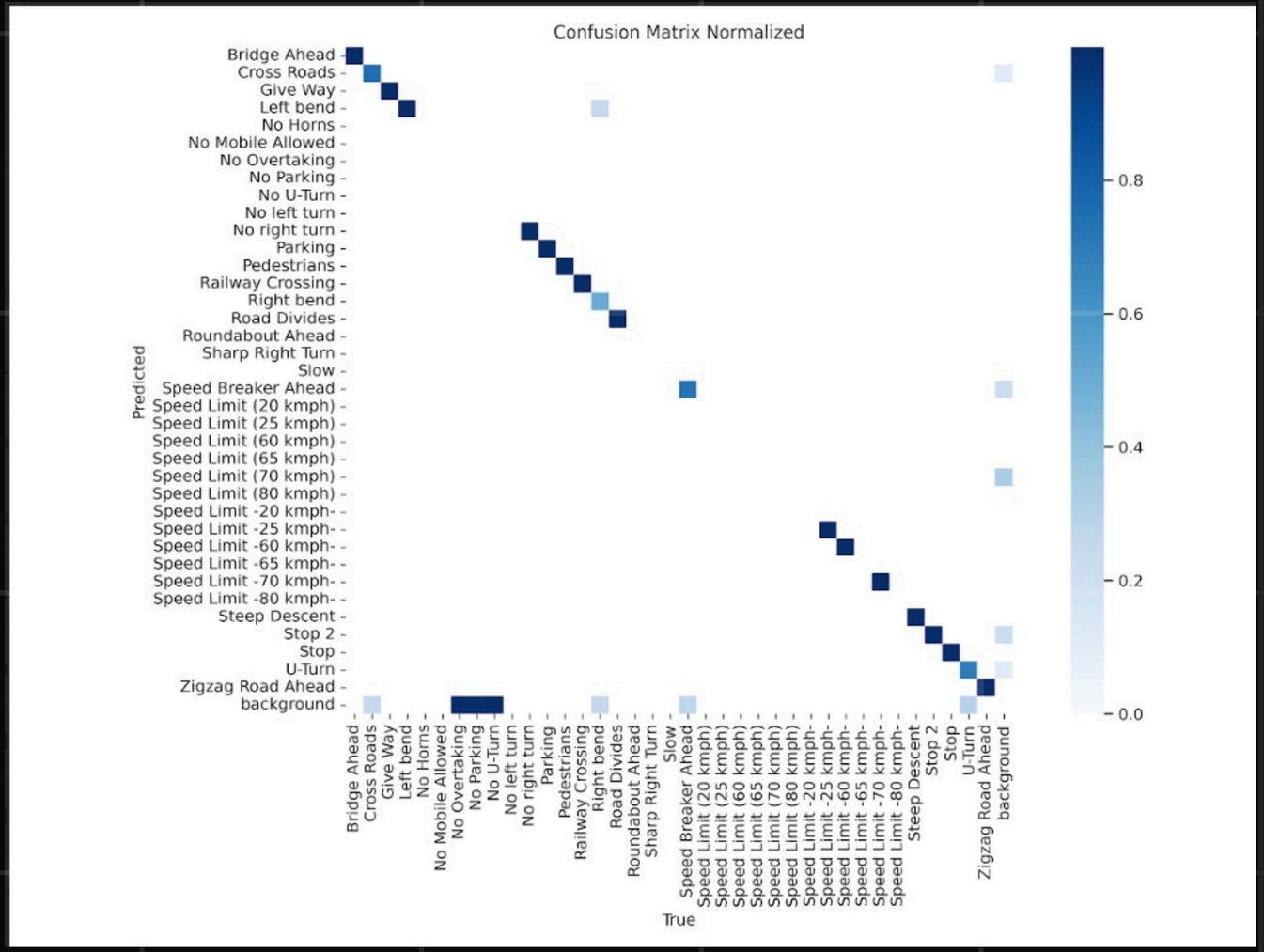
We applied transfer learning using the yolov8l.pt model and trained it on our custom Traffic Sign dataset to adapt the model to our specific task.

