YOLOPilot

Self driving through YOLOv8 and Transfer learning

Presented by: Mohammad Reza Azadi Tinat

Project Definition

Objective

To develop an object detection system for autonomous driving using YOLOv8, capable of identifying key objects such as:

- Motorbikes
- Vehicles
- Pedestrians
- Traffic Lights

Core Idea

Train and evaluate a YOLO-based deep learning model on two different real-world datasets, applying both training from scratch and transfer learning to compare performance and efficiency.

Goals

- Accurately detect traffic-relevant objects in road scenes
- Compare different training strategies (e.g., full training vs. transfer learning)
- Build an interactive **Gradio-based app** for testing detection

Application steps

YOLOPilot

1. Dataset Preparation

- Two datasets were organized into the YOLOv8 format: train/images, train/labels, etc.
- Label files were cleaned and converted for consistency with a unified class schema.

2. Initial Training on Dataset 1

- A YOLOv8 base model (yolov8s.pt) was trained on the first dataset.
- Performance was evaluated using F1 scores and confusion matrices.

3. Error Analysis

- Detected mismatches between predictions and ground truth.
- Visualized incorrect detections to identify model limitations.

4. Transfer Learning on Dataset 2

- Best weights from Dataset 1 were used to fine-tune on the second dataset.
- Experiments included resized and padded image inputs.

5. Evaluation and Visualization

- Metrics, confusion matrices, and sample results were collected.
- Validated model generalization and identified areas for improvement.

```
YOLOPilot/
├── first_dataset/
│ ├── train
│ │ ├── images
│ │ ├── lables_old
│ │ └── lables
│ ├── valid
│ │ ├── images
│ │ ├── lables_old
│ │ └── lables
│ ├── test
│ │ ├── images
│ │ ├── lables_old
│ │ └── lables
│ ├──label_modify.py
│ └── runs → Results after training
│
├── second_dataset/
│ ├── train(Resized), train_orginal
│ │ ├── images
│ │ └── lables
│ ├── val(Resized), val_orginal
│ │ ├── images
│ │ └── lables
│ ├── test(Resized), test_orginal
│ │ ├── images
│ │ └── lables
│ ├── labels_train.csv, labels_val.csv
│ ├── new_labels
│ ├── images_resized
│ ├── resized_labels
│ ├──label_modify.py
│ ├──resize.py
│ ├──split.py
│ └── runs → Results after training
│
├── docs/
│ ├── images/
│ │ ├── logo.png
│ │ └── app_screenshot.png
│
├── PostProcessing_box.py
├── train_total.py
├── valid_total.py
├── Dockerfile
├── requirements.txt
├── yolopilot_app.py
├── Dockerfile
└── README.md
```

Project Structure

YOLOPilot

- The **first_dataset/** and **second_dataset/** folders store the two main datasets, each with separate folders for images, labels_old, and modified labels.
- Scripts such as **label_modify.py**, **resize.py**, **and split.py** are used during data preprocessing.
- The root directory has all the essential scripts for training (**train_total.py**), validation (**valid_total.py**), post-processing (**PostProcessing_box.py**), and deployment files like the Dockerfile, requirements.txt, and **yolopilot_app.py** for running the Gradio interface.
- This structure keeps the project clean, scalable, and easy to reproduce.

Dataset 1 – CARLA Object Detection

Source: CARLA-based Self-Driving Dataset

Environment: Simulated driving scenes from the CARLA simulator

Classes:

- bike, motobike, person,
- traffic_light_*, traffic_sign_*, vehicle

Format: YOLOv5-style (images + labels in .txt)

