Fine-Tuning YOLOv8 for Traffic Sign Detection on the TT100K-4 Dataset

Syed Muhammad Mehran 24L-8014
FAST National University of Computing and Emerging Sciences Email: smehran.mme@gmail.com

***Abstract*—This paper investigates the fine-tuning of the YOLOv8m model for traffic sign detection on the TT100K-4 dataset. The primary goal was to evaluate the impact of batch size and learning rate on model performance and to compare the results before and after fine-tuning. Initially, the YOLOv8m model demonstrated suboptimal performance with precision and recall values of 59.49% and 18.07%, respectively. After fine-tuning with a batch size of 24 and a learning rate of 0.001, the model achieved significant improvements, reaching a precision of 81.87%, recall of 56.24%, and mAP@50 of 70.09%. These results highlight the importance of hyperparameter optimization for improving detection accuracy in traffic sign recognition, which is crucial for real-time applications in autonomous driving systems. Qualitative and quantitative analysis of the model's bounding box predictions further confirmed the effectiveness of fine-tuning. This study demonstrates that fine-tuning YOLOv8m with appropriate hyperparameters can significantly enhance its performance in traffic sign detection, offering valuable insights for future applications in autonomous vehicles and related fields.**

I. INTRODUCTION

Object detection is a fundamental problem in computer vision, playing a crucial role in numerous applications, including surveillance, autonomous driving, and industrial automation. Among these, traffic sign detection is of paramount importance, as it serves as the backbone for intelligent transportation systems (ITS). Accurate recognition and localization of traffic signs are essential for enhancing road safety, supporting driver assistance systems, and enabling autonomous vehicles to make informed decisions. Traditional computer vision techniques have struggled with challenges such as varying illumination, occlusion, and diverse environmental conditions, necessitating the development of robust deep learning-based approaches.

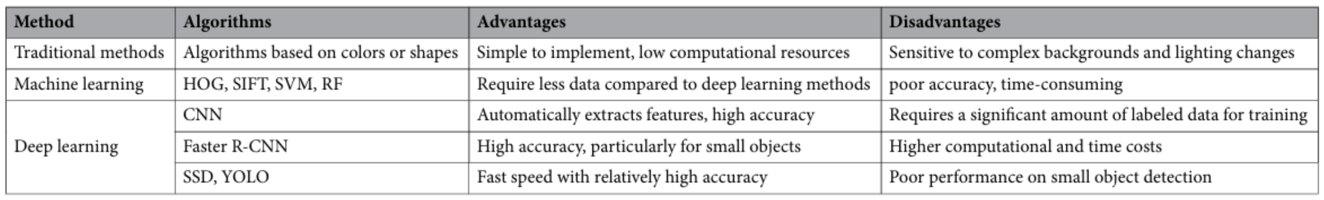
Various object detection approaches have been proposed to tackle traffic sign detection, ranging from classical methods utilizing handcrafted features and traditional machine learning classifiers to modern deep learning-based models. Early techniques relied on edge detection, color segmentation, and shape analysis to identify traffic signs, but they often failed under real-world conditions with poor lighting, motion blur, or complex backgrounds. Deep learning algorithms utilize neural networks to model intricate relationships between inputs and outputs. These algorithms have gained popularity in traffic sign recognition due to their ability to autonomously learn high-level features directly from raw data. This capability significantly diminishes the necessity for manual feature extraction, streamlining the process and enhancing the effectiveness of recognition systems. The advent of deep convolutional neural networks (CNNs) revolutionized object detection, with two primary paradigms emerging: two-stage detectors like Faster R-CNN and one-stage detectors like SSD and YOLO. While two-stage detectors provide high accuracy, they tend to be computationally expensive. In contrast, single-stage detectors, particularly the YOLO (You Only Look Once) family, have gained widespread popularity due to their balance between speed and accuracy, making them highly suitable for real-time applications.

The YOLO object detection framework has evolved significantly since its inception, becoming one of the most widely used architectures for real-time detection tasks. YOLO models process an entire image in a single forward pass, enabling fast and efficient object detection. The latest iteration, YOLOv8, builds upon previous versions with improvements in architecture, anchor-free detection, and enhanced feature extraction. YOLOv8 employs a CSPDarknet backbone for feature representation, a PANet-based neck for multi-scale feature aggregation, and a decoupled head for refined classification and localization. These advancements make YOLOv8 particularly well-suited for traffic sign detection, where high accuracy and real-time processing are essential.

