YOLOPilot

Self driving through YOLOv8 and Transfer learning

Final project submission for Code Labs Academy Data Science and AI Bootcamp

YOLOPilot software available at: https://github.com/rezazadi/YOLOPilot.git

Author: Mohammad Reza Azadi Tinat

Date: July 2025

Rev. 1

Contents

1. Introduction

In recent years, autonomous driving has emerged as one of the most transformative applications of artificial intelligence. Among the many challenges self-driving systems face, real-time object detection remains a critical task for safe and reliable navigation. This project, titled "YOLOPilot"**,** aims to develop an object detection pipeline optimized for autonomous vehicles using the YOLOv8 deep learning model and transfer learning techniques.

The core idea of this project is to leverage the power of pre-trained models and apply fine-tuning to adapt the detection system to different visual environments. By training on one dataset and fine-tuning on a more extensive, diverse second dataset, we aim to enhance the system's ability to generalize across various driving scenarios.

The main goals of the YOLOPilot project are:

- To build a robust object detection system using YOLOv8 tailored for self-driving applications.
- To apply transfer learning and fine-tuning strategies to improve performance across multiple datasets.
- To evaluate the model's performance using metrics such as confusion matrix analysis.
- To create a modular, reproducible pipeline for future use in real-world autonomous driving scenarios.

This project uses two synthetic datasets representing different self-driving environments, one of them inspired by the CARLA simulator and pre-annotated in YOLO format.

- **First Dataset**: A small-scale dataset with ~1120 training images with size 416x416 pixels. It features clear weather, basic urban environments, and limited object variation. This dataset is used to train the initial base YOLOv8 model.

  Link: https://www.kaggle.com/datasets/ibrahimalobaid/object-detection-carla-self-driving-car

- **Second Dataset**: A larger and more diverse dataset with ~18,000 training images with size 480x300 pixels. It includes varied lighting, environments, and complex scenes. This dataset is used to fine-tune the model using transfer learning.

Link: https://www.kaggle.com/datasets/alincijov/self-driving-cars/data

The object categories shared across both datasets include:

- Motobike
- Pedestrian
- Traffic light
- Vehicle

Each image may contain multiple instances of these objects, simulating real-world driving conditions.

Traditional CNN-based classifiers are insufficient for tasks requiring spatial awareness and multi-object detection. YOLO (You Only Look Once) models offer high-speed, high-accuracy detection by predicting bounding boxes and class probabilities in a single forward pass. In this context, YOLOv8 offers the latest architecture and training improvements, making it a strong candidate for object detection in self-driving systems.

By incorporating transfer learning, the project demonstrates how models trained on a limited dataset can be scaled effectively to larger domains, saving time and computational resources while increasing generalization.

2. Application steps

The YOLOPilot project follows a step-by-step approach for building and evaluating a robust object detection pipeline tailored for self-driving environments:

1. **Dataset Preparation**
   Organize the two datasets in YOLOv8-compatible structure (train/images, train/labels, etc.). Convert and match images with labels if necessary.
2. **Initial Training on Dataset 1**
   Use YOLOv8 with a base model (yolov8s.pt) to train on the first dataset. Evaluate results using validation metrics and confusion matrix.
3. **Error Analysis**
   Identify and visualize mismatches between predicted and ground truth bounding boxes.
4. **Transfer Learning & Fine-Tuning on Dataset 2**
   Load the trained weights from Dataset 1 (best.pt) and continue training on the second dataset. Monitor improvements via validation results.

5. **Evaluation and Visualization**
   Save metrics, confusion matrix, and mismatched images from test/validation sets to analyze generalization and performance.

2.1. YOLO model

YOLOv8 is a modern, anchor-free object detection model developed by Ultralytics. It is fast, accurate, and well-suited for real-time applications like self-driving systems.

The model consists of four main parts:

- The **Backbone** extracts features from the input image using convolutional layers (CSPDarknet).
- The **Neck** fuses these features at different scales using a Feature Pyramid Network (FPN).
- The **Head** predicts bounding boxes and class scores without using predefined anchor boxes.
- **Post-processing** applies Non-Maximum Suppression (NMS) to remove overlapping boxes.

The input image is resized (e.g., to 416×416 or 640×640), passed through the model, and results in final object detections. YOLOv8 supports transfer learning and can be fine-tuned on new datasets efficiently, making it ideal for this project.

2.2. Preprocessing of images and labels

Before training and fine-tuning the object detection model, several preprocessing steps were applied to both datasets to ensure consistency in label formats and directory structure.

First Dataset

The first dataset originally contained 10 label categories, including detailed traffic light and sign classes such as traffic_light_red, traffic_sign_30, and bike. For this project, the labels were simplified and unified into four main categories to better reflect general object types relevant to self-driving:

- motobike
- pedestrian
- traffic_light
- vehicle

The label files in YOLO format were modified accordingly using a Python script named **label_modify.py**.

Second Dataset

The second dataset was provided in a different format, where object annotations were stored in a CSV file. In order to use it for YOLOv8 training, the annotations were extracted and converted into YOLO-compatible .txt files. The labels were also mapped to match those used in the first dataset. Specifically, the following conversions were applied:

- car, truck → vehicle
- pedestrian → pedestrian
- bicyclist → motobike
- light → traffic_light

This unification ensures consistent class definitions across both datasets.

Additionally, while the label files were already split into train, valid, and test folders, the images were not. To address this, a script named **split.py** was used to copy and organize the corresponding images into YOLOv8-style folder structures (train/images, valid/images, etc.) based on the label filenames.

These preprocessing steps ensured both datasets were clean, consistent, and fully compatible with the YOLOv8 training pipeline.

3. Training on first data set

3.1. Setup and fine tunings

The first dataset was organized in a standard YOLOv8 format, consisting of three primary subsets:

- train/ – contains images and label files used for training the model
- valid/ – contains images and labels for validation during training
- test/ – contains images and labels for final performance evaluation

Each folder includes two subfolders:

- images/ – the resized input images
- labels/ – the corresponding annotation files in YOLO format

All labels were corrected and simplified during preprocessing to ensure they match the unified class structure: motobike, pedestrian, traffic_light, and vehicle.

For this stage, the YOLOv8 small model (yolov8s.pt) was selected. Training was performed on the first dataset with three different epoch settings:

- 20 epochs
- 50 epochs
- 100 epochs

This was done to compare how model performance evolves with increasing training time and to identify a balanced training configuration.

After the initial training rounds, an additional experiment was conducted using the best-performing checkpoint from the 50-epoch training run. Instead of starting again from the base model, the training was resumed from the saved best.pt file generated during the earlier run.

This continuation technique—also referred to as fine-tuning from a previous checkpoint—allows the model to build on its previous learning and potentially refine its understanding of the data. The goal was to explore whether further training from an already optimized state could yield improvements without restarting the learning process.

3.2. Results

3.2.1. Different epochs

To evaluate the performance of the YOLOv8 model on the first dataset, the training was performed using three different epoch settings: **20**, **50**, and **100**. For each configuration, a confusion matrix was generated using the validation dataset to assess how well the model distinguished between the four target classes: motobike, pedestrian, traffic_light, and vehicle. An additional background class was included in the evaluation to reflect cases of missed detections or false positives. The confusion matrices are normalized to allow for easier comparison across training durations.

Figure 1 presents the confusion matrix after 20 epochs of training. The model demonstrates the ability to begin separating the main classes, especially pedestrians and vehicles, with reasonable accuracy. However, there is still significant confusion between traffic-related classes and background, particularly with traffic lights, which show a 0.23 false detection rate as background. This suggests that while the model is learning basic features, it has not yet generalized robust patterns for all object types.

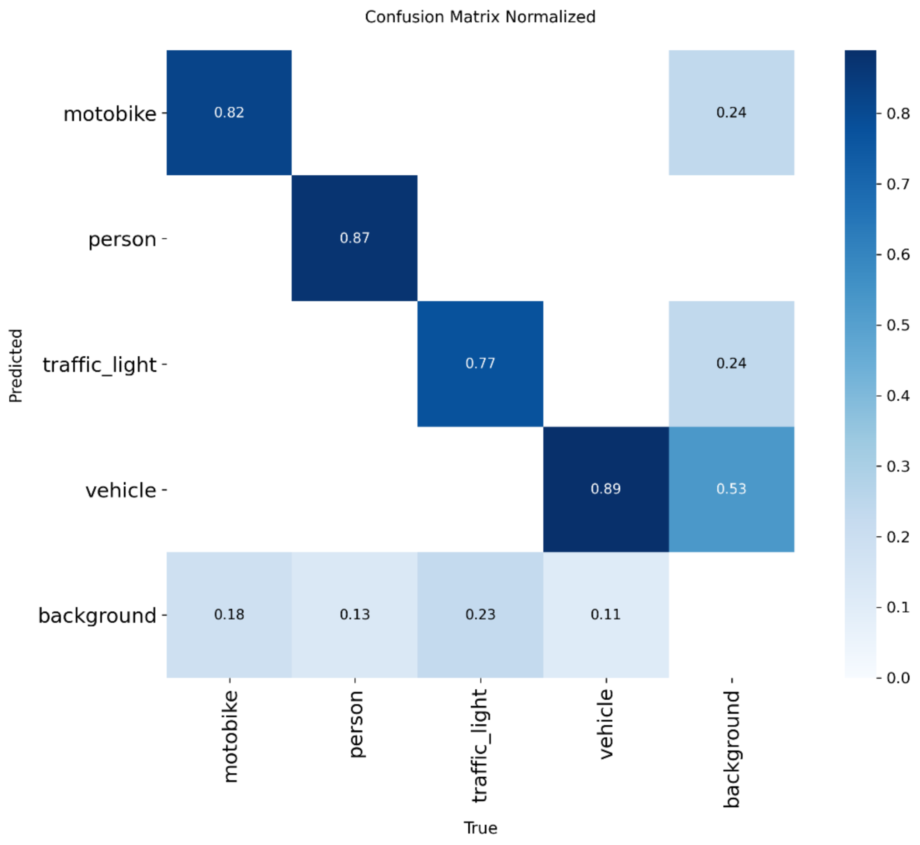

Figure 1 Normalized Confusion Matrix for first dataset with 20 epochs

Figure 2 shows the confusion matrix obtained after 50 epochs of training. At this stage, the model has improved its classification performance considerably. The pedestrian and vehicle classes achieve higher accuracy, and the confusion between traffic_light and background is reduced. The results indicate that the model has generalized better and is able to detect more instances with increased confidence. This training stage appears to offer a good trade-off between learning and overfitting.