Result



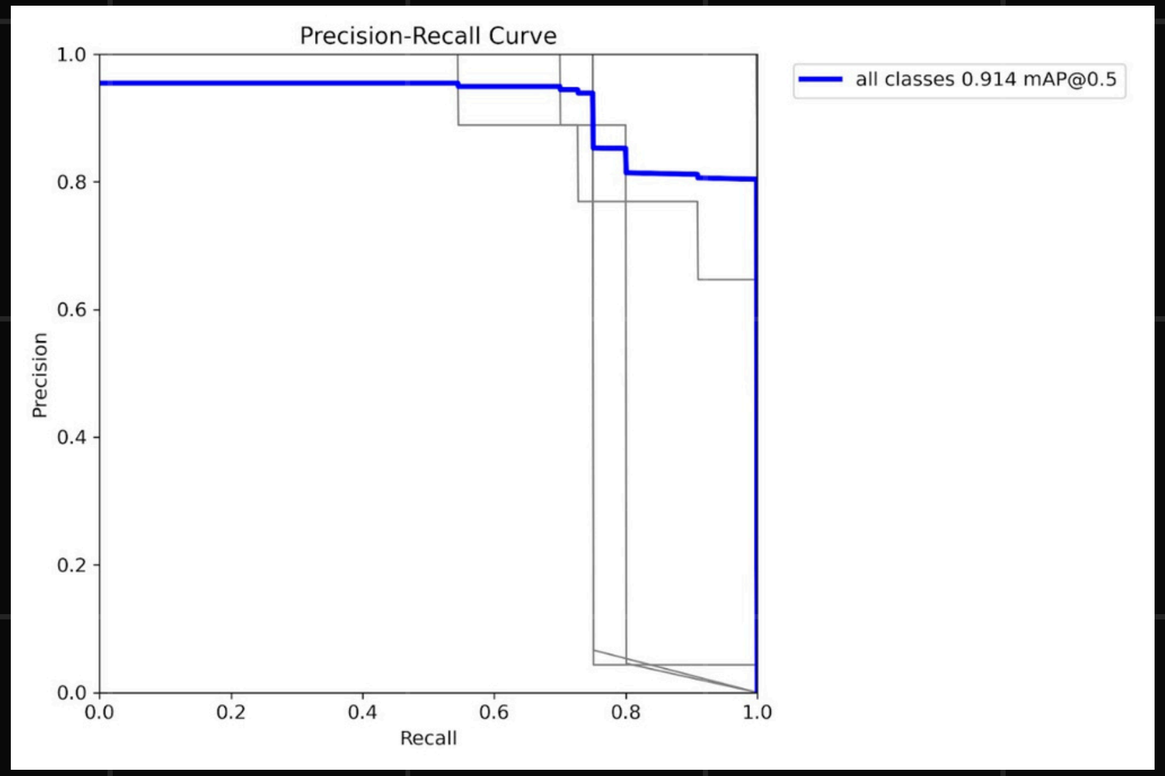
- Box, classification, and DFL losses decreased steadily, showing good model learning.
- Validation metrics also improved consistently.
- Precision, Recall, and mAP reached high, stable values by the end of training.