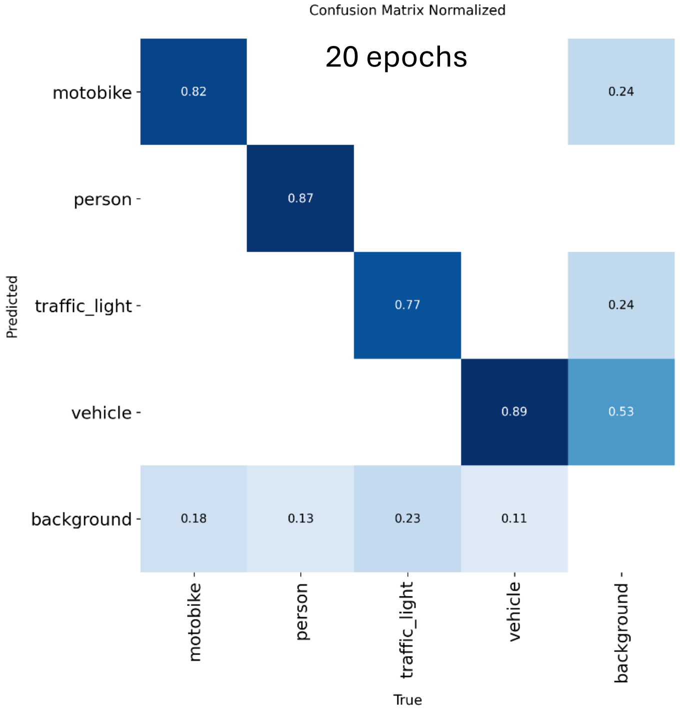
Label Issues:

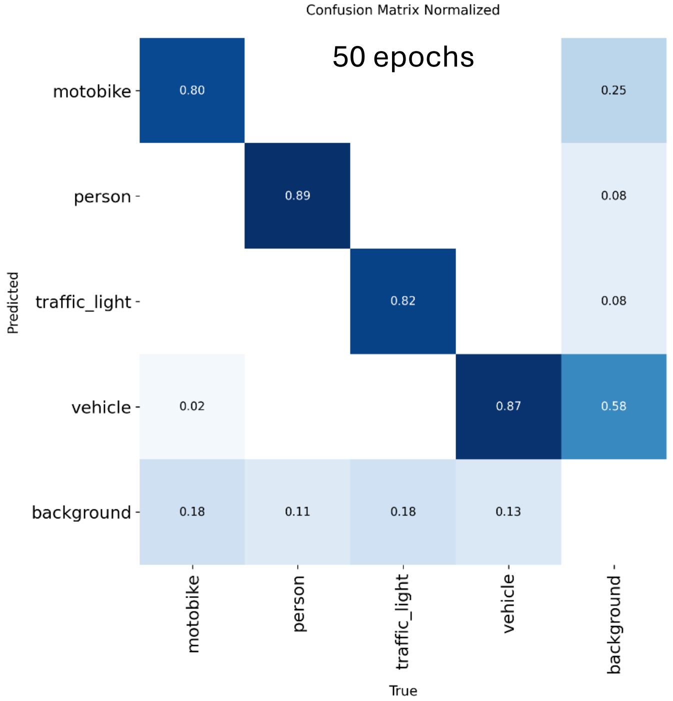
- Redundant categories (e.g., multiple traffic light classes)
- Needed mapping and simplification before training

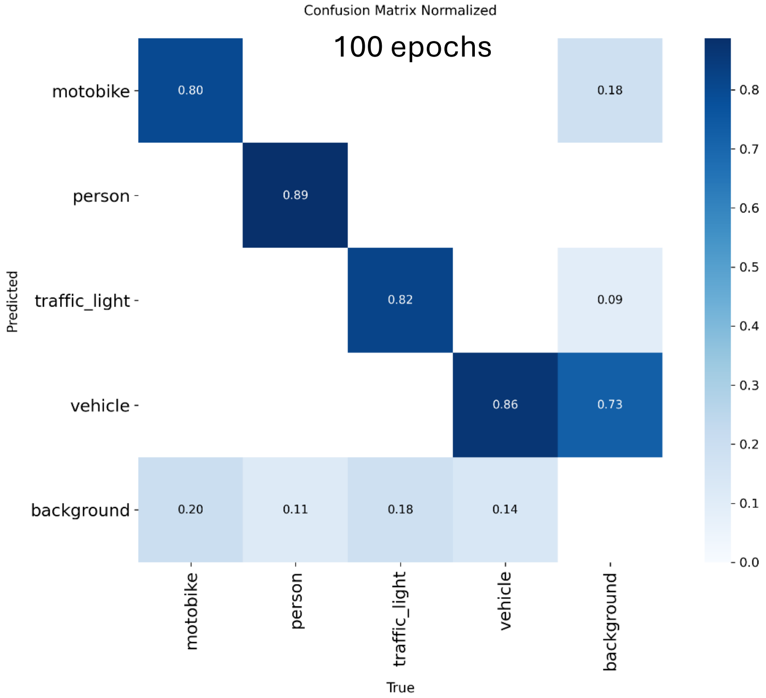
Training:

- Conducted 3 separate training runs with different epochs (20,50,100)
- One additional training via *continued training* from a previous run
- Model: yolov8s.pt

Confusion Matrix Comparison – Different Epochs (Dataset 1)







Epoch 20:

- High confusion between traffic_light and background (e.g., 23% misclassified)
- Early-stage learning with weak generalization

Epoch 50:

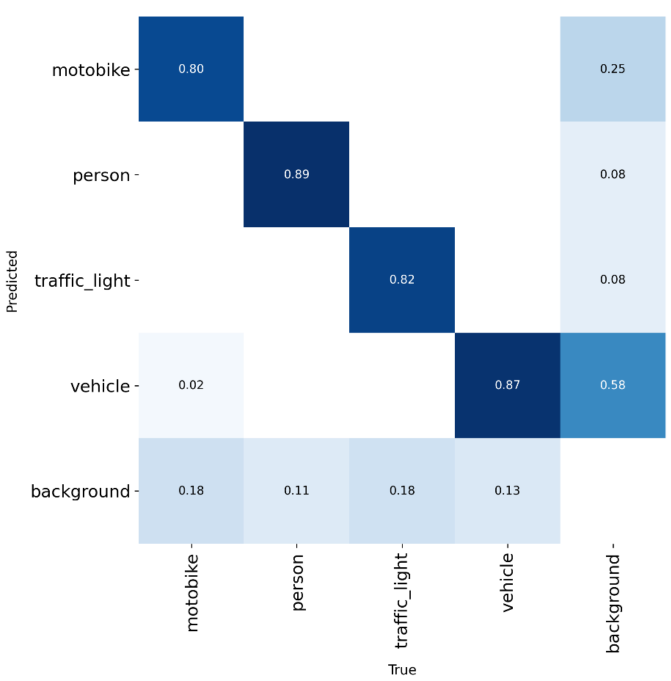
- Strong improvement in class separation
- High accuracy for pedestrian and vehicle

Epoch 100:

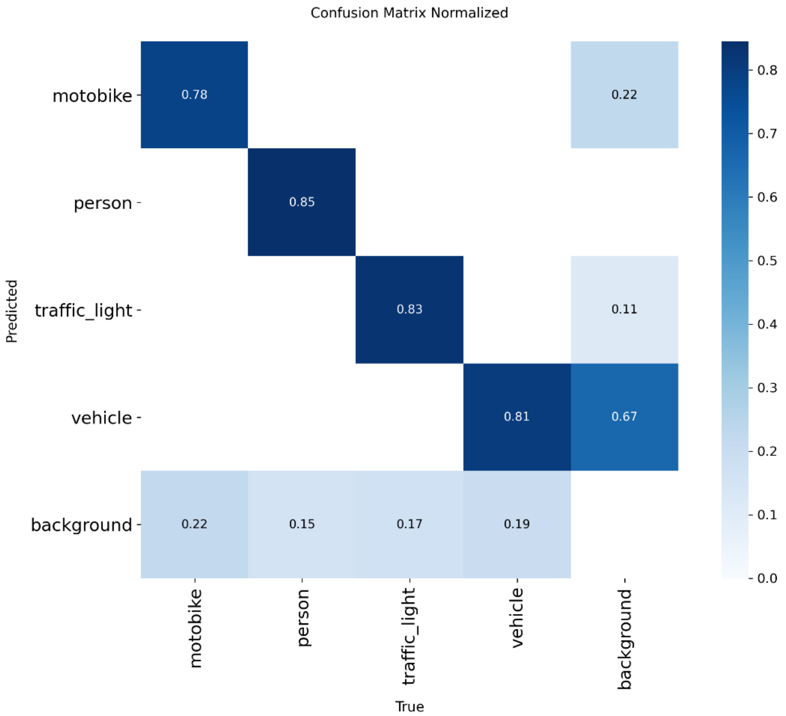
- Class-wise accuracy remains stable
- Gains are marginal compared to 50 epochs

Continued Training (Warm Start)

One Attempt from Scratch



Continued Training (Warm Start)



- Minor improvement in traffic_light (0.83 vs. 0.82)
- Drop in vehicle precision (0.81 vs. 0.86)
- Increased background confusion
- No significant gain over 50-epoch standalone model

Conclusion:

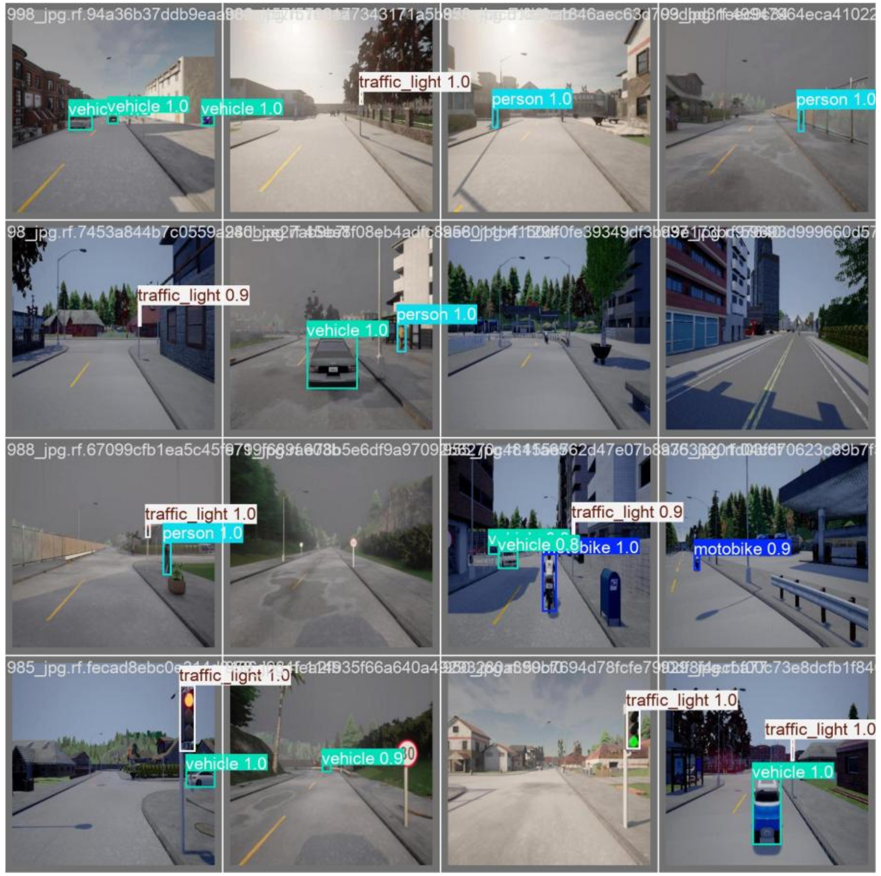
Training plateau reached after 50 epochs

→ Continued training provided diminishing returns

→ 50-epoch model selected for transfer learning on Dataset 2

Sample Visual Results – Dataset 1 (50 Epochs)

Actual labels from validation images



Predicted labels from validation images

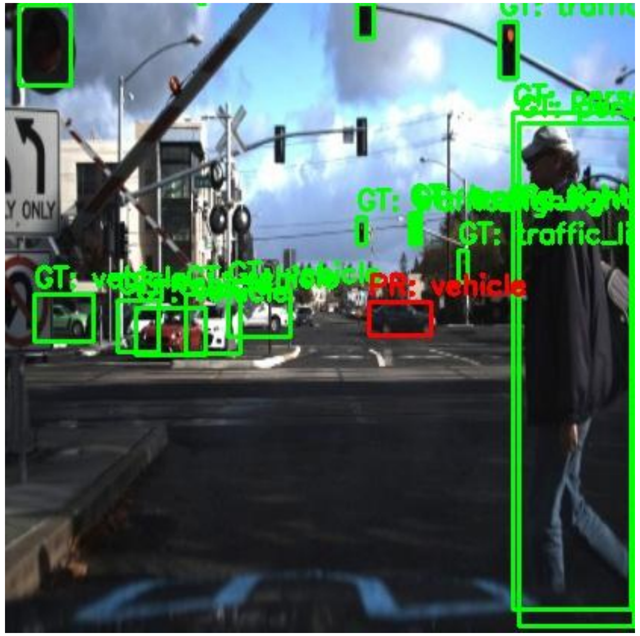
The model demonstrates:

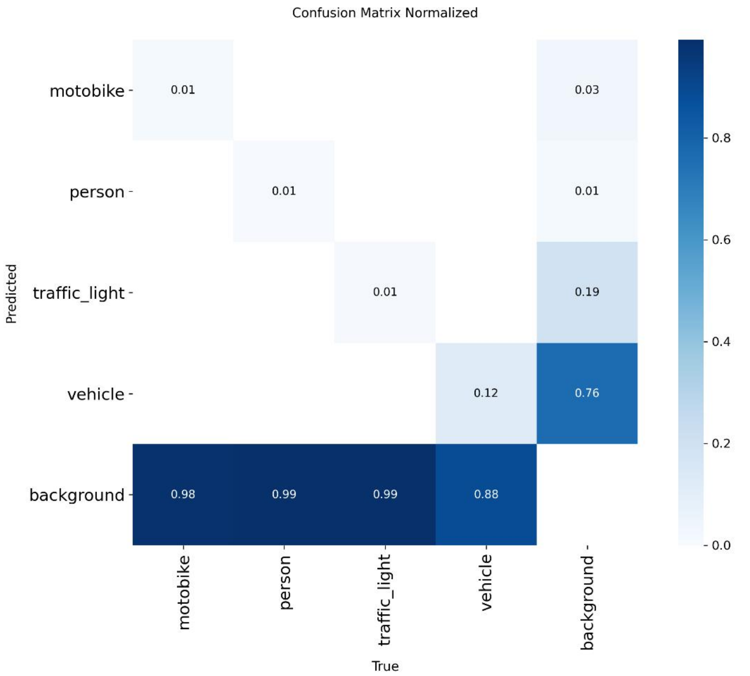
- Strong performance on vehicle, person, and motobike detection.
- Tight bounding boxes and high confidence scores for clearly visible objects.

Challenges observed:

- Occasional missed detections of traffic lights, especially when small, occluded, or poorly lit.
- Some misclassifications (e.g., traffic light predicted as person, or missed motobike in crowded scenes).

Evaluation on Second Dataset (Before Transfer Learning)





- Dataset 2 features real-world traffic scenes from video frames — introducing challenges like:
    - Varying lighting conditions
    - Occlusions and clutter
    - Diverse object sizes and angles

Initial Results (Without Transfer Learning)

- The original model fails to generalize to this new domain.
- Most detections are classified as background, with key classes like *traffic_light* and *motobike* frequently missed.
- The domain shift reveals that training only on synthetic or limited environments is insufficient for robust real-world deployment.

Second Dataset Setup

To adapt the YOLOv8 model to the second (real-world) dataset, two preprocessing strategies were tested.

Both setups used best.pt weights from 50-epoch training on the first dataset as the starting point.

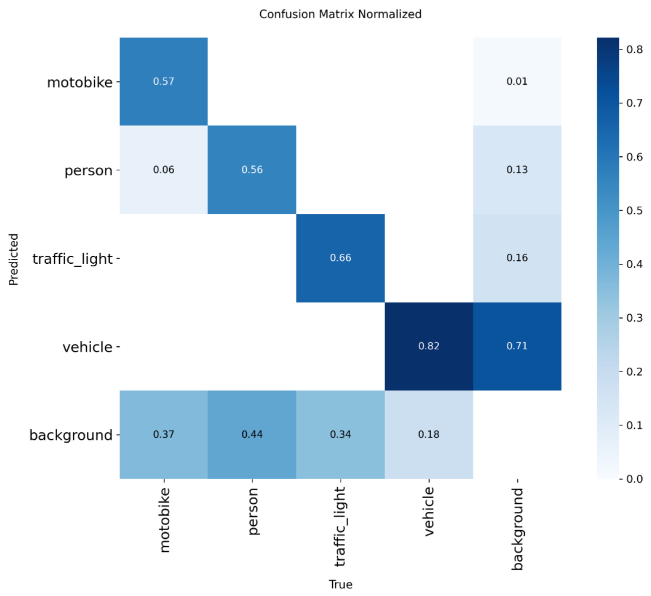
Approach 1: Explicit Resizing (416×416)