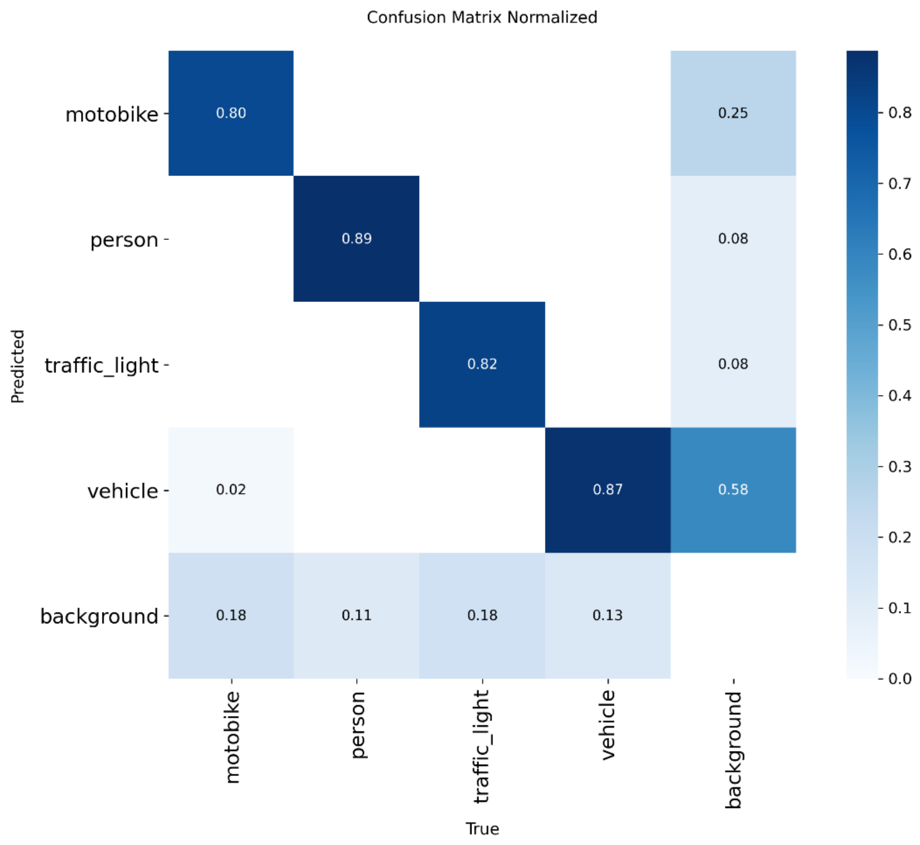

Figure 2 Normalized Confusion Matrix for first dataset with 50 epochs

Figure 3 depicts the confusion matrix after 100 epochs of training. The class-wise accuracy remains strong, particularly for pedestrian, but there is a slight increase in confusion between vehicle and background compared to the 50-epoch model. This suggests that the model may be starting to overfit or is no longer gaining significant improvements with additional training. While accuracy remains stable, the marginal gains do not justify the additional computational time beyond 50 epochs in this setup.

These results informed the decision to use the best-performing weights from the 50-epoch model for a second round of training (fine-tuning), which is discussed in the following chapter.

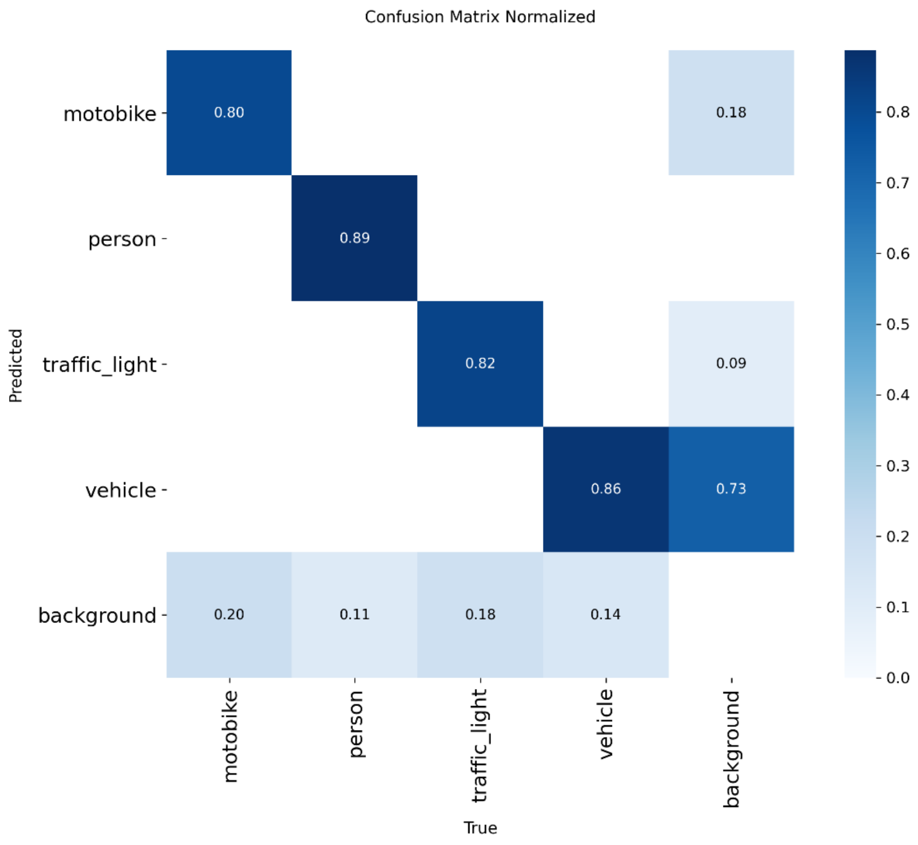

Figure 3 Normalized Confusion Matrix for first dataset with 100 epochs

3.2.2. Continued Training and Performance Saturation

Figure 4 illustrates the confusion matrix for a training setup where the model was not initialized from scratch (e.g., yolov8s.pt), but rather fine-tuned using the **best weights obtained from the 50-epoch training**. This strategy, known as *continued training* or *warm start*, leverages learned features and attempts to improve performance further. The intention was to determine whether fine-tuning the already well-trained model could lead to higher accuracy or better generalization.

However, compared to the original 50-epoch results shown in Figure 2, the fine-tuned model in Figure 4 did not significantly improve class-level predictions. While the traffic_light class slightly improved (0.83 compared to 0.82), other classes like vehicle dropped in precision (0.81 vs. 0.86), and the background confusion increased. This suggests that the model may have reached a performance plateau by epoch 50, and continued training on the same dataset yielded diminishing returns. Therefore, using the best.pt from epoch 50 appears to be a solid stopping point for this dataset before switching to transfer learning on the second dataset.

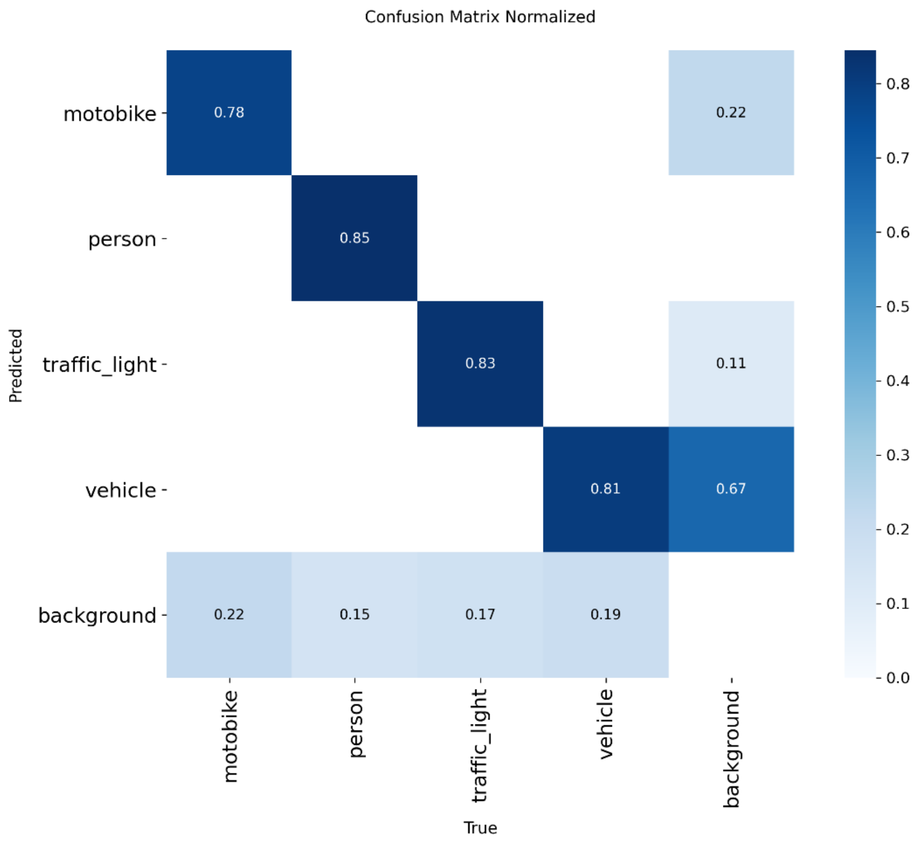

Figure 4 Normalized Confusion Matrix for first dataset with 50 epochs continue training attempt

3.2.3. Visual Validation of Predictions on Validation Set (Epoch 50)

To further evaluate the performance of the model trained for 50 epochs, sample outputs on validation images were analyzed. Figure 5 displays the ground-truth annotations, while Figure 6 shows the model's predictions on the same images.

Upon visual inspection, the model performs well in identifying most instances of vehicle, person, and motobike. The bounding boxes align closely with the actual objects, and the confidence levels are consistently high for clearly visible objects.

However, the traffic_light class occasionally goes undetected, even when visibly present in the scene. This suggests that the model struggles with detecting smaller or less prominent traffic lights, particularly under varied lighting or occlusion conditions.

In some cases, while the object is successfully predicted, the class label is confused — for example, a traffic light being misclassified as person, or motobike missed entirely in congested scenes. These misdetections align with the confusion matrix results presented earlier, where inter-class confusion was noted, especially with background elements.

This visual analysis confirms the quantitative findings and highlights specific challenges the model faces in real-world-like environments, making it a crucial diagnostic step before transferring the model to new data.

