Autonomous Driving System in CARLA Simulator Using Deep Learning-Based Perception

Justin Mascarenhas

Department of Computer Engineering
Rochester Institute of Technology, Rochester, NY

Abstract

This paper presents the design, implementation, and evaluation of an autonomous driving system developed within the CARLA 0.9.15 simulator. The system integrates a custom-trained YOLO11 model for real-time object detection and a fine-tuned Ultra Fast Lane Detection (UFLD) model for lane boundary identification. A novel lane-following control strategy with waypoint-based fallback is implemented, enabling robust navigation through diverse driving scenarios including intersections, traffic signals, and dynamic obstacles. The perception models were trained on custom datasets generated through automated data collection pipelines that leverage CARLA's semantic segmentation and 3D-to-2D projection capabilities. Experimental results demonstrate successful autonomous navigation in urban environments with 30+ vehicles and 10+ pedestrians, achieving reliable traffic light recognition and obstacle avoidance behavior.

Keywords—autonomous driving, CARLA simulator, YOLO, lane detection, deep learning, perception

I. INTRODUCTION

Autonomous driving represents one of the most challenging applications of robotics and computer vision, requiring seamless integration of perception, planning, and control systems. The ability to perceive the environment accurately, understand traffic rules, and make safe driving decisions in real-time poses significant technical challenges that have motivated extensive research in the field.

Simulation environments have become essential tools for developing and validating autonomous driving algorithms. The CARLA (Car Learning to Act) simulator provides a realistic testing platform that closely mimics real-world driving conditions, including accurate physics simulation, diverse weather conditions, and complex traffic scenarios. By developing and validating autonomous systems in simulation, researchers can iterate rapidly while ensuring safety during development.

This work presents a complete autonomous driving pipeline consisting of three primary modules: (1) a perception system utilizing YOLO11 for object detection and UFLD for lane detection, (2) a decision module for traffic light interpretation and steering source selection, and (3) a control module implementing lane-following with waypoint-based fallback. A key contribution is the development of automated data generation pipelines that leverage CARLA's ground truth capabilities to create high-quality training datasets.

II. METHODOLOGY

A. System Architecture

The autonomous driving system follows a modular architecture with clear separation between perception, decision-making, and control. An RGB camera mounted on the ego vehicle (1640×590 pixels, 150° FOV) captures frames that are processed in parallel by two neural networks. YOLO11 performs object detection while UFLD identifies lane boundaries. The outputs feed into a decision module that interprets traffic signals and selects appropriate steering strategies. Finally, the control module translates high-level decisions into vehicle actuator commands.

B. Data Generation Pipeline

A core contribution of this work is the automated data generation methodology that produces high-quality training datasets directly from the CARLA simulator. For object detection, the pipeline leverages CARLA's semantic segmentation camera and 3D actor information to automatically generate YOLO-format bounding box annotations.

YOLO Dataset Generation:

The data generator spawns an ego vehicle equipped with both RGB and semantic segmentation cameras at identical transforms. For each frame, the 3D bounding boxes of all actors (vehicles, pedestrians, traffic lights) are projected onto the

2D image plane using the camera intrinsic matrix. Occlusion filtering is performed using CARLA's raycast API—a ray is cast from the camera to each actor's center, and actors are only annotated if the ray reaches within 0.5m of the target without obstruction. This ensures that occluded objects do not generate erroneous training labels.

The projection matrix is computed from the camera field-of-view: focal length $f = w/(2 \cdot \tan(FOV \cdot \pi/360))$, with principal point at image center. The world-to-camera transformation converts 3D world coordinates to camera space, followed by perspective projection to obtain 2D pixel coordinates. Bounding boxes are clipped to image boundaries and filtered by minimum area (100 pixels) and maximum detection distance (60m for vehicles, 30m for pedestrians).