II. RELATED WORKS

- A number of studies has been conducted that focuses on traffic sign detection using deep learning and yolo models. Li and Wang [1] combined Faster R-CNN with MobileNets to precisely locate and classify small traffic signs. This innovative approach leveraged the strengths of both technologies: Faster R-CNN for its efficient and accurate detection capabilities and MobileNets for its lightweight, mobile-friendly architecture, resulting in enhanced performance in recognizing smaller traffic signs. Tabernik and Skočaj [2] enhanced the Mask R-CNN framework to better recognize small traffic signs and introduced a novel data augmentation technique to improve the model's generalization capabilities. Evaluations on both the DFG and the Swedish traffic sign datasets demonstrated significant performance gains, with the refined Mask R-CNN model achieving metrics such as mAP50 up to 95.5%.

The YOLO family has emerged as a leading framework for real-time traffic sign detection. Wang et al. [3] developed an enhanced lightweight traffic sign recognition algorithm based on YOLOv4-Tiny. The algorithm refines the K-means clustering method to generate anchor frames tailored to the traffic sign dataset, which significantly improves detection recall and target localization precision. When evaluated on



Method	Algorithms	Advantages	Disadvantages
Traditional methods	Algorithms based on colors or shapes	Simple to implement, low computational resources	Sensitive to complex backgrounds and lighting changes
Machine learning	HOG, SIFT, SVM, RF	Require less data compared to deep learning methods	poor accuracy, time-consuming
Deep learning	CNN	Automatically extracts features, high accuracy	Requires a significant amount of labeled data for training
	Faster R-CNN	High accuracy, particularly for small objects	Higher computational and time costs
	SSD, YOLO	Fast speed with relatively high accuracy	Poor performance on small object detection

Fig. 1: Pros and Cons of Different Object Detection Models

the TT100K dataset, the improved algorithm achieved a mean Average Precision (mAP) at 0.5 of 52.07% and demonstrated enhanced real-time performance. Dewi et al. [4] combined YOLOv3 and Densenet models, incorporating SPP to optimize the feature extraction. This innovation significantly boosted the recognition accuracy of small traffic signs.

YOLOv8 introduces multiple architectural enhancements, including anchor-free detection, transformer-based attention, and an improved backbone network. Redmon et al. [5] highlighted its advantages in detecting small objects with higher precision.

In the study by Galkin et al. [6], the authors present a comprehensive evaluation of the YOLOv8 object detection architecture, emphasizing its performance improvements over previous YOLO versions. YOLOv8 introduces an anchor-free detection head, decoupled head architecture for classification and regression, and enhanced backbone modules, including C2f modules for better feature representation. These modifications contribute to both higher accuracy and faster inference.

The experimental evaluation demonstrates YOLOv8's superior performance across multiple benchmarks. The authors also evaluate the models on PASCAL VOC, where YOLOv8s achieves 81.6% mAP@0.5 compared to YOLOv5s's 78.2%, further validating the architectural enhancements. These results affirm YOLOv8's state-of-the-art status for real-time object detection tasks and its suitability for deployment in resource-constrained environments.

The paper titled "YOLO-TS: A Lightweight YOLO Model for Traffic Sign Detection" [7] proposes a computationally efficient version of the YOLOv5 architecture—called YOLO-TS—tailored for traffic sign detection in resource-constrained environments. By replacing the backbone with ShuffleNetV2 and simplifying the neck structure, the model significantly reduces computational load and parameter size while maintaining competitive accuracy. Evaluated on the TT100K dataset, YOLO-TS achieves a mAP of 74.1% with only 3.7M parameters and 4.3G FLOPs, making it 82.7% smaller and 83.6% faster than the original YOLOv5s.

Fine-tuning YOLOv8 on the TT100K-4 dataset builds upon these advancements by optimizing the model for real-world traffic environments.