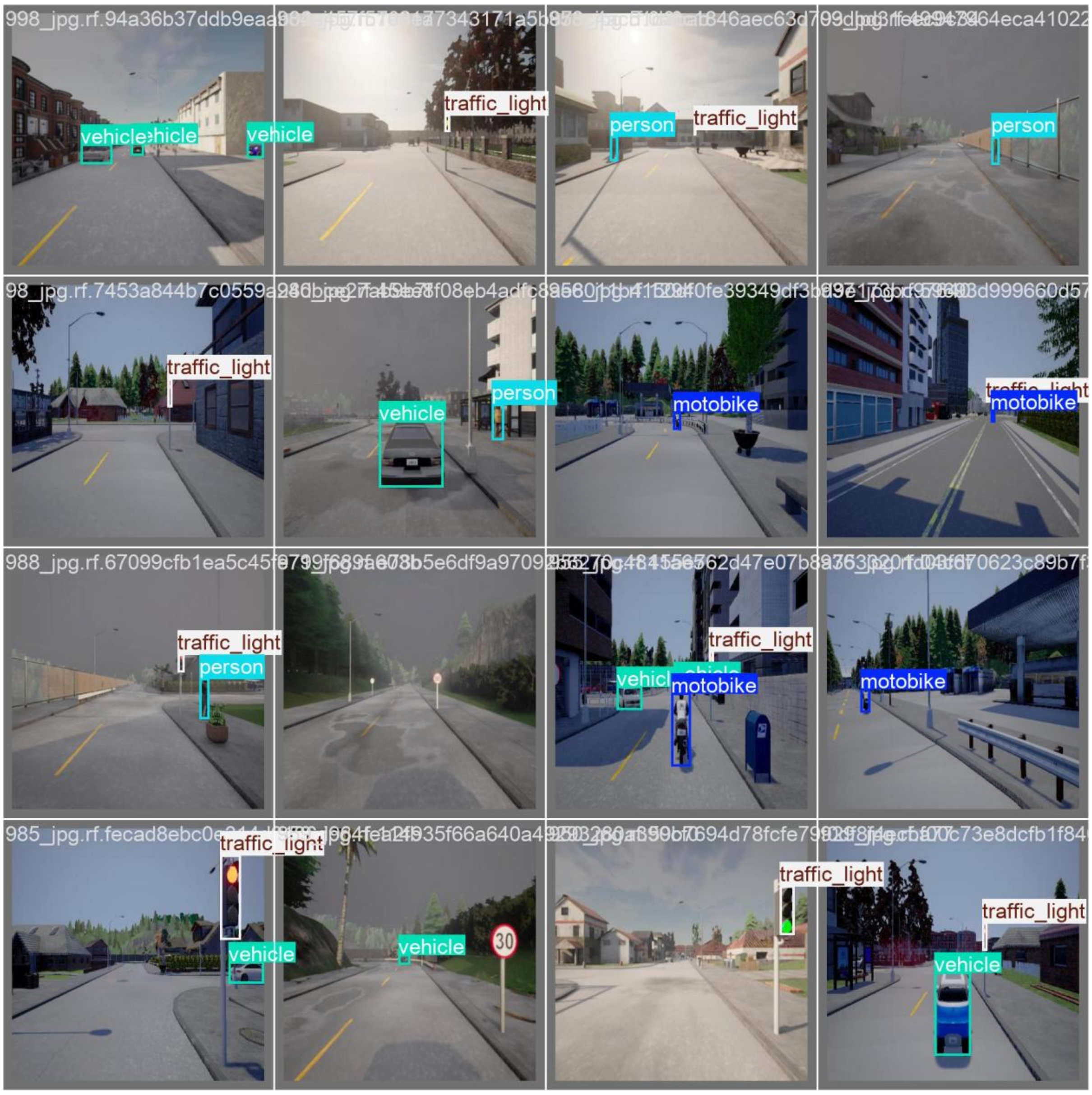

Figure 5 Actual labels from validation images

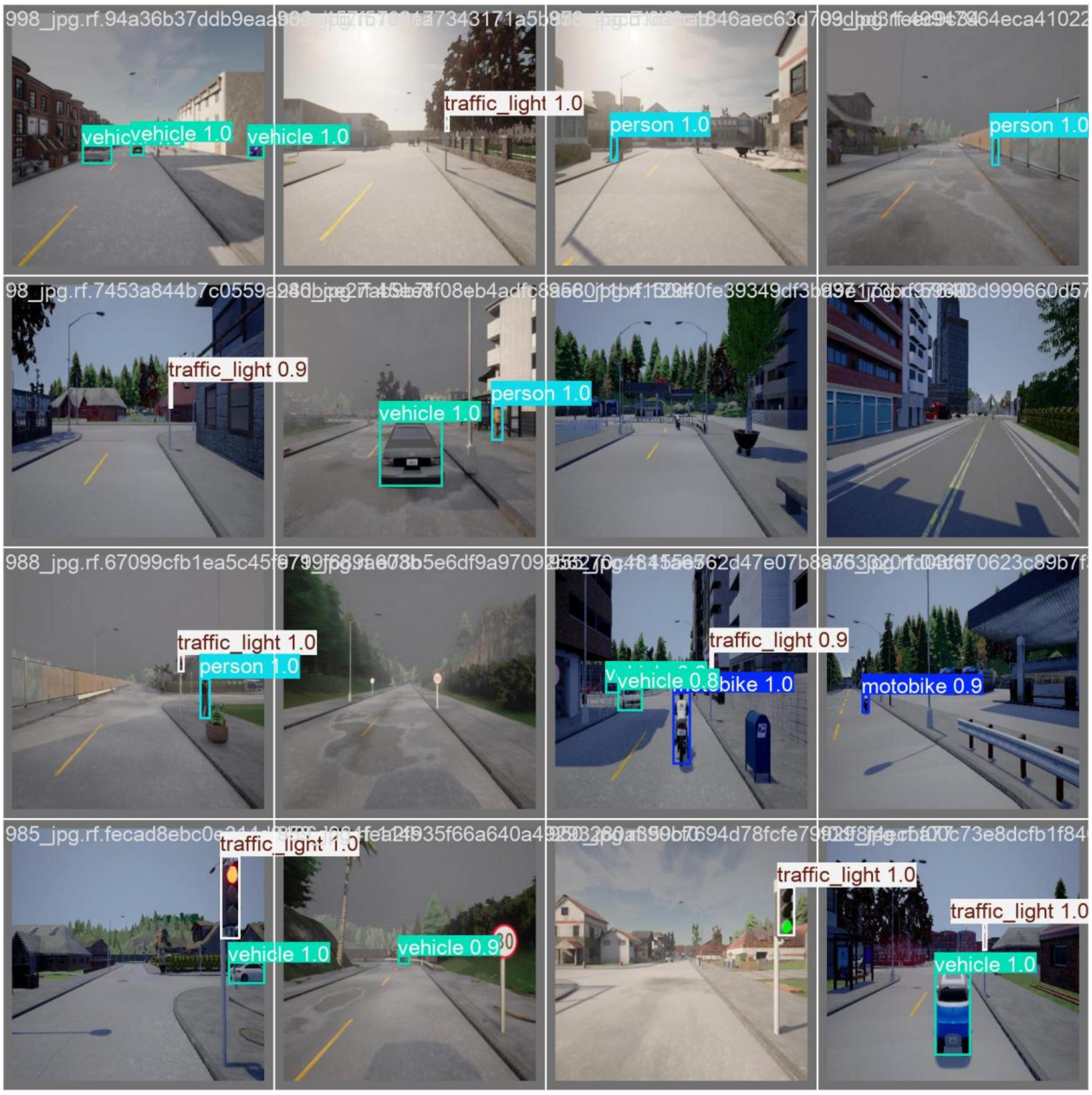

Figure 6 Predicted labels from validation images

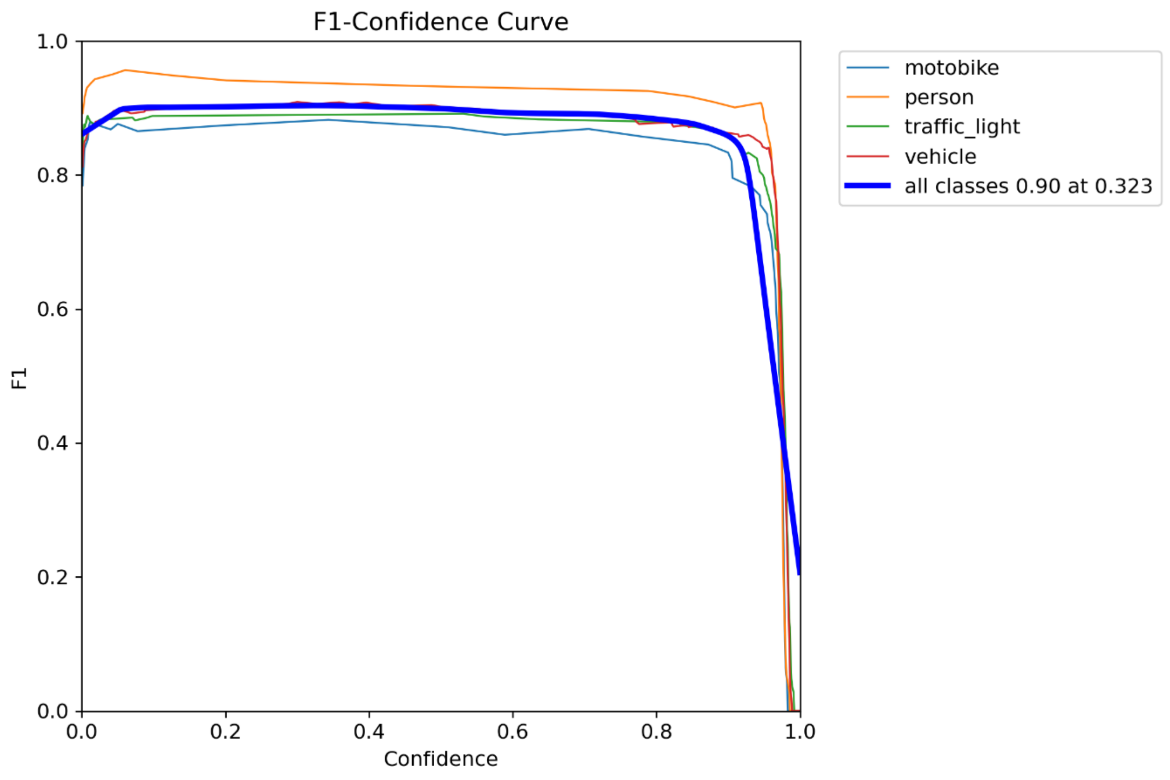

Figure 7 F1 curve

The F1-Confidence Curve provides insight into the trade-off between prediction confidence and classification performance. In this figure, each line represents the F1 score across varying confidence thresholds for a specific class: *motobike*, *person*, *traffic_light*, and *vehicle*. The bold blue line aggregates the performance across all classes.

The horizontal axis shows the **confidence threshold**, ranging from 0 to 1. As the confidence threshold increases, the model only considers predictions it is more certain about. On the vertical axis, the **F1 score** is plotted, which balances precision and recall — higher values indicate better overall classification.

The peak of the thick blue curve is reached at a confidence threshold of **0.323**, where the overall F1 score across all classes is **0.90**. This means the model achieves its best precision-recall balance at this point. Beyond this threshold, the F1 score drops sharply, indicating that overly strict confidence filtering causes missed detections (lower recall). On the other hand, lower thresholds may include more false positives.

Class-wise, the *person* class consistently maintains the highest F1 score across all thresholds, suggesting strong detection performance. The *motobike* and *traffic_light* classes show slightly lower scores, with *traffic_light* being more sensitive to the confidence threshold due to its small size and frequent occlusions.

Overall, this curve is essential for choosing the optimal confidence threshold when deploying the model, balancing sensitivity and precision based on the target application's requirements.

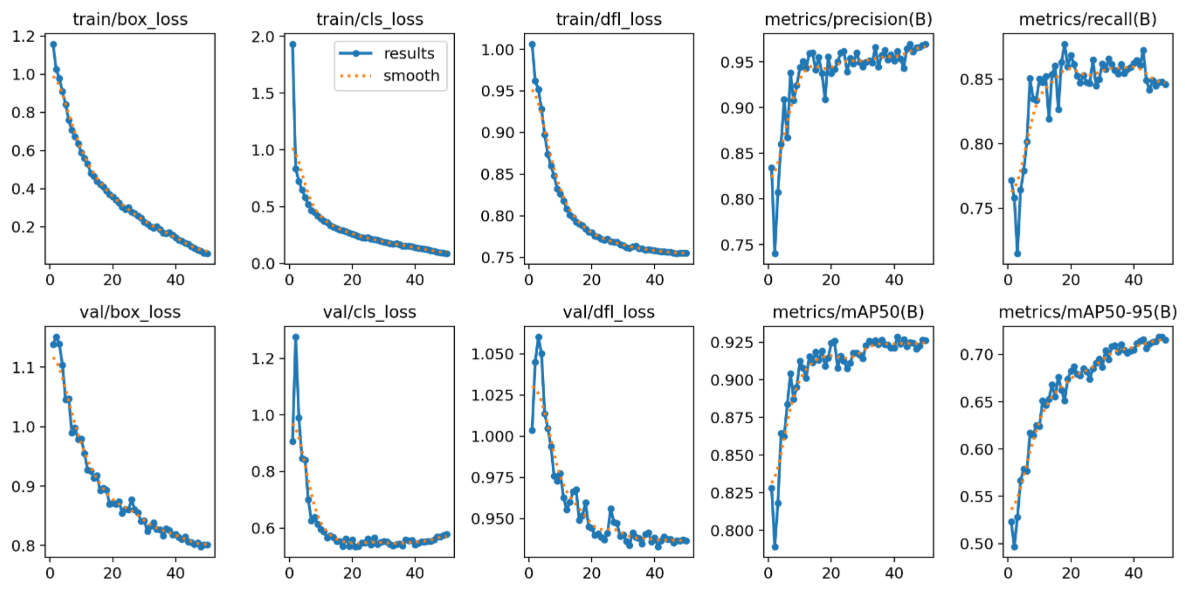

Figure 8 Overall Results

The figure8 presents the training and validation performance curves over 50 epochs for the YOLOv8 model trained on the first dataset. Each subplot captures a key metric or loss component, allowing us to evaluate learning dynamics and convergence behavior.