UFLD Dataset Generation:

For lane detection, the data generator utilizes CARLA's semantic segmentation to identify lane marking pixels (class ID 6). The segmentation mask is processed row-by-row at fixed y-positions (row anchors) spanning the bottom 60% of the image. Connected clusters of lane marking pixels in each row are identified, and cluster centers are computed as the lane x-position for that row. This produces the TuSimple-format labels required for UFLD training—each lane is represented as a list of x-coordinates at predefined y-positions, with -2 indicating no lane detection at that row.

C. YOLO11 Object Detection Model

The object detection module employs YOLO11n (nano variant), selected for its optimal balance between inference speed and accuracy. The model was trained on a custom CARLA dataset comprising approximately 5,000 training images collected across multiple maps with varying weather and lighting conditions.

Training Configuration:

- Architecture: YOLO11n with pretrained COCO weights
- Input resolution: 640×640 pixels
- Optimizer: AdamW with learning rate 0.005
- Epochs: 250 with patience 60 for early stopping
- Batch size: 128, Augmentation: enabled
- Classes: vehicle, pedestrian, traffic_light, speed_limit

The training process utilized the Ultralytics framework with cosine learning rate scheduling. Data augmentation including random flipping, scaling, and color jittering was applied to improve generalization. The model outputs bounding boxes with class labels and confidence scores, which are filtered at inference with a threshold of 0.3.

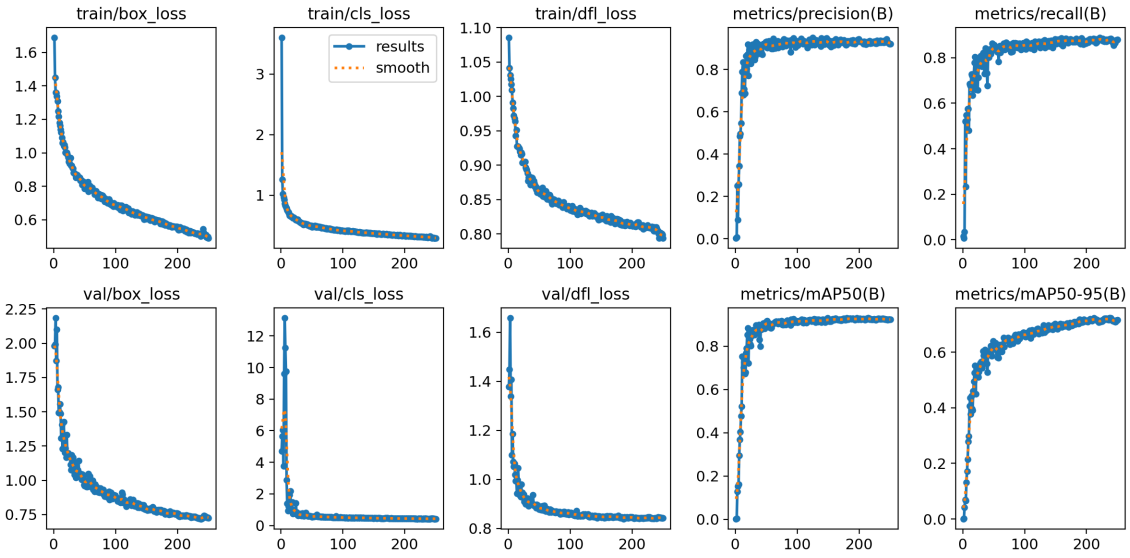

Fig. 1. YOLO11 training metrics showing loss curves (box, cls, dfl) and performance metrics (precision, recall, mAP50, mAP50-95) over 250 epochs. The model demonstrates stable convergence with validation mAP50 reaching approximately 0.85.