III. METHODOLOGY

A. *Dataset*

The TT100K (Tsinghua-Tencent 100K) dataset is one of the largest publicly available traffic sign datasets, containing 100,000 images collected from Chinese roads. It includes over 30,000 annotated traffic signs spanning 221 categories, making it a diverse benchmark for traffic sign detection and classification tasks. The dataset captures real-world driving scenarios, including varying weather conditions, lighting variations, occlusions, and motion blur, making it highly challenging for object detection models.

TT100K-4 Subset The TT100K-4 dataset is a filtered and optimized subset of the original TT100K dataset, designed to focus on four major traffic sign categories that are critical for autonomous driving and intelligent transportation systems. These categories include:

Speed Limit Signs – Indicating regulatory speed limits.

Prohibitory Signs – Representing restrictions such as "No Entry" or "No U-turn."

Mandatory Signs – Providing instructions like "Keep Left" or "Turn Right."

Warning Signs – Alerting drivers to potential hazards, such as sharp curves or pedestrian crossings.

The TT100K-4 dataset maintains the high-resolution image quality of the original dataset while improving class balance and reducing redundancy. It serves as an ideal benchmark for fine-tuning real-time object detection models like YOLOv8, ensuring robust performance in traffic sign recognition tasks.

Fine-tuning object detection models on TT100K-4 poses multiple challenges, including: small object detection, environmental variability and class imbalance.

B. *Architecture of YOLOv8m*

YOLOv8m (medium-sized YOLOv8 model) is a state-of-the-art anchor-free object detection model that balances accuracy and efficiency. It builds upon previous YOLO versions with several key architectural improvements, making it highly suitable for real-time traffic sign detection. The model consists of three major components:

Backbone: A CSP-DarkNet-based feature extractor optimized with ELAN (Efficient Layer Aggregation Network) to capture multi-scale features efficiently.

Neck: A bi-directional feature pyramid network (BiFPN) that enhances feature fusion across different scales.