Top row (Training metrics):

- **train/box_loss**: The bounding box regression loss steadily decreases, indicating the model is improving its localization capabilities.
- **train/cls_loss**: Classification loss also drops consistently, showing that the model is getting better at identifying object classes.
- **train/dfl_loss**: The Distribution Focal Loss, used for precise bounding box regression, decreases gradually and stabilizes near 0.75, reflecting improved confidence calibration for object boundaries.
- **metrics/precision(B)**: Precision improves significantly after the initial few epochs and stabilizes above 0.9, indicating that the majority of detected boxes are correct (low false positives).
- **metrics/recall(B)**: Recall increases rapidly in the first 20 epochs and then fluctuates slightly, suggesting the model maintains a high rate of detecting actual objects (low false negatives).

Bottom row (Validation metrics):

- **val/box_loss**, **val/cls_loss**, and **val/dfl_loss**: These losses follow a similar decreasing trend as in training, indicating good generalization and no signs of overfitting.
- **metrics/mAP50(B)**: The mean Average Precision at IoU 0.50 shows a clear upward trend and plateaus above 0.92, confirming solid object detection accuracy at the standard threshold.
- **metrics/mAP50-95(B)**: This stricter mean Average Precision (averaged over IoUs from 0.50 to 0.95) improves consistently, ending just above 0.70. It demonstrates the model's precision across a wider range of localization tolerances.

Overall, this set of plots confirms stable and effective training. Losses steadily decrease, while precision, recall, and mAP metrics improve throughout, indicating a well-tuned learning process without overfitting.

4. Training on second data set through transfer learning

This chapter focuses on adapting the trained YOLOv8 model from the first dataset to a more complex and realistic dataset, referred to as the second dataset. This dataset consists of real-world traffic scenes extracted frame-by-frame from driving videos. It provides a more challenging domain due to variations in lighting, occlusion, and object scale.

To evaluate the performance of the original model trained solely on the first dataset, we first tested it on the validation images of the second dataset without applying transfer learning. The results reveal a significant domain shift: the model fails to detect most objects accurately and classifies the majority of instances as background.

Figure 9 shows the normalized confusion matrix for this evaluation. The matrix clearly indicates high background prediction rates for all true classes, confirming the poor generalization of the model to the new domain.

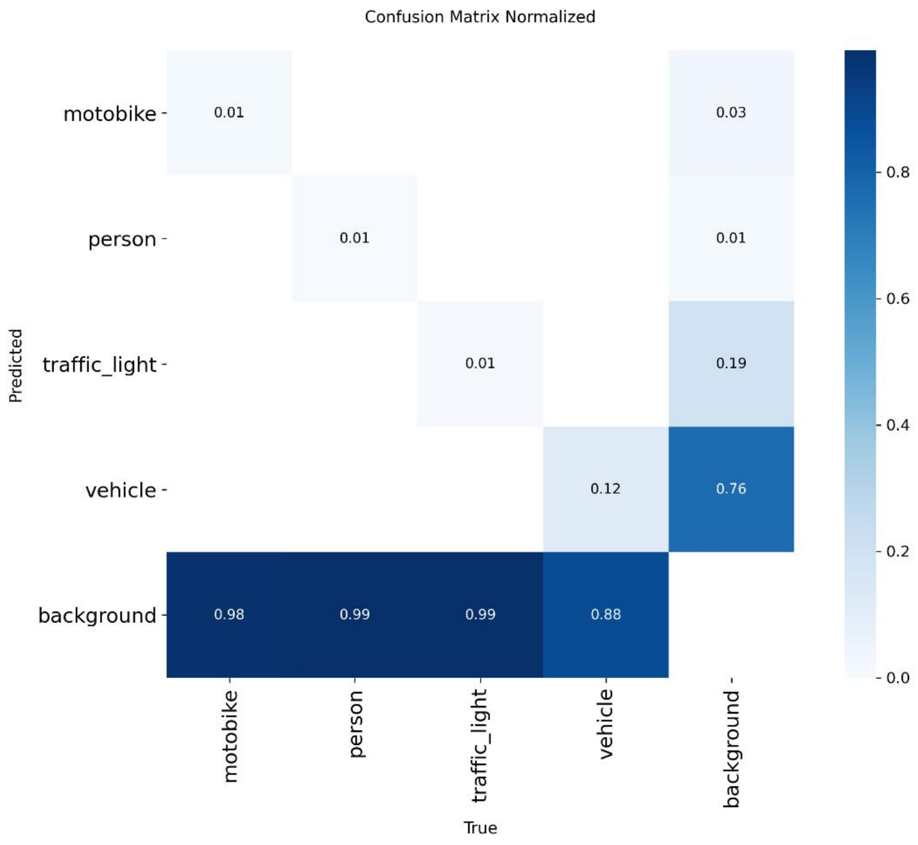

Figure 9 Confusion matrix before transfer learning on second dataset

Figure 10 provides a visual example of this failure. Ground truth annotations are marked in green (GT), while predictions from the model are marked in red (PR). It can be seen that none of the true objects were correctly detected. The model incorrectly predicts traffic lights as motobikes and misses several objects entirely.

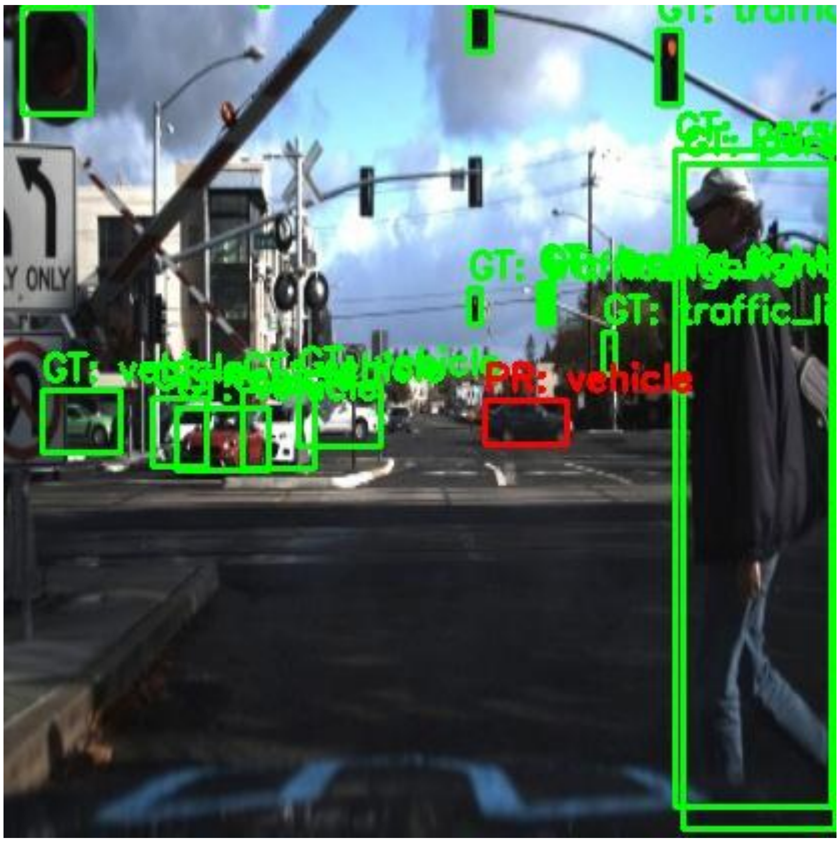

Figure 10 Second dataset image before transfer learning

These observations highlight the need for transfer learning, which will be discussed in the next sections. By fine-tuning the model on the second dataset, we aim to adapt it to the new visual domain and improve its detection capabilities.

4.1. Setup

To perform transfer learning on the second dataset, two different approaches were implemented to investigate how image preprocessing and resolution affect the model's adaptation and performance.

In the first approach, all images were explicitly resized to a fixed size of 416×416 pixels. Since the original resolution of the images was 480×300, direct resizing would distort the spatial layout of objects. To preserve label accuracy, bounding box coordinates were scaled using the following formula for each bounding box:

$$new_x = old_x \times \frac{416}{480} \qquad new_y = old_y \times \frac{416}{300}$$

$$new_{width} = old_{width} \times \frac{416}{480} \qquad new_{height} = old_{height} \times \frac{416}{300}$$

Using this corrected annotation set, transfer learning was applied starting from the best weights obtained after 50 epochs of training on the first dataset.

In the second approach, instead of resizing manually, the images were passed to the YOLOv8 training pipeline with automatic adjustment. The images were padded and resized to 640×640 pixels using YOLO's built-in preprocessing logic, which preserves the aspect ratio by padding the missing pixels equally around the image. This method helps prevent geometric distortion and retains more visual context, which is especially useful for detecting small objects like traffic lights.

This dual-strategy setup was intended to evaluate the trade-off between input standardization and information preservation, and to determine which preprocessing technique yields better performance in fine-tuning for real-world scenes.

4.2. Results

4.2.1. Manual resizing approach

The F1-Confidence Curve shows that the overall model performance improved substantially compared to training without transfer learning. The optimal confidence threshold for all classes is determined to be 0.238, where the aggregated F1-score reaches approximately 0.74. Among the individual classes, vehicles maintain the highest F1 score across all confidence levels, while the person class exhibits the lowest performance, likely due to the small scale or occlusion of instances in the real-world dataset.

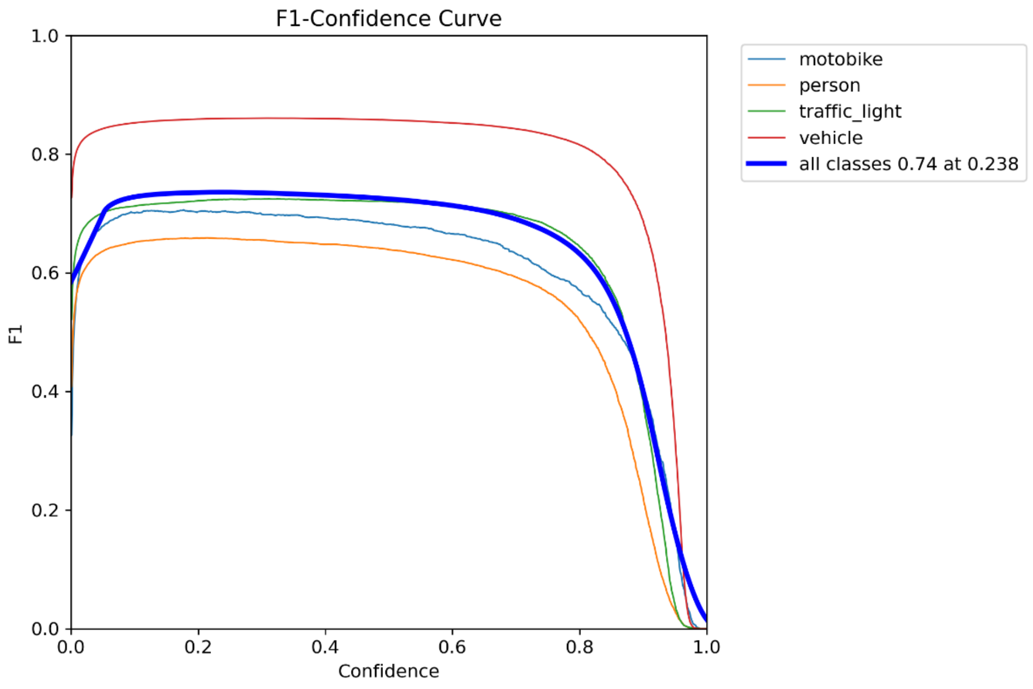

Figure 11 F1 Curve

The training and validation loss curves confirm consistent learning. All major losses box, classification (cls), and distribution focal loss (dfl) exhibit a smooth and steady downward trend. The metrics graphs indicate continuous improvement in both precision and recall across epochs. The mAP50 metric approaches 0.75, and mAP50-95 nears 0.55, reflecting effective generalization. These values demonstrate meaningful gains over the non-transfer baseline and indicate that manual preprocessing helped facilitate knowledge transfer from synthetic to real data.