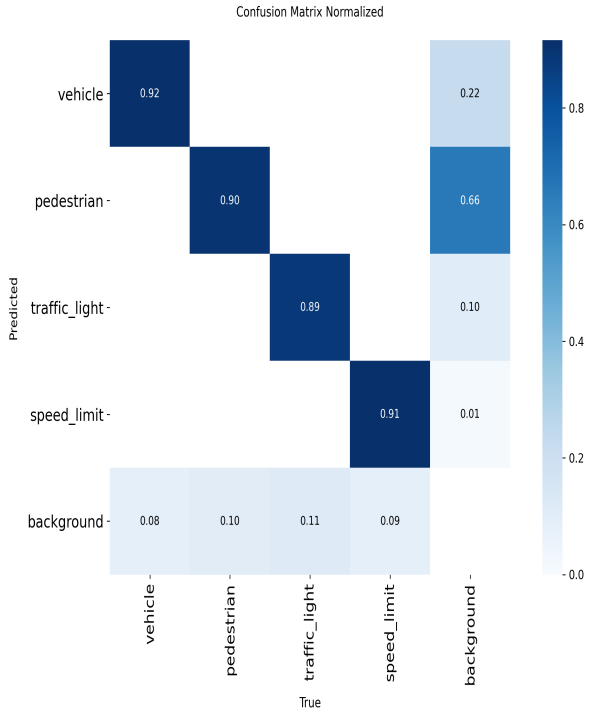

Fig. 2. Normalized confusion matrix for YOLO11 object detection. The model achieves high accuracy for vehicle and pedestrian classes, with some confusion between traffic_light and background classes due to the small size of traffic light bounding boxes.

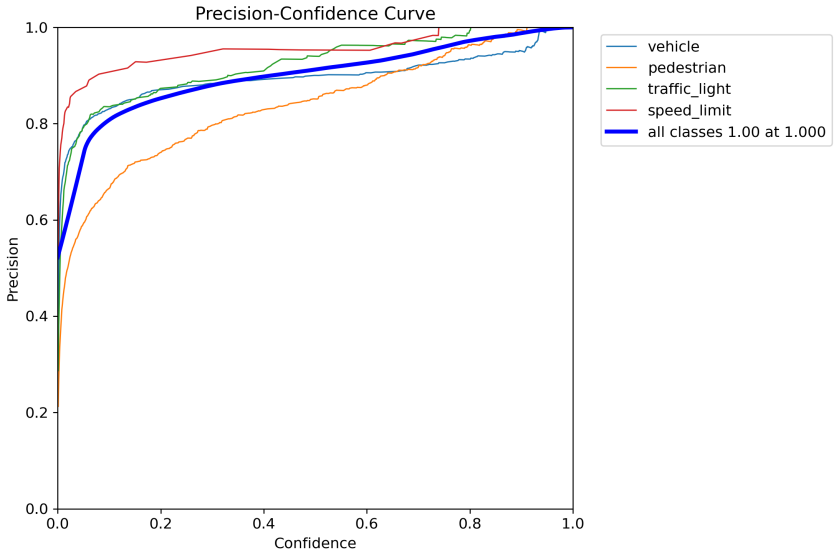

Fig. 3. Precision-Confidence curve showing the relationship between detection confidence threshold and precision for each class. Higher confidence thresholds yield higher precision at the cost of recall.

D. Ultra Fast Lane Detection Model

Lane detection utilizes the Ultra Fast Lane Detection (UFLD) architecture, which reformulates lane detection as a row-wise classification problem rather than pixel-wise segmentation. This approach enables real-time inference while maintaining high accuracy.

Model Architecture:

The model employs a ResNet-18 backbone pretrained on ImageNet, followed by classification heads that predict lane positions at 56 row anchors. The input image (800×288 pixels) is divided into 100 horizontal grid cells, and for each row

anchor, the model predicts which grid cell contains the lane (or outputs a "no lane" class). This grid-based formulation significantly reduces computational complexity compared to dense segmentation approaches.

Loss Functions:

Training employs three complementary loss functions: (1) SoftmaxFocalLoss for the primary row-anchor classification task, which focuses learning on hard examples by down-weighting easy classifications with γ=2; (2) ParsingRelationLoss for structural consistency, which penalizes large position differences between adjacent row anchors to encourage smooth lane curves; (3) ParsingRelationDis for shape consistency, which penalizes second-order differences (curvature changes) to maintain lane continuity. The total loss combines these with weights: $L_{total} = L_{cls} + 1.0 \cdot L_{sim} + 0.0 \cdot L_{shp}$.