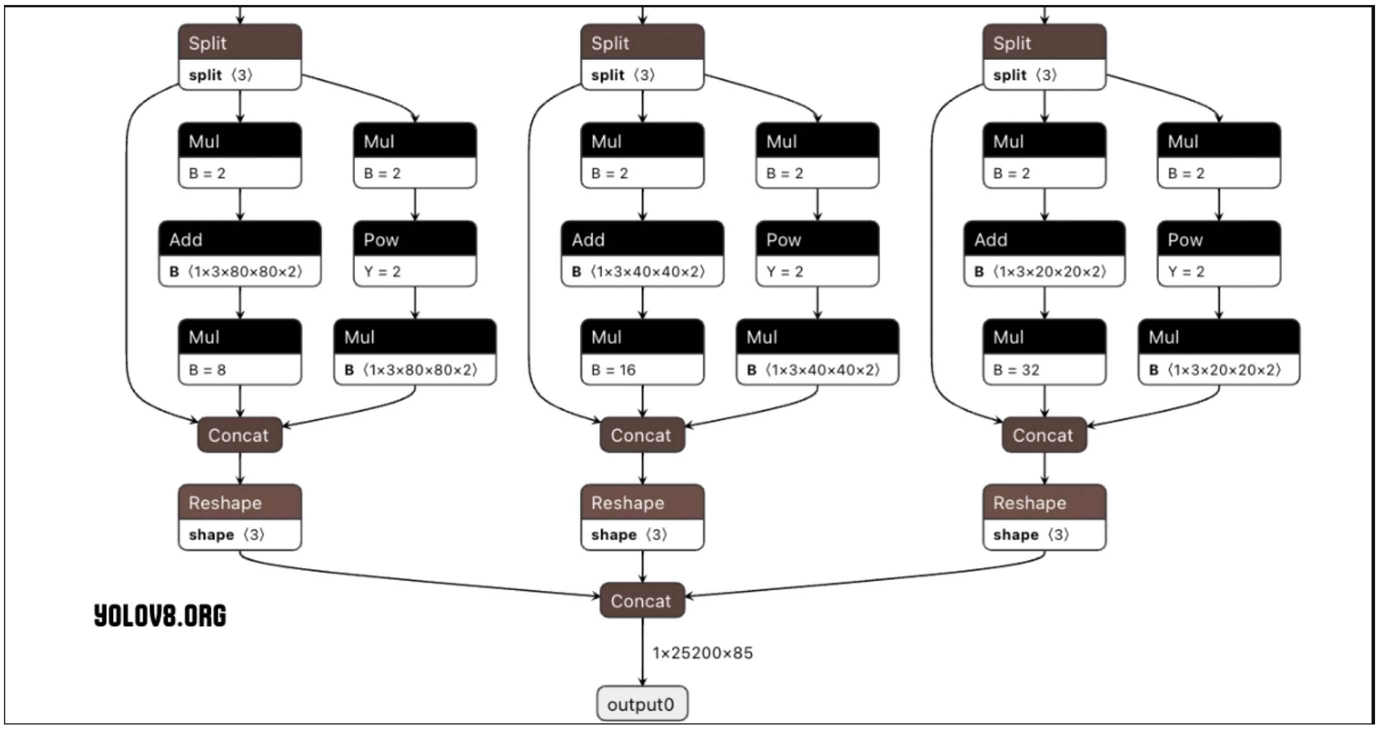


Fig. 2: Architecture of YOLOv8m, consisting of Backbone, Neck, and Detection Head.

Head: A decoupled detection head that predicts class probabilities, objectness scores, and bounding box coordinates separately for improved accuracy.

1) Why YOLOv8m for Traffic Sign Detection?: Better Small Object Detection: Traffic signs are often small in images, and YOLOv8m's improved neck (BiFPN) and anchor-free mechanism enhances small object detection.

Real-Time Performance: Running at 130 FPS, it is suitable for real-time autonomous driving applications.

Balanced Speed-Accuracy Trade-off: Unlike YOLOv8s (faster but less accurate) and YOLOv8l (accurate but slower), YOLOv8m is optimized for high accuracy while maintaining real-time performance.

C. Error Metrics

To evaluate the performance of the fine-tuned YOLOv8m model on the TT100K-4 dataset, we use standard object detection metrics. These metrics provide insights into the model's accuracy, robustness, and generalization capabilities in detecting traffic signs under real-world conditions.

A. Precision (P): Precision measures the proportion of correctly identified traffic signs out of all predicted signs. A high precision score indicates that the model produces fewer false positives.

$$P = \frac{TP}{TP + FP} \tag{1}$$

TP (True Positives) = Correctly detected traffic signs FP (False Positives) = Incorrectly detected objects

B. Recall: Recall measures the proportion of correctly detected traffic signs out of all actual traffic signs present in the dataset. A high recall score indicates that the model detects most traffic signs but may produce more false positives.

$$P = \frac{TP}{TP + FN} \tag{2}$$

FN (False Negatives) = Missed traffic signs

C. Mean Average Precision (mAP@50 and mAP@50-95):

Mean Average Precision (mAP) is a standard metric for object detection that measures precision-recall trade-offs. It calculates the area under the Precision-Recall (PR) curve for different Intersection over Union (IoU) thresholds.

mAP@50 (IoU = 50%): Measures how well the model detects objects with at least 50% overlap between predicted and ground truth bounding boxes.

mAP@50-95 (IoU = 50% to 95%): Measures how well the model detects objects across a range of IoU thresholds (50% to 95%), making it a stricter metric.

where AP (Average Precision) is calculated for each class and then averaged.

$$\text{mAP} = \frac{1}{n}\sum_{i=1}^{n} \text{AP}_i \tag{3}$$

D. Fitness Score: The fitness score is a composite metric used in YOLO models to optimize training performance. It combines multiple evaluation metrics (such as precision, recall, and mAP) into a single value, guiding the best checkpoint selection.

TABLE I: Comparison of YOLO Models on Key Metrics

Model	Parameters (M)	FLOPs (B)	Speed (FPS)	mAP@50 (%)	Best Use Case
YOLOv5m	21.2	52.9	140	48.2	General object detection
YOLOv6m	34.2	89.4	120	51.0	Faster real-time detection
YOLOv8m	25.9	75.0	130	52.3	Balanced speed-accuracy
YOLOv8s	11.2	32.4	160	50.5	Faster but less accurate
YOLOv8l	43.7	105.3	100	53.7	Higher accuracy, slower speed

IV. RESULTS

The primary objective of this study was to fine-tune the YOLOv8m model on the TT100K-4 dataset and analyze the impact of batch size and learning rate (lr0) on model performance. The model's performance was evaluated before and after fine-tuning using key object detection metrics, including precision, recall, mean average precision (mAP@50, mAP@50-95), and fitness score.