Confusion Matrix Normalized

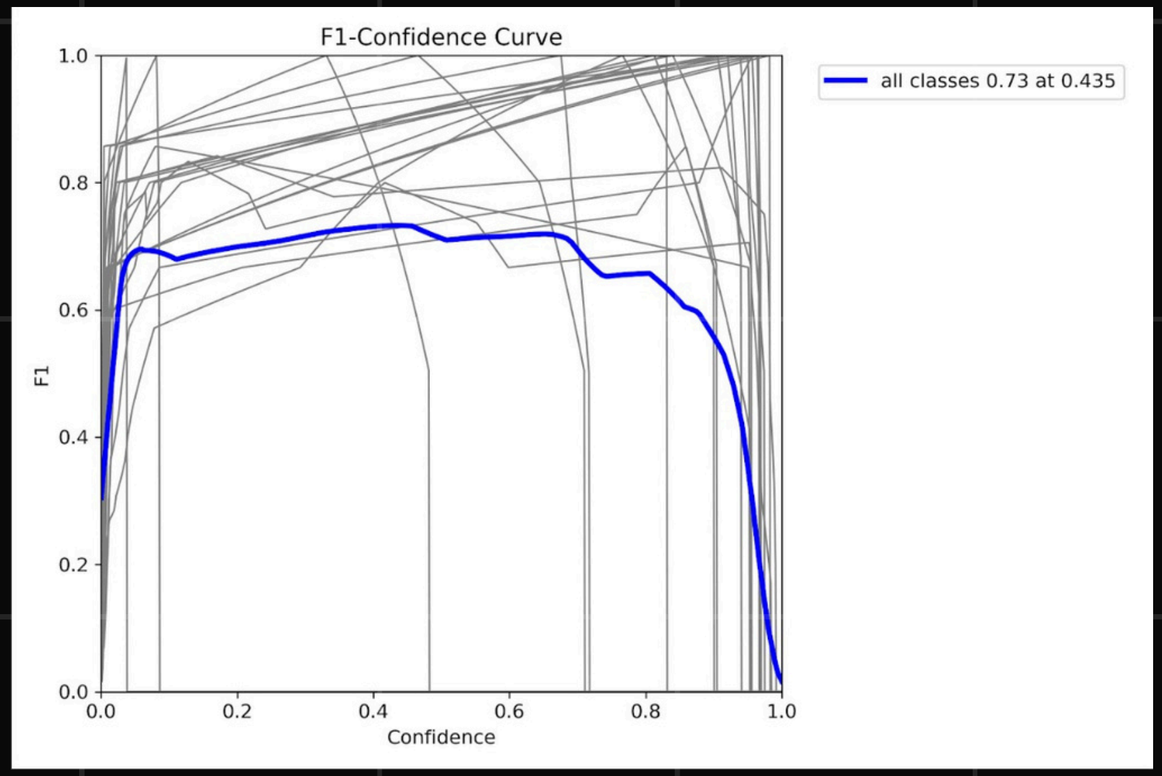


- On this multi-class traffic sign dataset, most signs were classified correctly.
- Some confusion occurred between similar classes (e.g., speed limits).
- Overall, the model demonstrated strong class separation performance.

PR_Curve



F1 Curve



- PR Curve: Average mAP@0.5 is 91.4%, indicating high model performance.
- F1 Curve: Peak F1 score is 0.73, achieved at ~43% confidence.
- Curves reflect the model's balanced and stable predictions across classes.

train_batch0

train_batch1

- Shows data augmentation samples applied during training: rotation, lighting changes, etc.
- Labels are properly placed across diverse images.
- This variety helps improve the model’s generalization ability.

Person- Car Detection

1. Project Objective

“In this Model, our goal was to detect key traffic-related objects such as persons, cars, buses, and trucks in video streams — accurately and in real time — using the YOLOv8 object detection model.”

2. Script Logic – detect.py

- The yolov8n.pt model is loaded using the Ultralytics API.
- Detection is restricted to specific classes: person (0), car (2), bus (5), truck (7).

For each frame:

objects are detected,

bounding boxes and confidence scores are drawn,

the frame’s FPS is calculated and overlaid.

- All processed frames are saved into an .avi output video.