Fine-tuning Strategy:

- Base model: TuSimple pretrained weights (tusimple_18.pth)
- Optimizer: Adam with learning rate 1×10e-4
- Epochs: 50 with cosine annealing scheduler
- Batch size: 32, Weight decay: 1×10e-4
- Gradient clipping: 1.0 for training stability

The model was fine-tuned on a custom CARLA dataset of approximately 3,000 frames. The fine-tuning approach preserves the pretrained feature representations while adapting the classification heads to CARLA's visual characteristics, which differ from real-world TuSimple highway imagery in terms of rendering style and lane marking appearance.

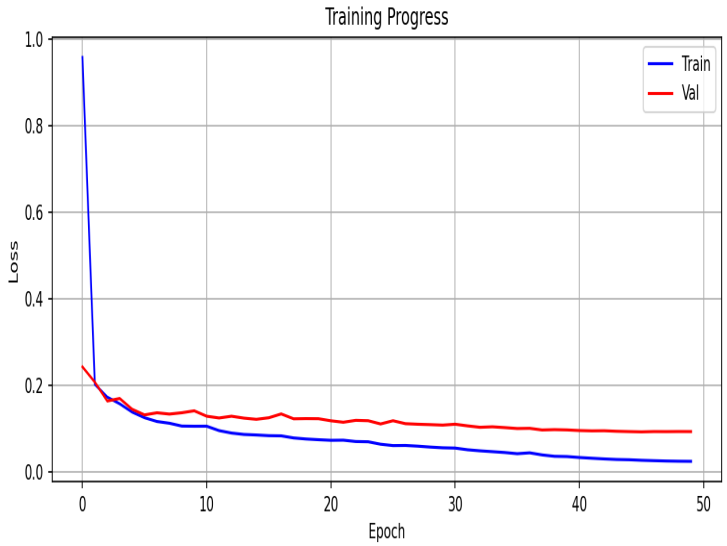

Fig. 4. UFLD training and validation loss curves over 50 epochs. The model shows rapid initial convergence followed by gradual refinement, with validation loss closely tracking training loss indicating good generalization without overfitting.

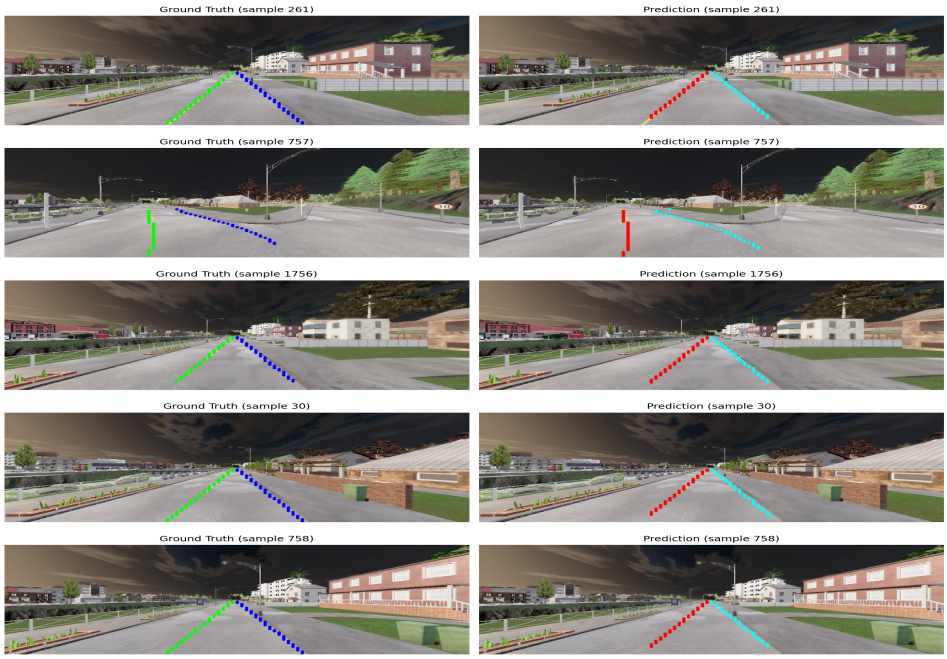

Fig. 5. Sample lane detection predictions on CARLA validation images. The model accurately identifies lane boundaries with the predicted lane points (colored lines) closely matching the actual lane markings.