The results obtained in different stages of training are summarized below:

A. *Before Fine Tuning*

Before fine-tuning, the YOLOv8m model was tested on the TT100K-4 dataset without any modifications. The results indicate poor recall (18.07%) and low mAP scores, indicating that the model struggled to detect traffic signs effectively in its default state.

B. *Fine Tuning at Batch Size=16 and Learning Rate=0.1*

The model was first fine-tuned with a batch size of 16 and an initial learning rate (lr0) of 0.1. The results showed a significant improvement in recall (+29.05%) and mAP scores, confirming that the model was learning traffic sign patterns more effectively. However, precision slightly dropped due to increased false positives. While recall improved significantly compared to the model before fine tuning, the slight drop in precision suggests that the model started detecting more objects but at the cost of increased false positives.

C. *Fine Tuning at Batch Size=24 and learning rate=0.001*

To further optimize the model, the batch size was increased to 24, and a smaller learning rate of 0.001 was applied. This setup yielded the best results, achieving the highest precision (81.87%), recall (56.23%), and mAP scores. This configuration not only improved recall and precision but also significantly enhanced mAP@50 (+53.98%) and mAP@50-95 (+43.36%) compared to the pre-fine-tuning results. The increase in precision suggests that the model was not only detecting more signs but also reducing false positives.

V. DISCUSSION

Before fine tuning, model performed poorly for all performance metrics as it used pre-trained weights which were not optimized for TT100k dataset. Fine-tuning allowed the model to adjust to the specific characteristics of traffic signs in China (TT100K dataset), improving detection accuracy. This is illustrated in figure 2, where bounding box generated before and after fine tuning are compared. In figure 2(a), the model detects a traffic sign object but label it as a clock and only has a confidence score of 0.27. This is because model is pre-trained on a generalized dataset. After fine tuning, model is optimized on our specified traffic sign detection dataset which allows it to correctly identify the object with a high confidence score of 0.83. Similarly, the dramatic +211.19% increase in recall suggests that fine-tuning helped the model detect previously missed small-scale or low-contrast signs. The improved IoU thresholds in mAP@50-95 indicate that the model now generates better bounding box predictions.

A. *Effect of Learning Rate and Batch Size*

Fine-tuning deep learning models, particularly YOLOv8m, requires careful selection of hyperparameters such as learning rate (lr0) and batch size (BS). These parameters directly impact the convergence speed, generalization ability, and stability of the model during training. The results in this study illustrate how different configurations affect object detection performance.

*1) **Learning Rate:*** The learning rate determines how much the model updates its weights after each training iteration. A poorly chosen learning rate can lead to suboptimal training, affecting detection accuracy. At high learning rate of 0.1, the model converge quickly by making large updates to the weights. This resulted in a significant boost in recall (47.12%), indicating that the model learned to detect more objects. However, precision slightly dropped (58.57%) compared to the pre-trained model (59.49%), suggesting an increase in false positives. The model was adapting too aggressively, causing slight overfitting to certain sign patterns, leading to misclassifications. A lower learning rate (0.001), on the other hand, led to more gradual weight updates, allowing the model to stabilize and generalize better to new images. This approach yielded the best precision (81.87%) and a balanced recall (56.24%), indicating an optimal trade-off between false positives and false negatives. The slower learning rate reduced noisy updates in weight adjustments, enabling more refined feature learning, especially for small and complex signs. The mAP@50-95 increased significantly to 55.89%, demonstrating that the model performed well across varying IoU thresholds.

*2) **Batch Size:*** Batch size defines how many images are processed before updating the model weights. It influences training stability, memory usage, and convergence behavior. At smaller batch size of 16, model updated weights more frequently, leading to faster convergence. However, the frequent updates caused higher variance in gradient estimation, leading to unstable training, as reflected in the lower precision (58.57%). The model learned rapidly but lacked generalization, leading to more misclassifications.