3. Why This Design?

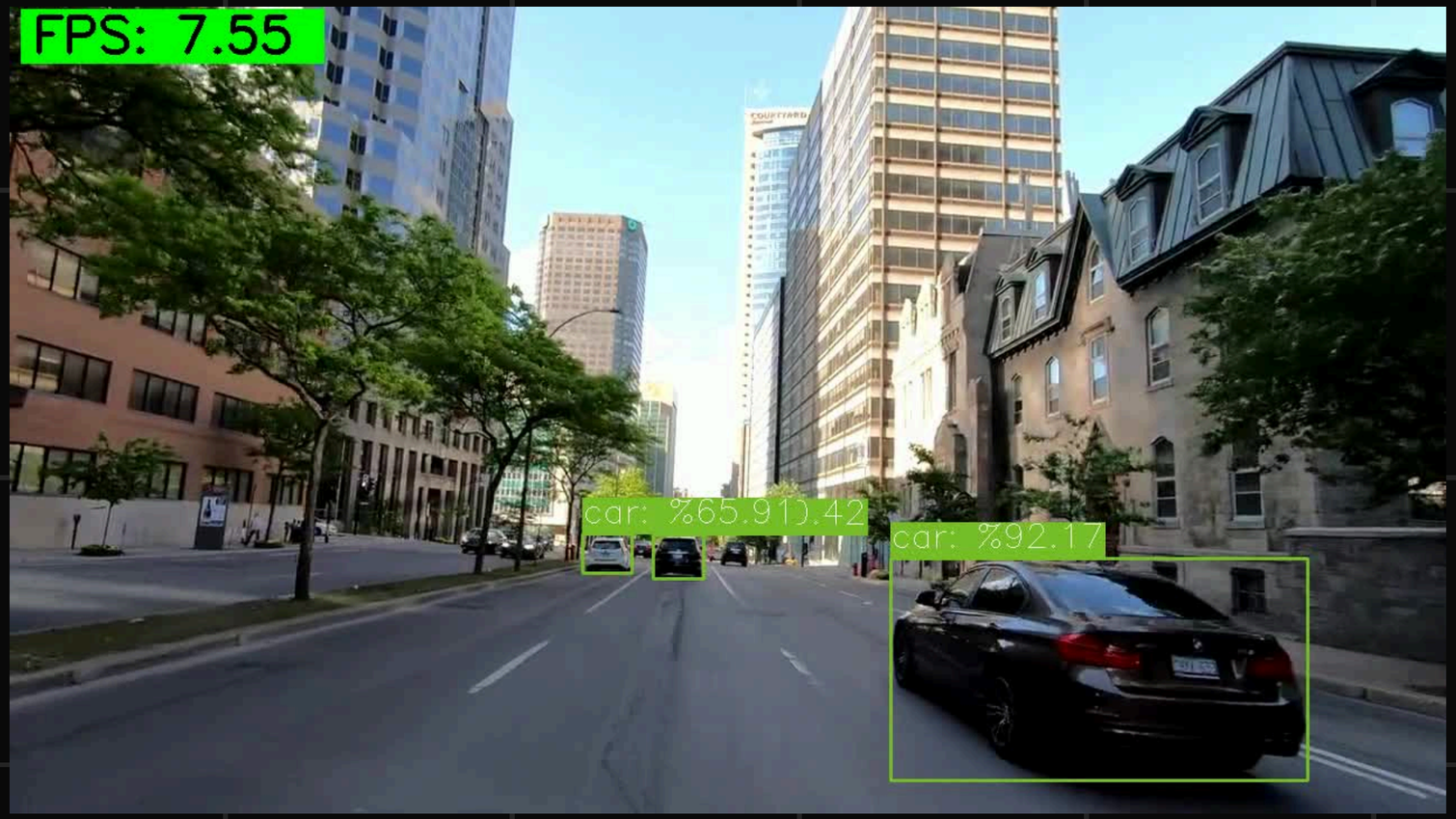
“We chose this approach because it is:”

Modular → easy to maintain and scale.

Lightweight & fast → YOLOv8n runs in real time.

Focused → only processes the classes we actually care about.

Output



Real-time detection output.

- The model successfully detects cars with high confidence scores and overlays class names and bounding boxes on each frame.
- The achieved frame rate (FPS: 8.11) indicates the system runs close to real time, even with full HD video input.

Model Architecture and Integration

Architectural Approach

The Project uses an "Ensemble Learning" approach. A separate YOLO model was trained for each task, and these models run in parallel. This approach was preferred over "Multi-Task Learning" because:

1. Specialized model performance is higher for each task
2. Models can be updated independently
3. Easier debugging and development process

Model Integration

The models are integrated as follows:

1. Parallel Processing:
- All models run in parallel for each video frame
- Each model processes in its area of expertise (traffic signs, lights, etc.)

Decision Making Mechanism

Outputs from all models are combined in the `make_decision()` function
- Decisions are made based on priority order (e.g., red light > pedestrian detection)
The system makes decisions based on the following conditions:
- Red light detection → "STOP"
- STOP sign detection → "STOP"
- Pedestrian detection → "SLOW DOWN"
- Other conditions → "GO

Alternative Approaches Tried

Multi-Task Learning Attempt

- Attempted to solve all tasks with a single YOLO model
- Result: Low performance and complex training process
- Not preferred

Sequential Processing

Tried running models sequentially
- Result: Slow processing time
- Not preferred

Alternative Approaches Tried

Advantages of Chosen Approach

1.M**odularity**: Each model can be updated independently
2. **Performance**: Specialized model for each task
3. **Scalability**: New models can be easily added
4. **Fault Tolerance:** Errors in one model don't affect others