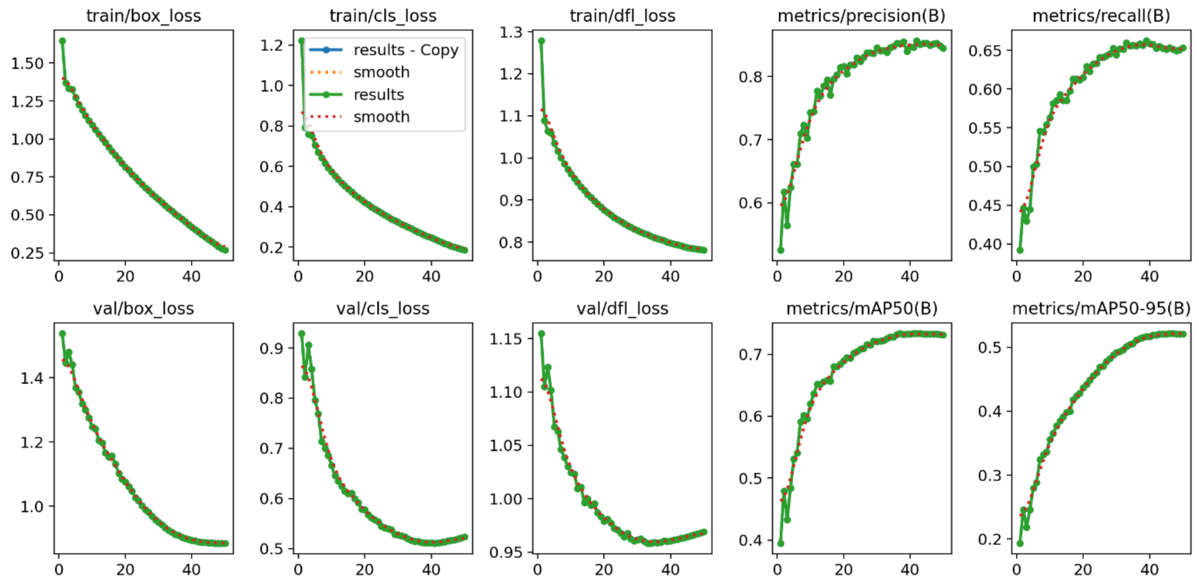

Figure 12 Overall Results

The confusion matrix reinforces this conclusion. While not perfect, most classes are now correctly detected with moderate confidence. The vehicle class shows high accuracy (0.82), and the motobike and traffic_light classes reach acceptable prediction levels (0.57 and 0.66, respectively). Although the person class remains challenging (0.56), the overall matrix shows a clear departure from the background-dominated misclassifications seen earlier, validating the effectiveness of the manual resizing and transfer learning approach.

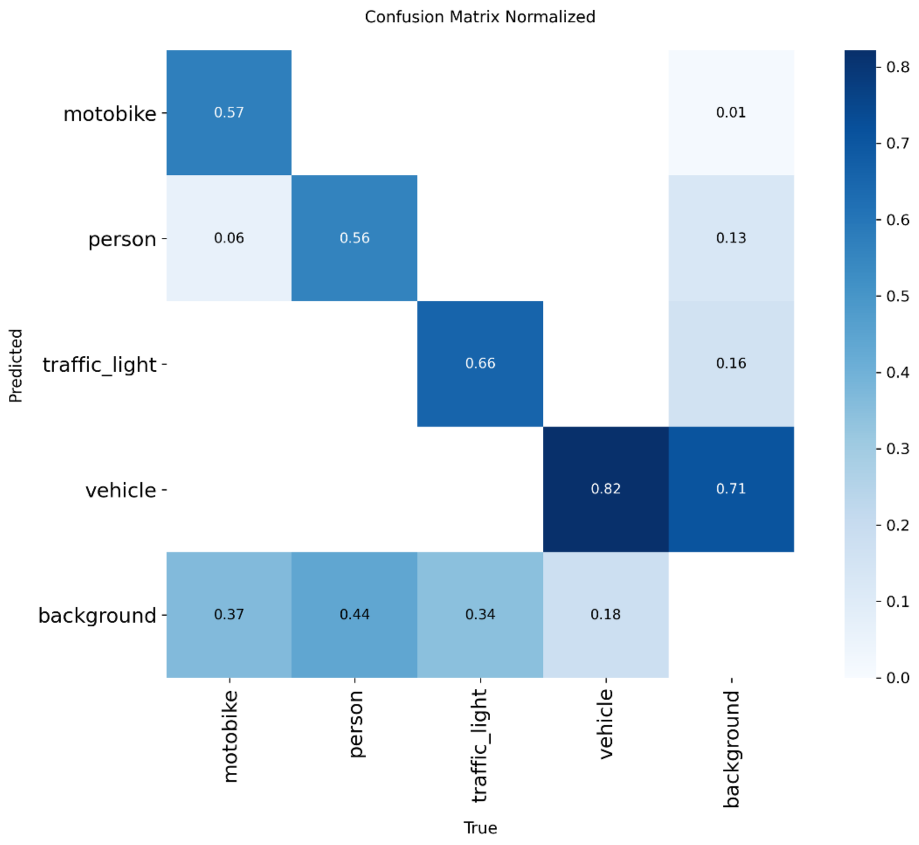

Figure 13 Confusion matrix

The figure 14 illustrates a real-world scene where object detection is performed using the model trained via manual resizing and transfer learning. The green boxes indicate the ground truth labels, while the red boxes show the predicted results.

Compared to the performance before transfer learning, this image demonstrates a clear improvement. Previously, the model often missed all objects or misclassified them as background. Now, the model successfully detects multiple vehicles, traffic lights, and persons, even though there are still some duplicate or overlapping predictions. For instance, several traffic lights are correctly identified, and most vehicles are accurately localized.

However, some predictions are slightly redundant or not perfectly aligned with the ground truth, such as overlapping boxes or multiple predictions for the same object. Despite these minor inaccuracies, the image shows that the transfer learning approach, combined with careful resizing, significantly enhanced the model's ability to generalize from synthetic to real data.

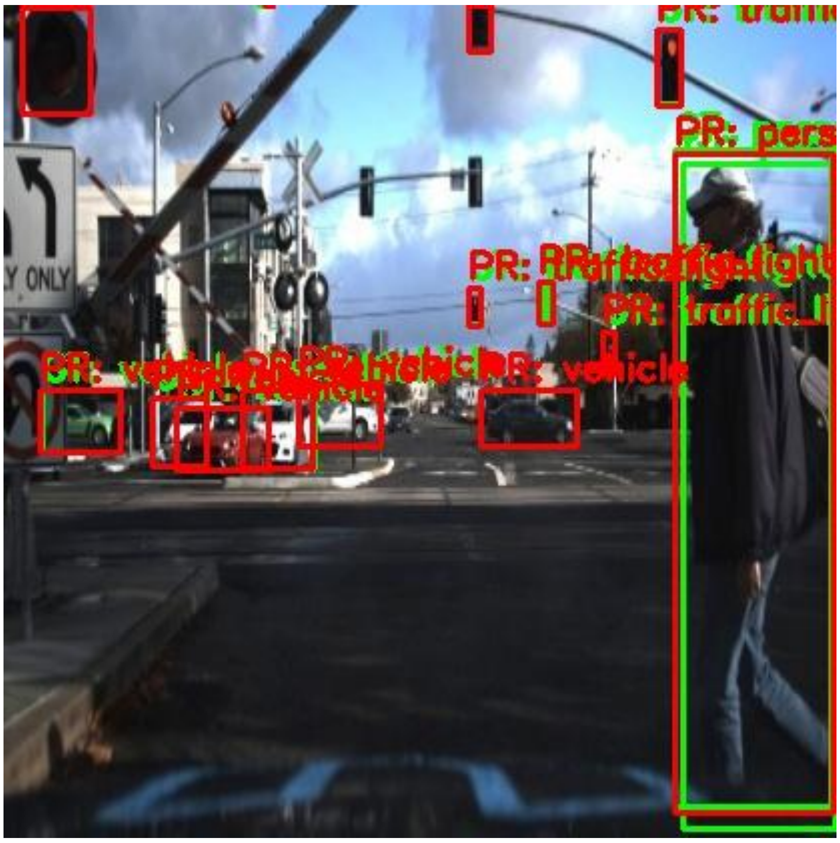

Figure 14 Second dataset image after transfer learning

The figures 15 and 16 show the object detection performance after applying the second approach in transfer learning, where the input images are automatically resized and padded by YOLO to a 640×640 resolution while preserving the original aspect ratio. The first image displays the detection results with bounding boxes and class names ("vehicle" and "person") on each object in the frames. The second image further includes the detection confidence scores alongside the predicted labels.

We can observe a consistent and accurate detection of multiple vehicles across various frames. The bounding boxes are tightly fit around the vehicles, and most predictions have very high confidence scores (often above 0.9), indicating the model's reliability in distinguishing objects in complex road scenes. Additionally, a few pedestrians are also detected with reasonable accuracy, confirming that the model has generalized beyond vehicles alone.

Compared to earlier qualitative results before transfer learning—where objects were misclassified or missed altogether—these images show a significant improvement in prediction robustness and label consistency. The automatic resizing and padding technique has clearly preserved spatial integrity, enabling better recognition performance on real-world sequential data.