TABLE II: Performance Comparison Before and After Fine-Tuning

Metric	Before Fine-Tuning	After Fine-Tuning (BS=16, lr0=0.1)	After Fine-Tuning (BS=24, lr0=0.001)
Precision (B)	0.5949 (59.49%)	0.5857 (58.57%)	0.8187 (81.87%)
Recall (B)	0.1807 (18.07%)	0.4712 (47.12%)	0.5624 (56.24%)
mAP@50 (B)	0.1710 (17.10%)	0.5069 (50.69%)	0.7009 (70.09%)
mAP@50-95 (B)	0.1253 (12.53%)	0.4010 (40.10%)	0.5589 (55.89%)
Fitness Score	0.1299 (12.99%)	0.4116 (41.16%)	0.5731 (57.31%)

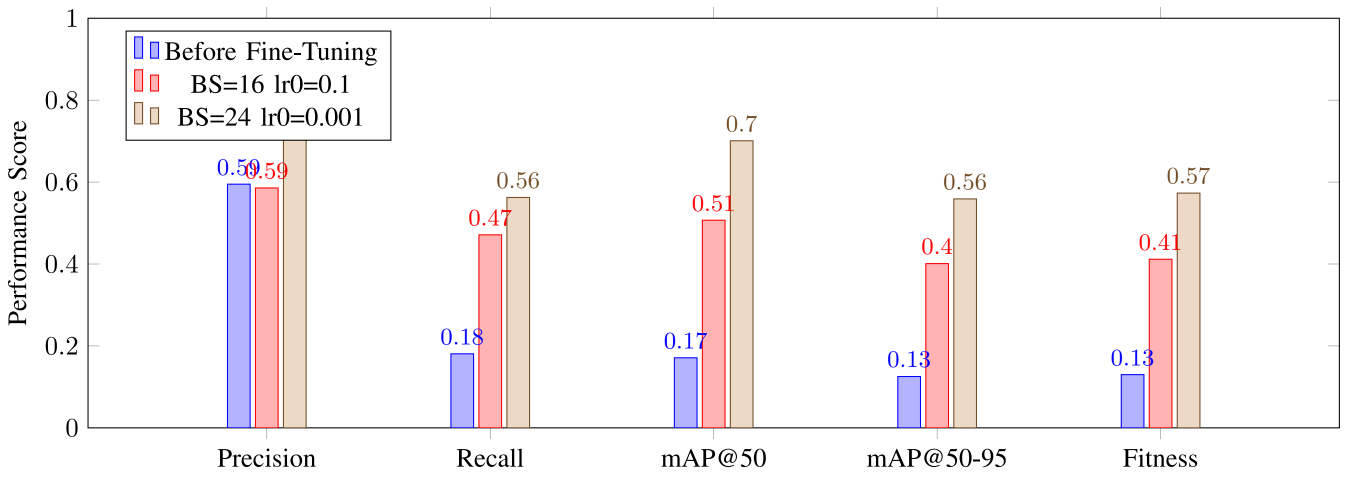


Fig. 3: Performance comparison before and after fine-tuning YOLOv8m on the TT100K-4 dataset.

A larger batch size (24) provided a more stable gradient estimation, leading to smoother convergence. The increased recall (56.24%) suggests that the model was able to detect a greater variety of traffic signs. A larger batch size also improved mAP scores, allowing the model to make more accurate bounding box predictions. This batch size effectively balanced computational efficiency and model generalization, making it the optimal setting for real-world deployment.

VI. CONCLUSION

In this research, we have fine-tuned the YOLOv8m model on the TT100K-4 dataset to enhance the performance of traffic sign detection. The primary objective was to investigate the impact of different batch sizes and learning rates on the model's effectiveness in recognizing traffic signs in images. Our study showed that fine-tuning the model significantly improved its performance across key metrics, such as precision, recall, mAP@50, and fitness scores, as compared to the pre-fine-tuning results.

Initially, the YOLOv8m model showed relatively low performance with precision at 59.49% and recall at 18.07%. After fine-tuning, we observed an enhancement in model performance, with the highest improvement seen at a batch size of 24 and learning rate of 0.001. In this configuration, the model achieved a precision of 81.87%, recall of 56.24%, and an mAP@50 of 70.09%. These results clearly demonstrate that fine-tuning, especially with optimal hyperparameters, can dramatically improve the detection accuracy for traffic sign detection, which is crucial for real-time applications like autonomous driving systems.