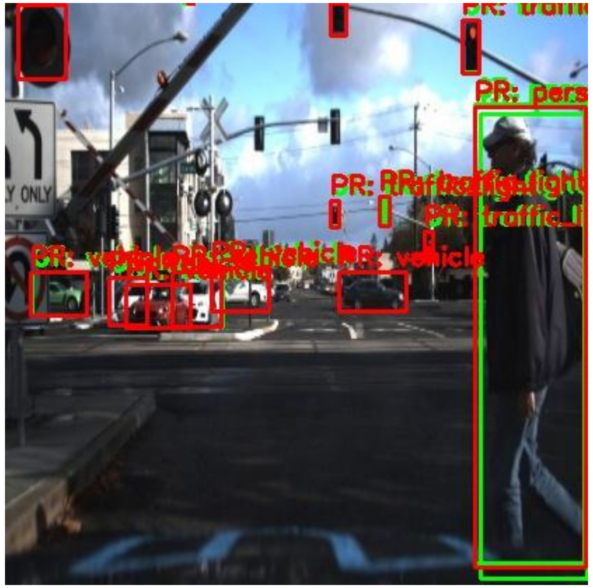
- All images resized from **480×300 → 416×416**

Approach 2: Auto Resize & Padding (YOLOv8 Default - 640×640)

- Original images passed without resizing
- YOLOv8 internally pads and resizes to 640×640

Approach 1: Explicit Resizing





Confusion Matrix Insights (Figure 13)

•Vehicle detection is highly accurate (0.82)

•Motobike and Traffic Light classes achieve moderate performance (0.57, 0.66)

•Person class remains difficult (0.56)

•Clear improvement from background-dominated misclassifications seen prior to transfer learning

✅ Confirms manual resizing + transfer learning is effective in adapting to real-world scenes

•Red boxes = Model predictions; Green boxes = Ground truth

•Objects such as vehicles, traffic lights, and pedestrians are successfully detected

•Significant improvement over pre-transfer model, which often missed all objects

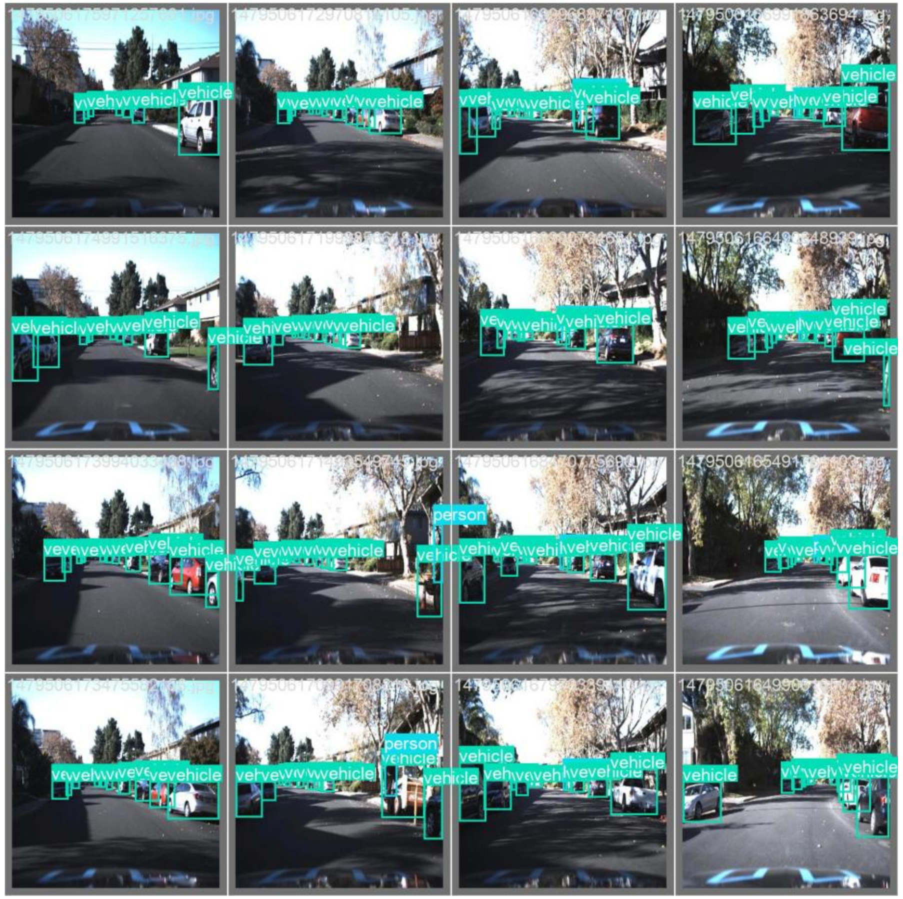
•Minor issues:

•Some duplicate or overlapping boxes

•Occasional misalignment between prediction and actual object

Approach 1: Explicit Resizing

YOLOPilot

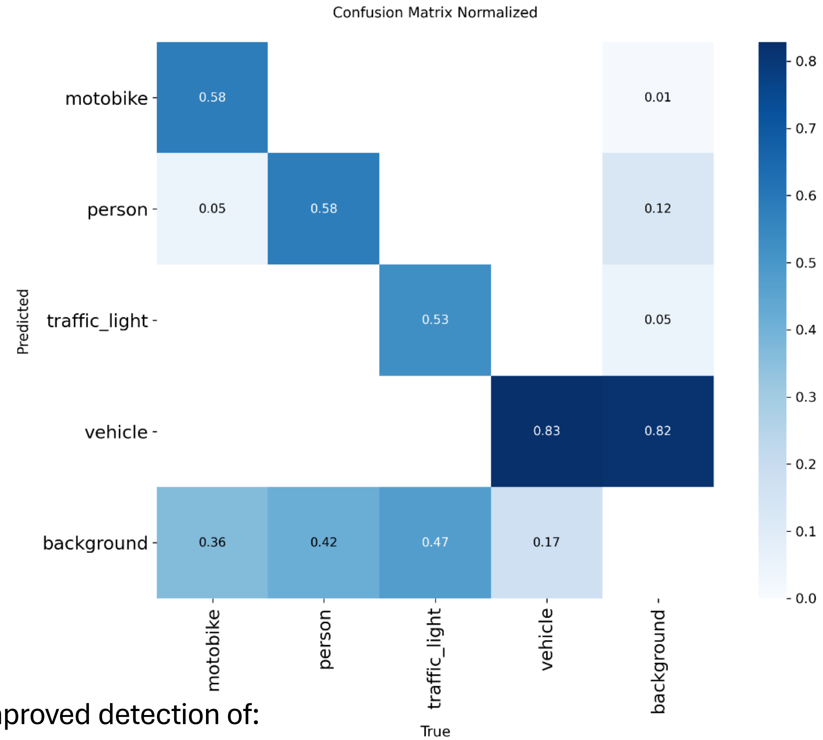


Actual labels from validation images

Predicted labels from validation images

- Accurate and consistent detection of multiple vehicles in real traffic scenes
- Tight bounding boxes with confidence scores above 0.9 for most vehicles
- Occasional successful pedestrian detections, proving class generalization

Approach 2: Auto Resize & Padding



Improved detection of:

- **Persons** – thanks to better spatial preservation
- **Motobikes** and **vehicles** – aided by higher input resolution
- Slight drop in **traffic light** accuracy
- Padding may reduce visibility of small-scale or distant objects
- Background confusion reduced, indicating clearer distinction between object and non-object areas

Conclusion:

- This method offers enhanced generalization for larger objects
- Performs slightly better overall than Approach 1 for complex, real-world scenes
- However, manual resizing may still be preferable for small object detection like traffic lights