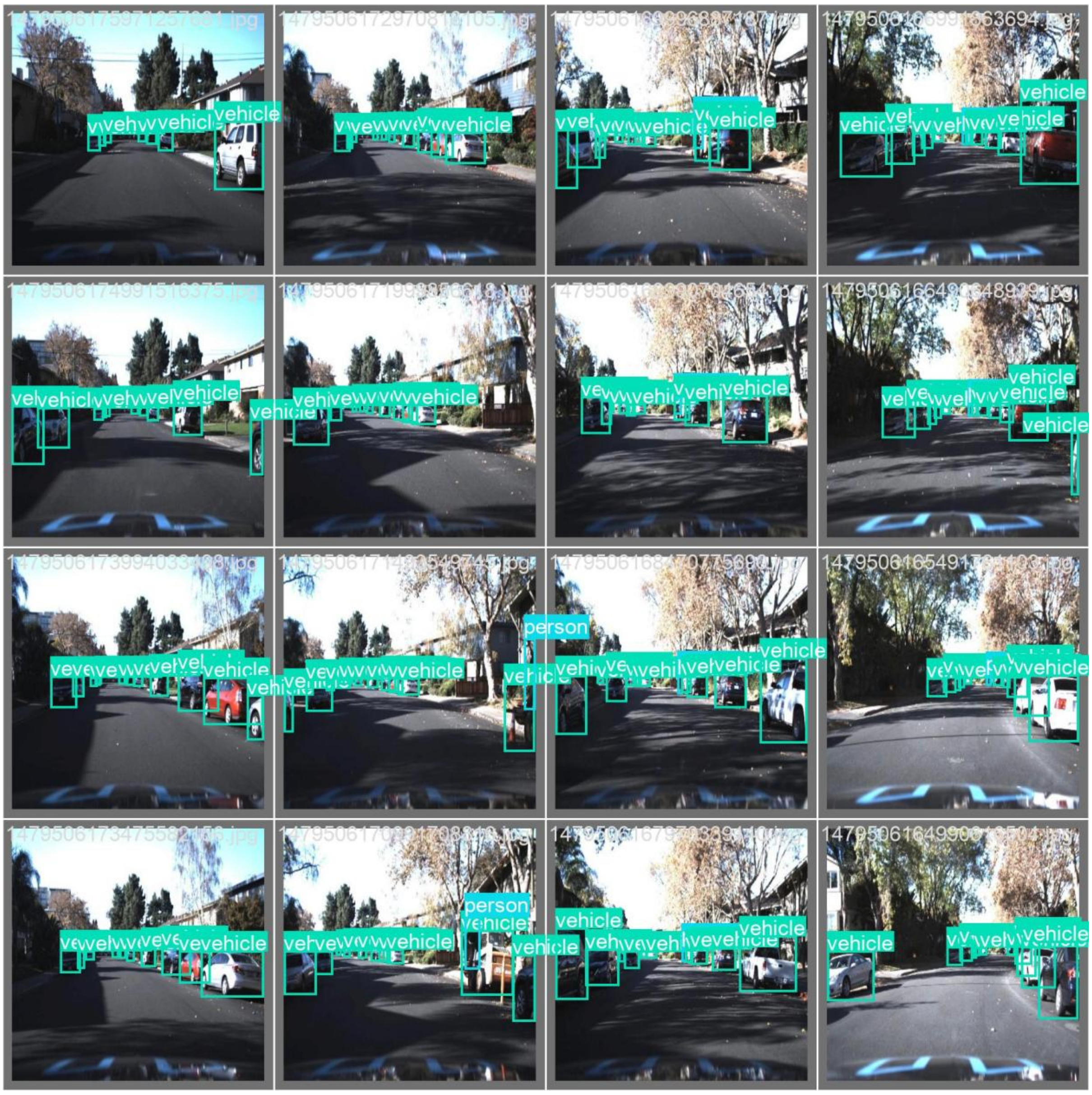

Figure 15 Actual labels from validation images

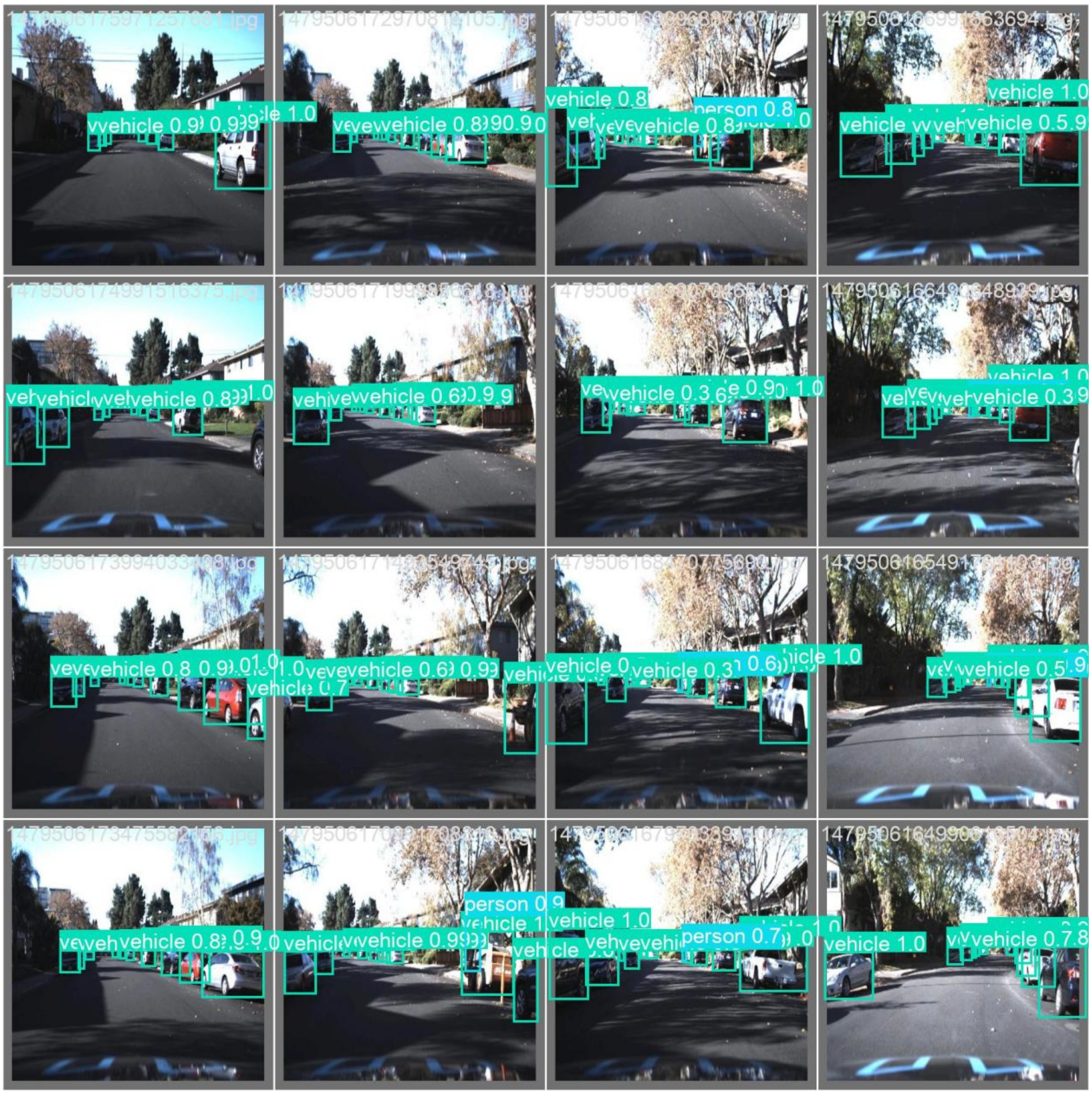

Figure 16 Predicted labels from validation images

4.2.2. Resizing and padding based on YOLO

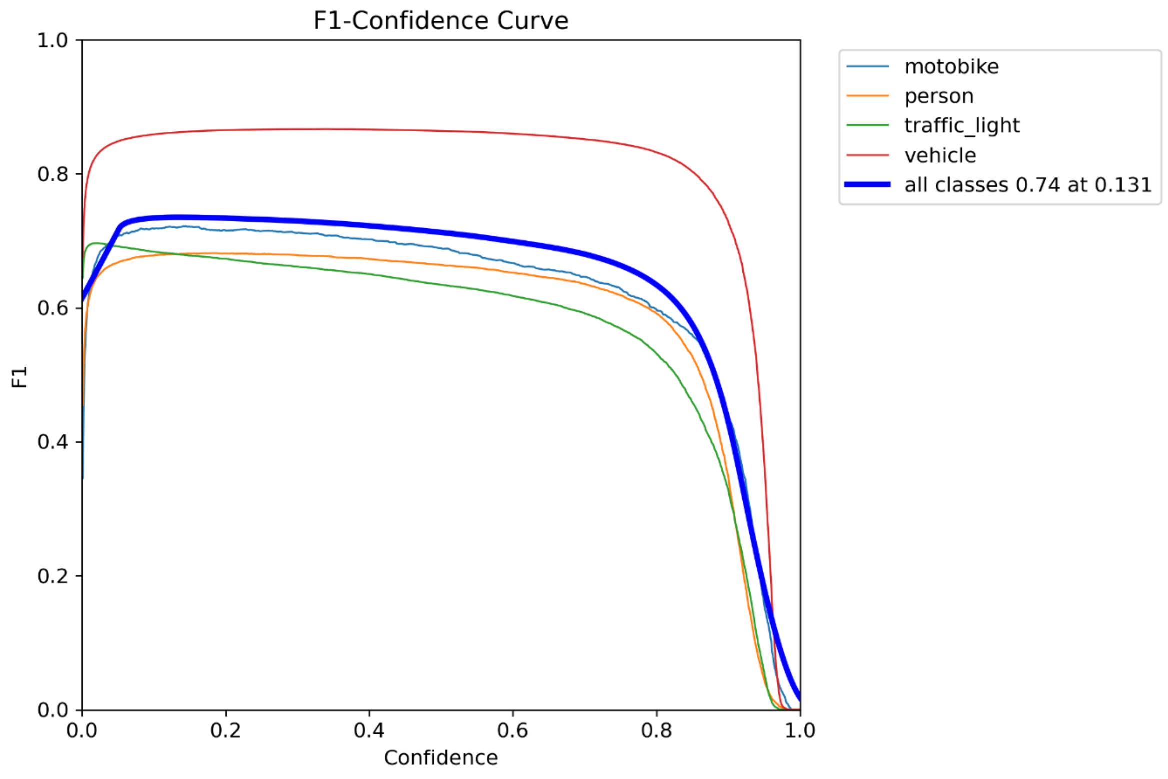

Figure 17 F1 curve

In this subchapter, the second transfer learning strategy is evaluated—resizing and padding images using YOLO's internal mechanism to a fixed resolution of 640×640 while maintaining the original aspect ratio. The F1-Confidence curve shown in this chapter can now be compared to the previous manual resizing approach.

Looking at the new F1-Confidence curve, we observe that the overall F1 score across all classes again peaks around 0.74, but this time at a lower confidence threshold of approximately 0.131 (compared to 0.238 in the manual approach). This shift indicates that the model becomes confident enough at an earlier threshold, which can be particularly beneficial for real-time inference in autonomous driving systems where rapid detection is critical.

Class-wise F1 trends are also improved or remain stable:

- Vehicle continues to outperform other categories, maintaining a strong and high F1 score across a wide range of confidence levels.
- Traffic_light and person, which showed weaker performance in the manual resizing approach, benefit more visibly from YOLO's padding strategy. Their F1 scores stay higher over a broader range of confidence thresholds, indicating better generalization.

- Motobike shows similar behavior but gains a smoother and slightly more stable curve.

In contrast to the manual resizing strategy, this approach avoids any distortion caused by stretching images to fit a fixed resolution. The use of padding ensures the spatial ratios are maintained, which improves the localization accuracy and results in more robust object detection. Overall, the second approach offers comparable or slightly improved performance in terms of F1 score while being more aligned with YOLO's intended preprocessing pipeline.

In figure 18, The second approach, which uses YOLO’s automatic resizing with padding to preserve aspect ratio, shows slight improvements in detecting motobikes, persons, and vehicles compared to the first method. The classification of persons becomes more accurate, likely due to better preservation of proportions. Vehicle detection also improves slightly, benefiting from the higher resolution. However, performance for detecting traffic lights drops, as padding may introduce noise or reduce clarity for small-scale objects. Overall, background confusion decreases slightly across most classes, indicating better separation between objects and non-object regions in the second approach.