The results highlight the importance of carefully selecting hyperparameters such as batch size and learning rate. Batch sizes of 16 and 24 both showed improvement in model performance, but the best results came from using a smaller learning rate of 0.001, which allowed the model to converge more effectively. This insight is valuable for future implementations in similar tasks, where hyperparameter optimization is key to maximizing performance.

In addition to the quantitative improvements, the qualitative results, as shown by side-by-side comparisons of bounding box predictions before and after fine-tuning, visually corroborated the effectiveness of our fine-tuning approach. The model's bounding boxes became more precise and consistent after fine-tuning, further affirming its capability to handle complex traffic sign detection tasks.

This work underscores the potential of YOLOv8m as an effective solution for traffic sign detection, which is essential for advancing autonomous systems and improving traffic safety. Future work could explore additional fine-tuning strategies, including data augmentation techniques, further optimization of hyperparameters, and even domain adaptation to handle more diverse traffic sign datasets from different regions.

Through this study, we have demonstrated that with the right fine-tuning process, YOLOv8m can significantly enhance its performance on specific detection tasks, contributing valuable insights for future machine learning applications in the field of computer vision and autonomous driving.

REFERENCES

[1] J. Li and Z. Wang, "Real-time traffic sign recognition based on efficient cnns in the wild," *IEEE Transactions on Intelligent Transportation*

Systems, vol. 20, no. 3, pp. 975–984, 2018.

[2] D. Tabernik and D. Skočaj, "Deep learning for large-scale traffic-sign detection and recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1427–1440, 2019.

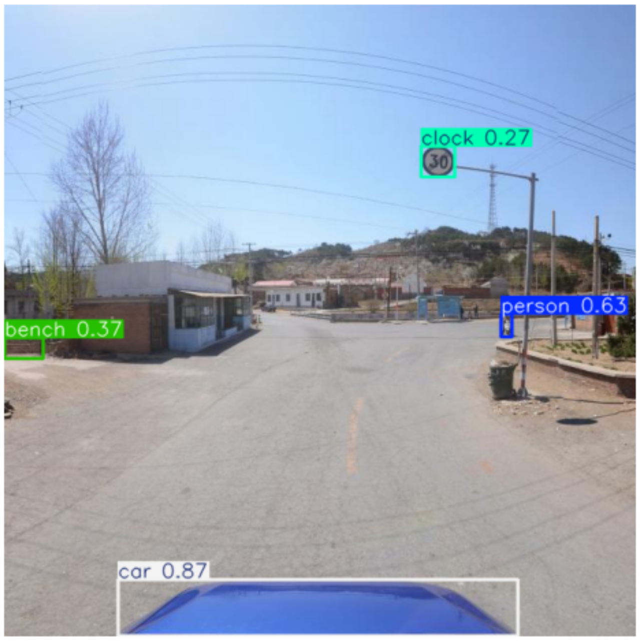
[3] L. Wang, K. Zhou, A. Chu, G. Wang, and L. Wang, "An improved light-weight traffic sign recognition algorithm based on yolov4 tiny," *IEEE Access*, vol. 9, pp. 124 963–124 971, 2021.

[4] C. Dewi, R.-C. Chen, H. Yu, and X. Jiang, "Robust detection method for improving small traffic sign recognition based on spatial pyramid pooling," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 7, pp. 8135–8152, 2023.