Approach 2: Auto Resize & Padding

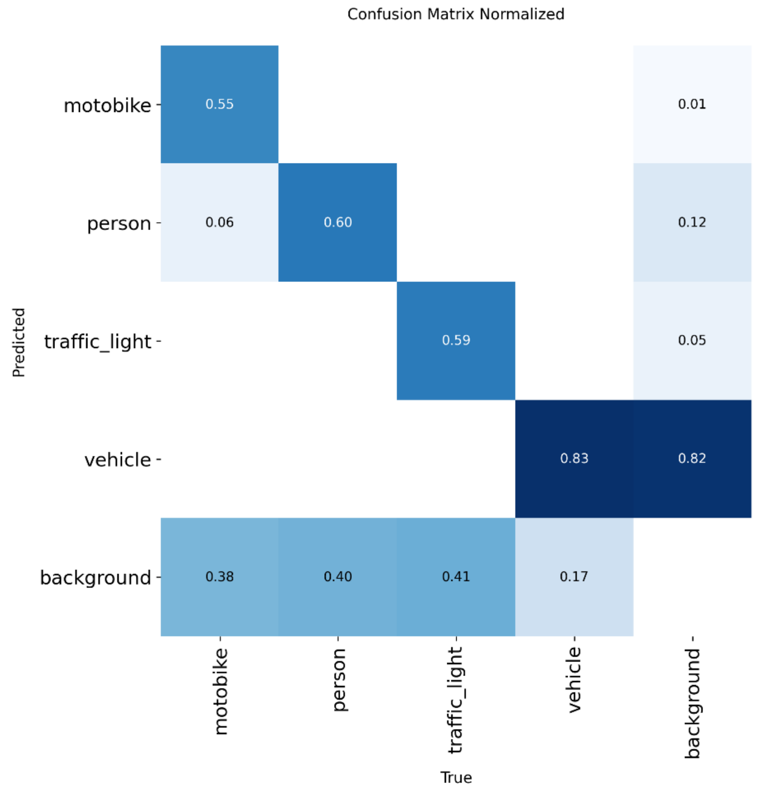
YOLOPilot

Actual labels from validation images

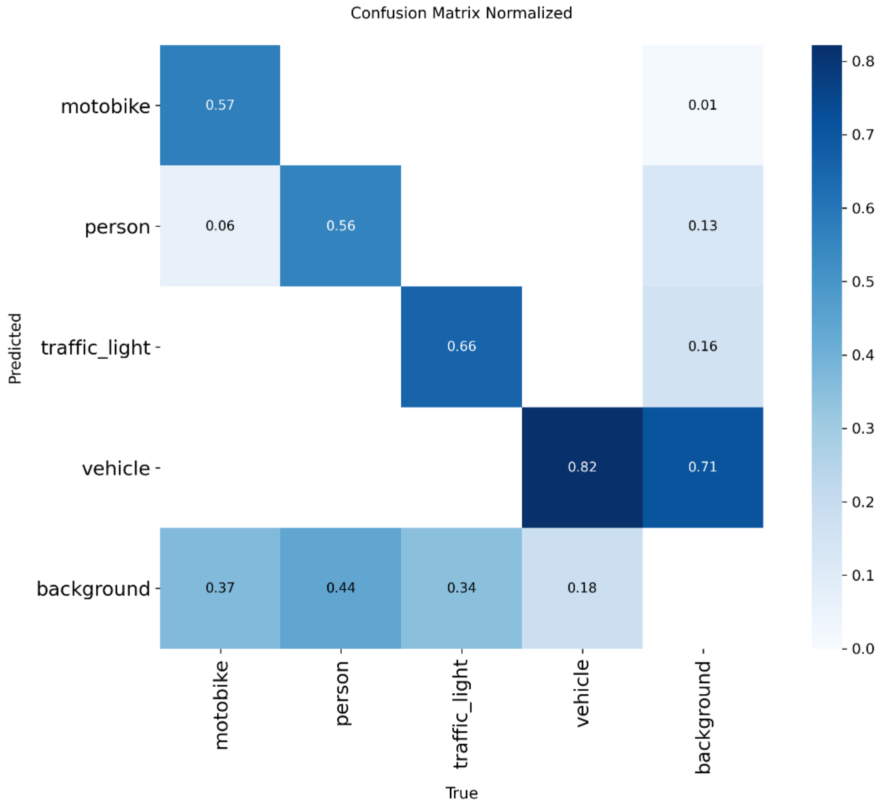
Predicted labels from validation images

Training on second data set without transfer learning

Without Transfer Learning



With Transfer Learning



- Improved class separation overall
- Motobike and traffic_light accuracy slightly better than with transfer learning
- Person detection more consistent
- Vehicle detection remains strong and comparable
- Confusion matrix shows cleaner boundaries between object classes and background

- Training time: 5h 30m (vs. 7h with transfer learning)
- Transfer learning required extra adaptation overhead

Conclusion and Further Work

Summary of Study

Evaluated multiple training strategies on a real-world object detection task using YOLOv8:

- Transfer Learning – Resized with label adjustment
- Transfer Learning – YOLO’s Auto Resize & Padding
- Training from Scratch on Dataset 2

Key Findings

- Transfer learning improves generalization over no pretraining
- Manual resizing + corrected labels performs better than auto-padding for small objects
- Training from scratch gives best accuracy and fastest training time.
    - Especially for motobike and traffic_light classes

Conclusion

- YOLOv8s can be effectively trained from scratch on real-world data. Transfer learning helps, but is not always necessary when dataset size and quality are sufficient.

Future Directions

- Experiment with larger models: YOLOv8m, YOLOv8l

Thank you for your attention

YOLOPilot