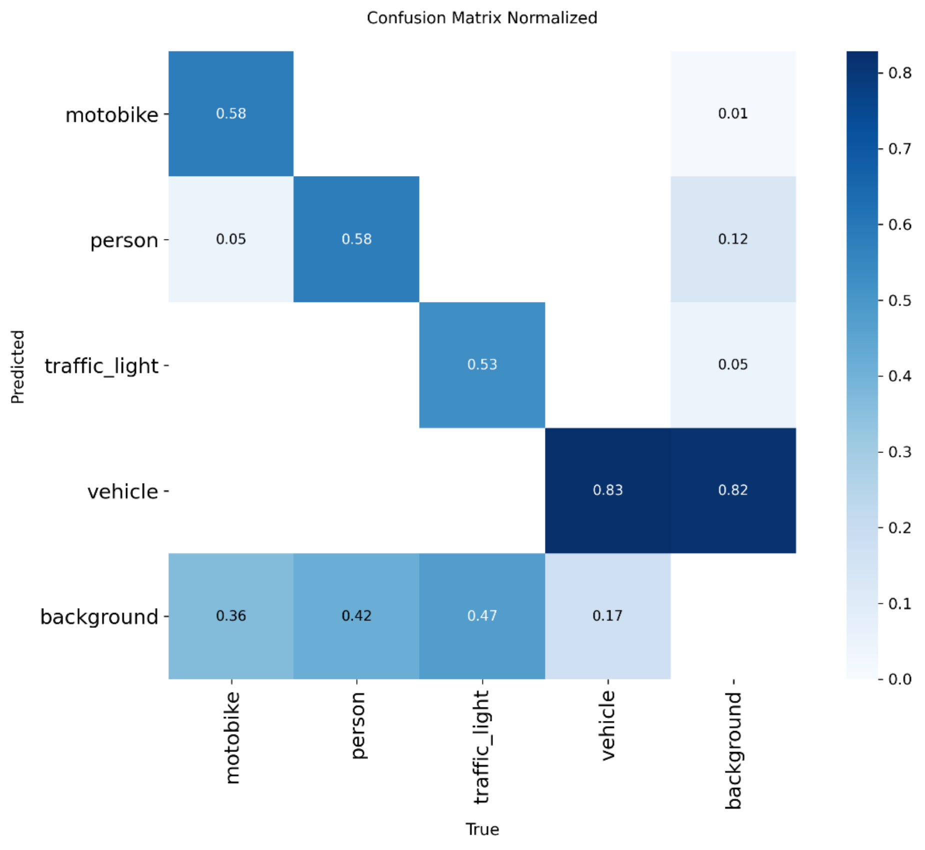

Figure 18 Confusion matrix

Figures 19, 20, and 21 further support the improved performance of the second approach in terms of bounding box accuracy. The object labels—particularly for vehicles and persons—are more consistently and precisely placed compared to the first approach. This improvement is largely due to the use of padded resizing, which preserves the original aspect ratio of the images and avoids distortion. However, for smaller objects like traffic lights, performance slightly worsens. This decline is likely because the added padding reduces the effective pixel density for these small features, making them harder for the model to detect and localize accurately.

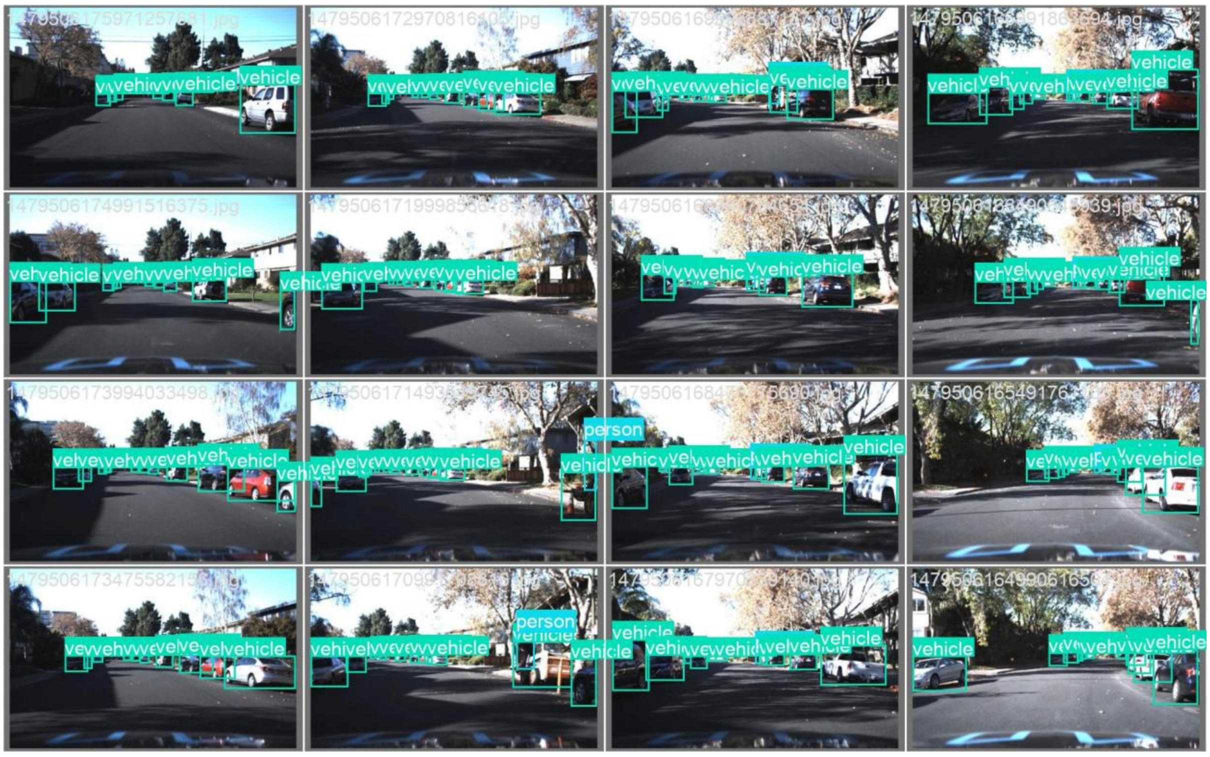

Figure 19 Actual labels from validation images

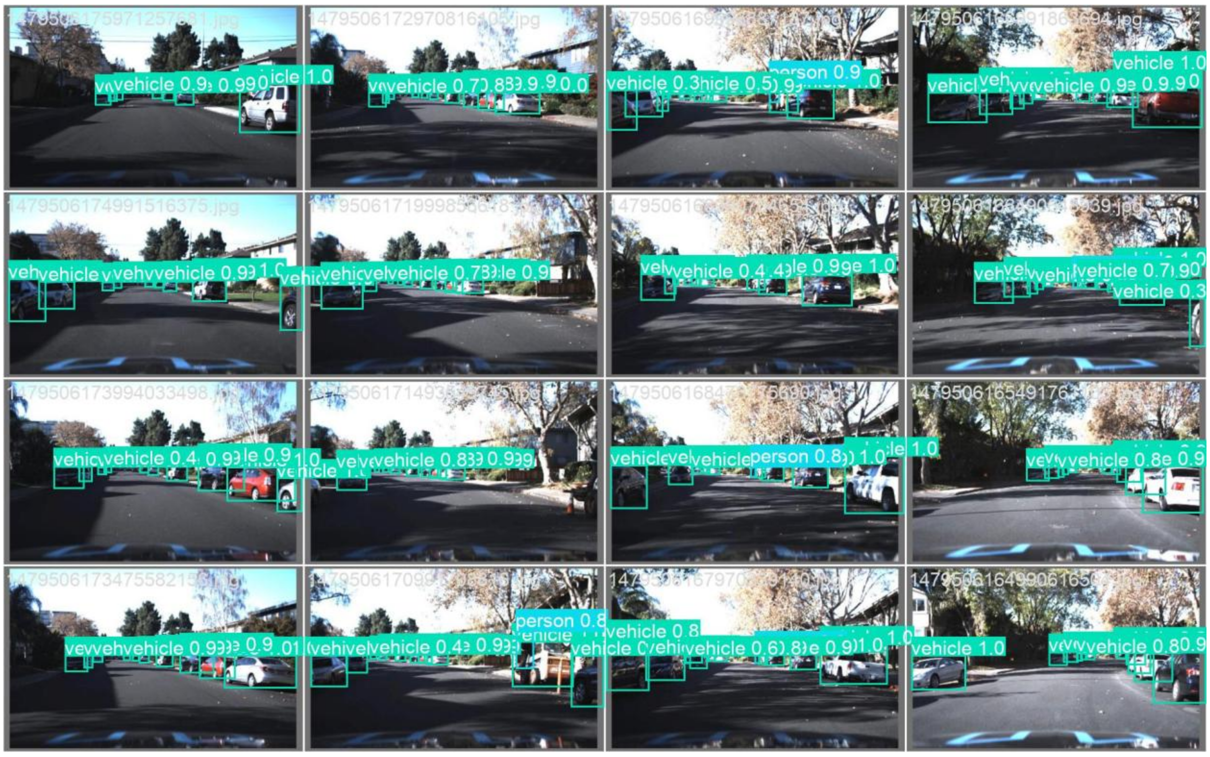
Figure 20 Predicted labels from validation images

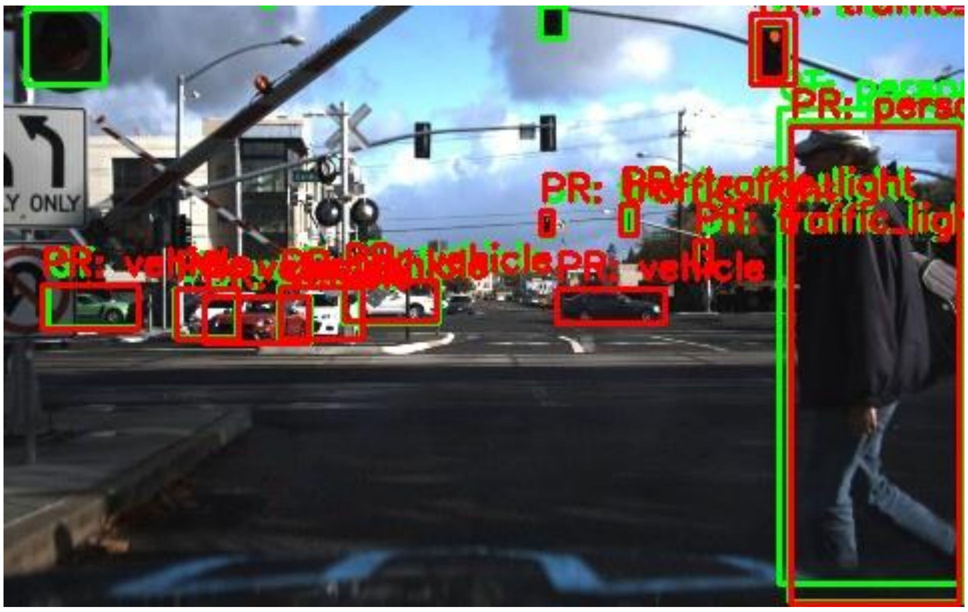
Figure 21 Second dataset image after transfer learning with keeping the ratio of image

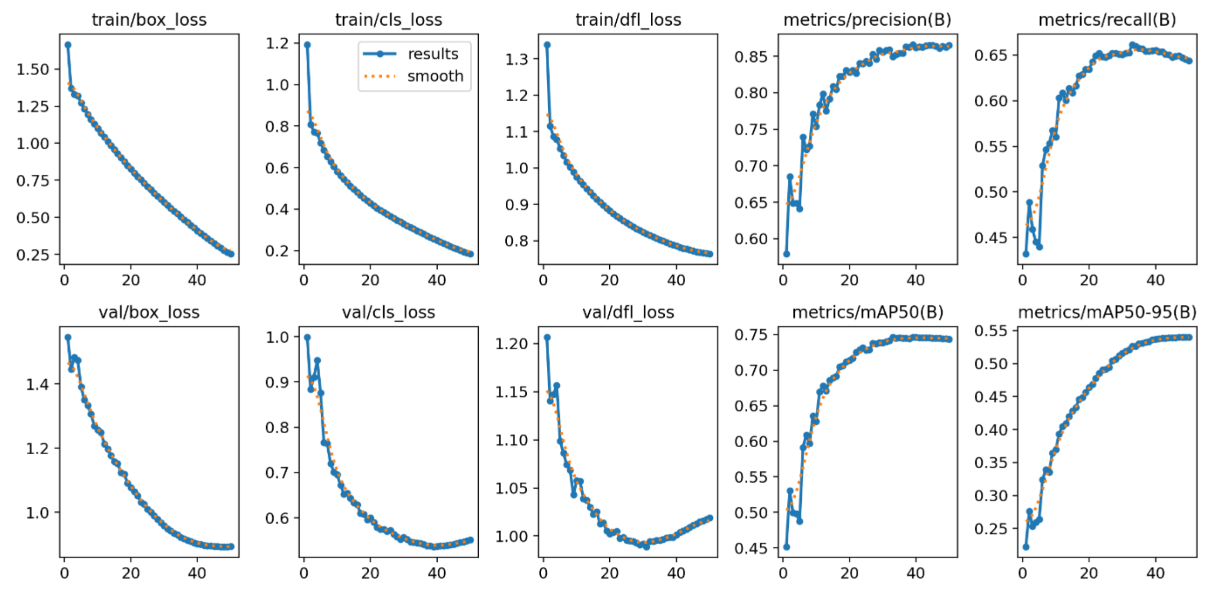

Figure 22 Overall result

4.3. Training on second data set without transfer learning and comparison

In this chapter, YOLOv8s is trained directly on the second dataset without leveraging the model trained on Dataset 1. While transfer learning can often accelerate convergence and improve generalization, this experiment is designed to assess the standalone performance of YOLOv8s when trained from scratch solely on Dataset 2, with no influence from prior training.