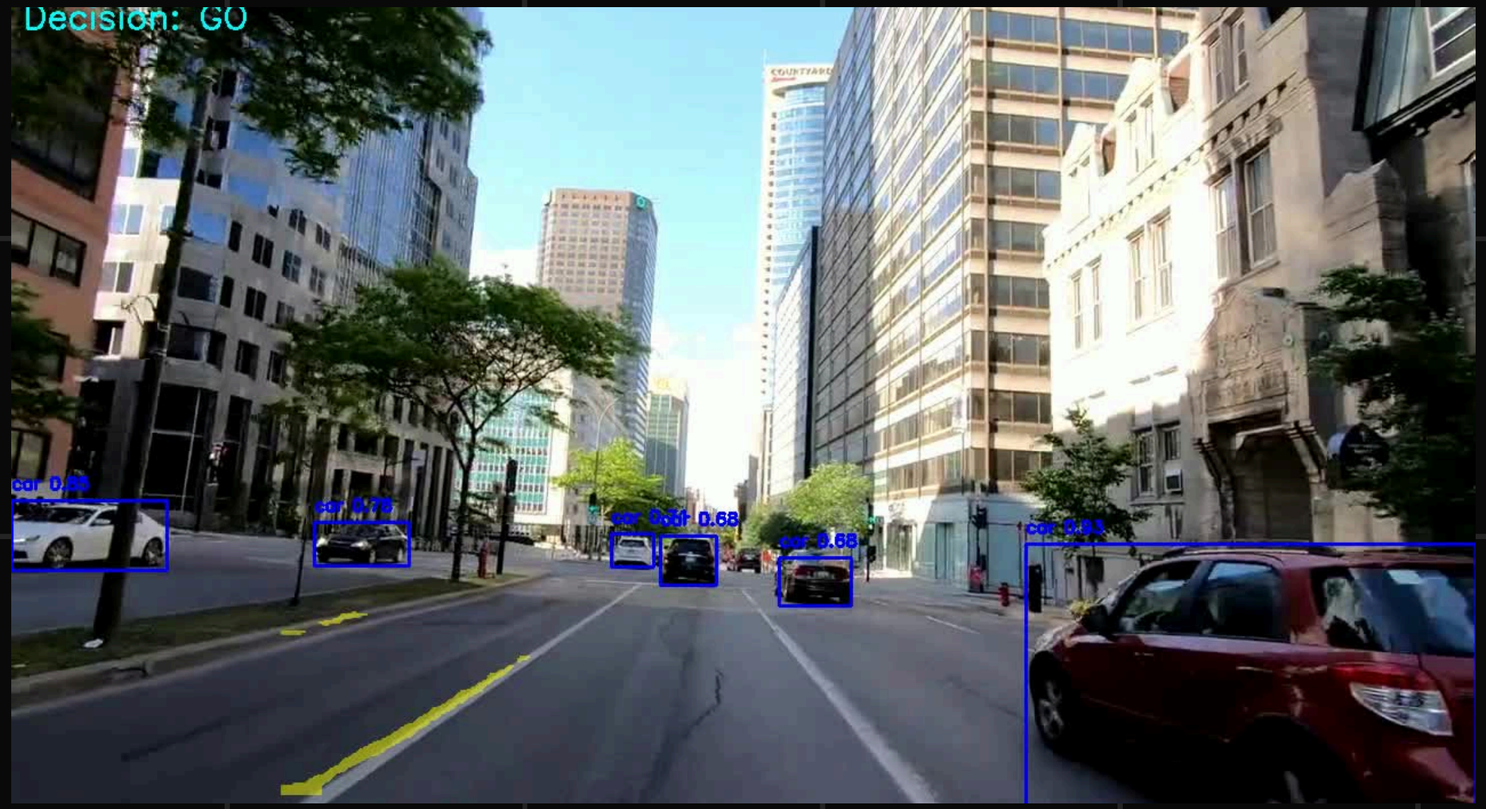
Disadvantages of Chosen Approach

1. **Memory Usage**: Each model is loaded separately

2. **Processing Time**: All models need to be run

3. **Synchronization:** Inter-model synchronization required

Result of All Models Trained



- Here you can see a sample output from our autonomous vehicle perception system
- At the top left, the system's current driving decision is displayed — in this case, 'SLOW DOWN', meaning the system has detected a situation that requires reduced speed, such as the presence of vehicles or pedestrians ahead.>
- All detections and decisions are performed in real time, and the results are visualized directly on the video.

Future Improvements

- Improve model performance
- Increase real-time processing speed
- Add more traffic signs and conditions
- Add depth estimation
- Add object tracking feature
- Increase safety checks for error conditions

Thank You!

Zeynep Hanife ÇELİKOĞLU

Ömer PANAY

Ramazan EDİZ