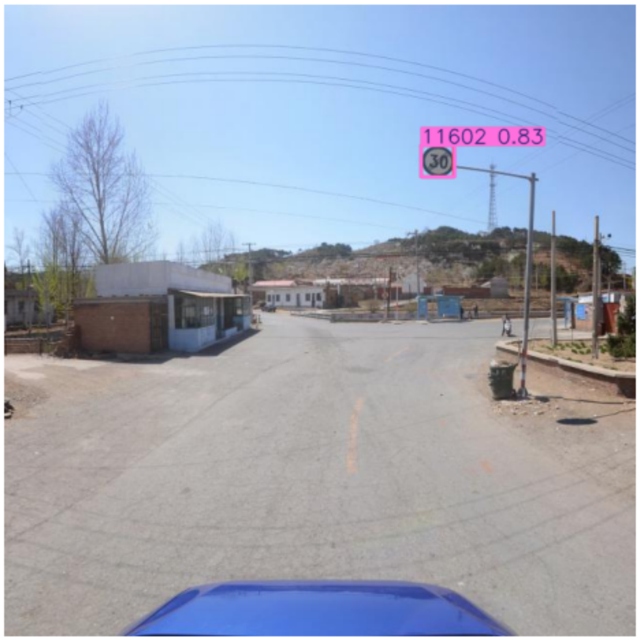
[5] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263–7271.

[6] A. Galkin, M. Lagunov, A. Pugach, and A. Sozykin, "YOLOv8: Improvements and Performance Evaluation for Object Detection Tasks," *Applied Sciences*, vol. 14, no. 1, pp. 1–21, Jan. 2024.

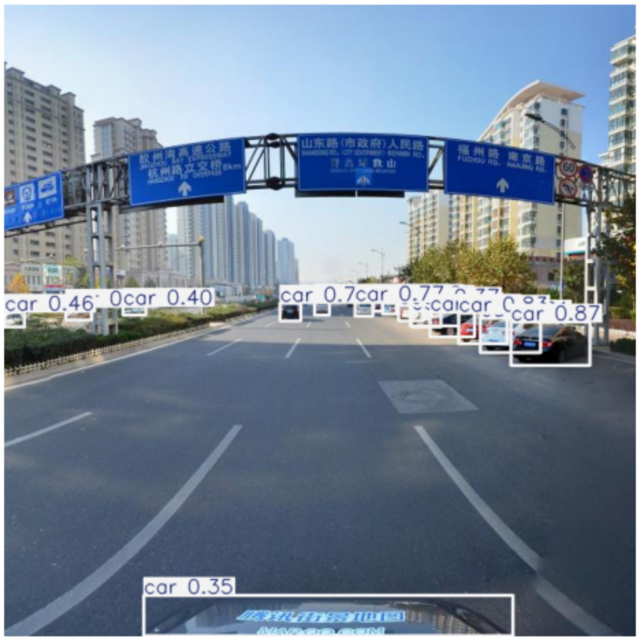
[7] B. Zhang, M. Li, and S. Li, "YOLO-TS: A Lightweight YOLO Model for Traffic Sign Detection," *Electronics*, vol. 12, no. 21, p. 4424, Oct. 2023.



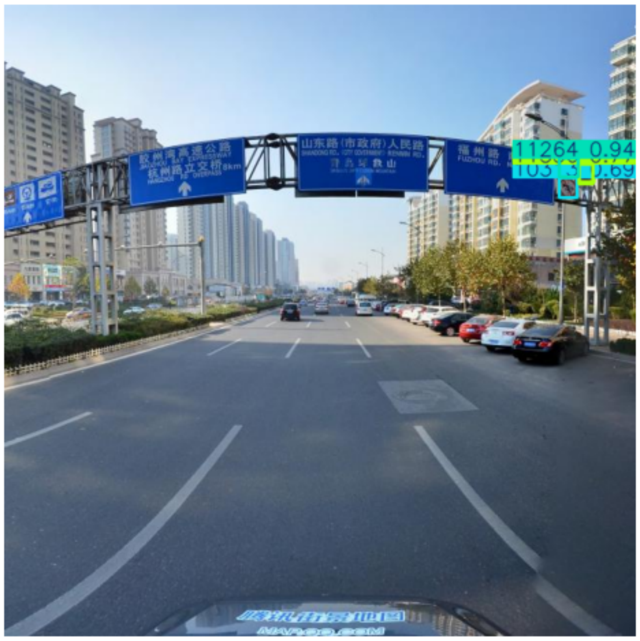
(a) Before Fine-Tuning (Example 1)



(b) After Fine-Tuning (Example 1)



(c) Before Fine-Tuning (Example 2)



(d) After Fine-Tuning (Example 2)

Fig. 4: Comparison of Bounding Box Predictions Before and After Fine-Tuning