The dataset is used in its original format, and training is conducted over the same number of epochs and under similar hyperparameter conditions as the transfer learning approaches to ensure fair comparison. The goal is to evaluate whether a model trained entirely on real-world frames can reach comparable accuracy to models adapted from a more controlled dataset.

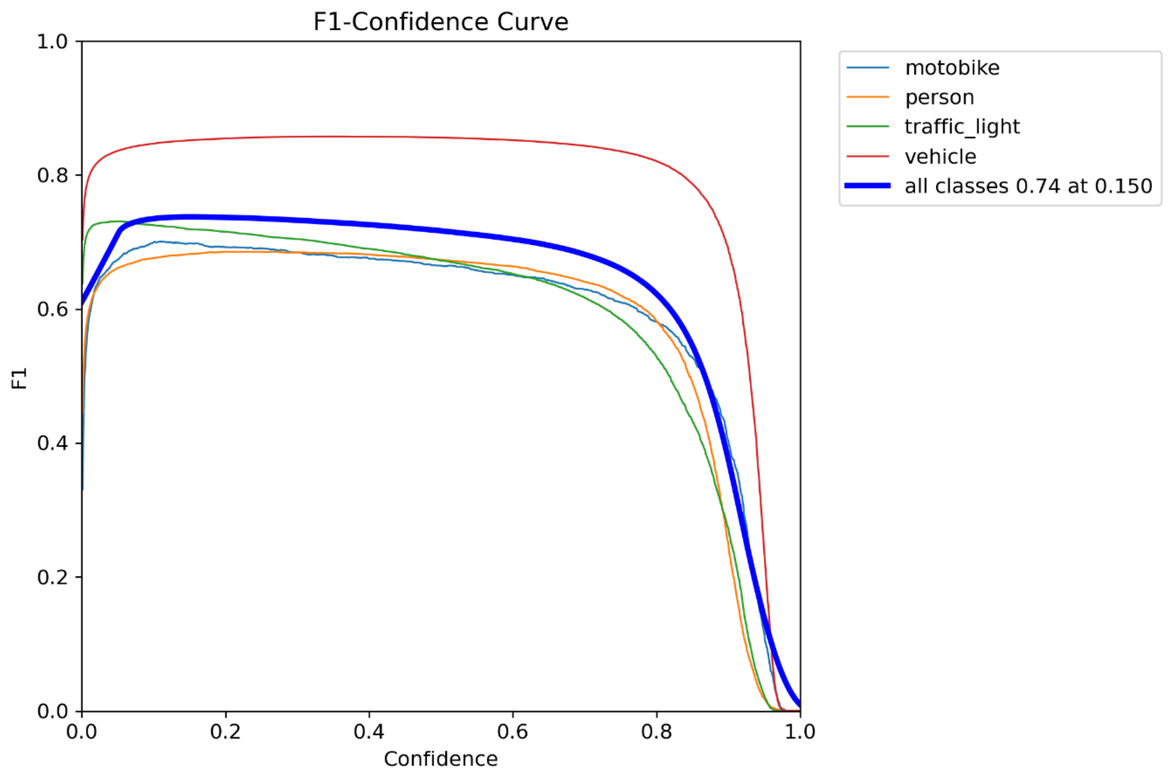

Figure 23 F1 curve

Figure 23 shows the F1-Confidence Curve of the model trained from scratch on Dataset 2. Compared to the curve obtained from the transfer learning approach, we observe that the overall shape remains consistent, and the vehicle class maintains a high F1 score. However, the F1 scores of other classes such as person and motobike slightly decrease, indicating that the previous model benefited from pre-learned features when adapted from Dataset 1.

This comparison suggests that although training from scratch on Dataset 2 is feasible and yields acceptable results, transfer learning provides a performance boost, especially in cases where class variation and edge cases are limited in the target dataset.

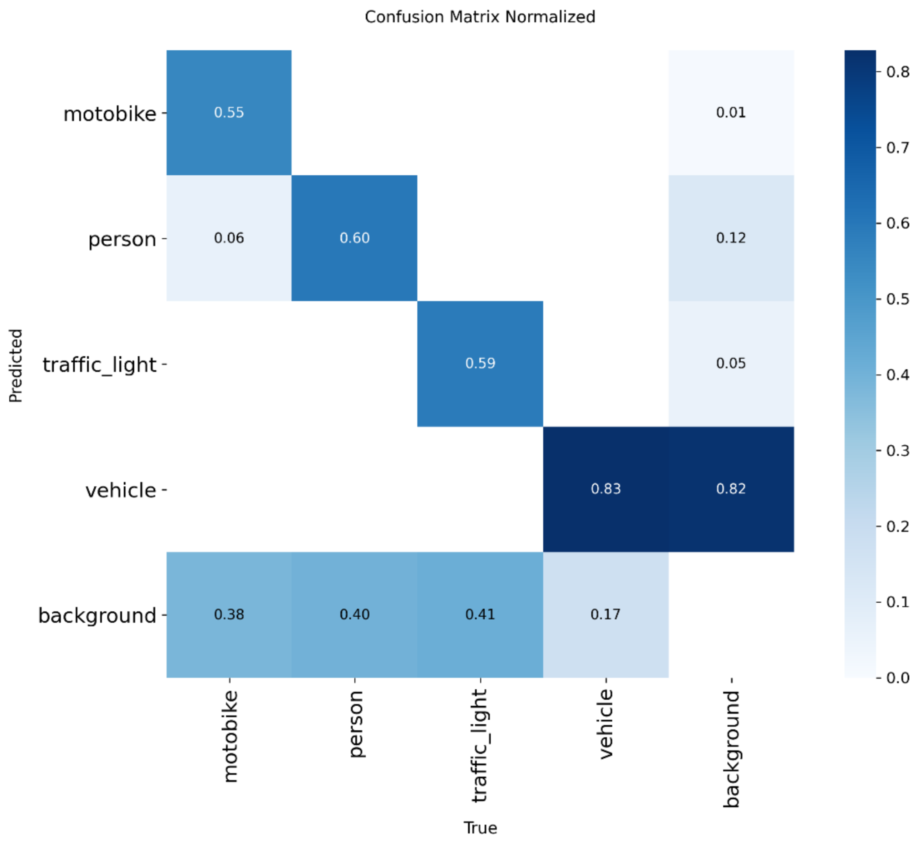

Figure 24 Confusion matrix

Comparing the confusion matrix of the model trained from scratch with the automatic resize and padding transfer learning approach shows that training from scratch delivers better overall performance. Notably, motobike and traffic light detection improves slightly, and class separation is generally cleaner. Although both models perform similarly in detecting vehicles, the scratch-trained model demonstrates better adaptation to the dataset, particularly in distinguishing persons.

In addition to accuracy, computational cost is an important factor. Despite transfer learning often being assumed more efficient, in this case, the training from scratch was faster, completing in 5 hours and 30 minutes, compared to 7 hours for the transfer learning setup. Both experiments used the same resolution, 4 workers, a batch size of 32, and 50 epochs. This highlights that transfer learning is not always more time-efficient, especially when the pre-trained model requires adaptation overhead.

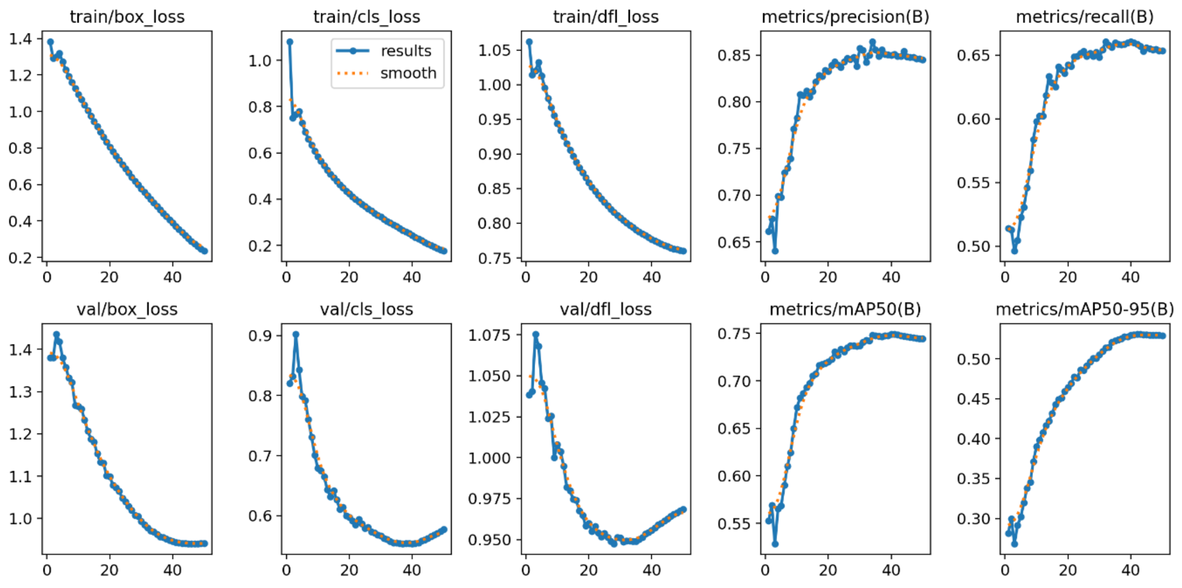

Figure 25 Overall Results of training from scratch

5. Conclusion and further work

In this study, multiple strategies were evaluated for training a YOLOv8 object detection model on a second dataset using transfer learning and training from scratch. Two transfer learning approaches were tested: (1) resizing images to a fixed size with corrected labels, and (2) automatic resizing and padding based on YOLO's native preprocessing. These were compared against training a YOLOv8s model from scratch.

The results show that transfer learning improves performance compared to no pretraining, especially when the input resolution is adapted appropriately. However, training from scratch ultimately yielded the best detection accuracy, particularly for challenging classes such as motobike and traffic_light. Additionally, it proved to be more computationally efficient, taking less training time under identical hardware and batch settings.

For future work, it is recommended to explore larger YOLOv8 variants such as YOLOv8m or YOLOv8l, as the dataset size justifies the use of more complex models. However, due to GPU memory and compute limitations, this project was conducted using the lightweight YOLOv8s architecture.