E. Lane Following with Waypoint Fallback

A key contribution of this work is the adaptive steering control strategy that combines lane-following with waypoint-based navigation. Pure lane-following approaches suffer from instability at intersections and areas with absent or ambiguous lane markings, while pure waypoint following lacks the precision of lane-centering on well-marked roads.

The steering angle from lane detection is computed using both the lane trajectory angle and lateral offset: $\theta_{steer} = \theta_{lane} + (offset/center_x) \cdot 15°$, where θ_{lane} is the arctangent of the lane direction vector and offset is the horizontal distance from the lane center to the image center. This combined formulation provides both directional and positional feedback for smooth lane centering.

The adaptive selection mechanism operates as follows: when lane detection confidence exceeds 0.3 and the computed steering angle magnitude is within 5°, the system uses lane-based steering for precise centering. When the steering angle exceeds 5° (indicating turns or lane changes) or lane detection is unreliable, the system transitions to waypoint-based steering. Waypoints are generated from CARLA's map API with 2-meter spacing, and the steering angle to the next waypoint is computed using the dot product between the vehicle's forward vector and the waypoint direction.

F. Traffic Light Detection

Traffic light state recognition is performed using HSV color space analysis on YOLO-detected traffic light regions. The detected bounding box is cropped from the original image and converted to HSV color space. Pixel counts are computed for three color ranges tuned specifically for CARLA's rendering: red (H: 0-15, 160-180), yellow (H: 16-40), and green (H: 40-90), all with S,V thresholds of 70-255. The dominant color determines the traffic light state, triggering a full stop for red/yellow/unknown when the traffic light is within 20 meters.

III. EXPERIMENTAL RESULTS

The integrated system was evaluated in CARLA's Town02 urban environment with dynamic traffic comprising 30 spawned vehicles and 10 pedestrians. The simulation operated in synchronous mode with a fixed time step of 0.3 seconds to ensure deterministic behavior and prevent frame drops. Key system parameters are summarized in Table I. The maximum vehicle speed was capped at 20 km/h to facilitate safe navigation in dense traffic. The lane/waypoint steering threshold was set at 5° to balance precision and responsiveness. Safe distance thresholds for vehicles and pedestrians were set at 15 meters and 10 meters respectively, with a traffic light detection distance of 20 meters.

Table I: System Parameters

Parameter	Value
Maximum Speed	20 km/h
Maximum Steering Angle	±70°
Lane/Waypoint Threshold	5°
Vehicle Safe Distance	15 m
Pedestrian Safe Distance	10 m
Traffic Light Distance	20 m
Detection Confidence	0.3
Simulation Time Step	0.3 s

The system demonstrated the following capabilities: (1) stable lane-following on straight road segments with smooth centering behavior, (2) successful transition to waypoint navigation at intersections and turns, (3) correct traffic light recognition with appropriate stop/go behavior, (4) emergency braking for pedestrians within the 10-meter threshold, and (5) speed modulation based on detected vehicle proximity. The vehicle successfully completed multiple laps of the test route without collisions or traffic violations.

IV. CHALLENGES AND SOLUTIONS

Traffic Light Color Misdetection: CARLA's rendered red lights appeared orange in HSV space. Solution: Extended red hue range to 0-15 and lowered S/V thresholds to 70.

Coordinate Mismatch: UFLD outputs at 800×288 while camera is 1640×590. Solution: Scale all coordinates by 2.05× in both dimensions.

Ego Vehicle Detection: The Tesla Cybertruck hood was detected as a vehicle. Solution: Changed to Tesla Model 3 with lower hood profile.

Lane Flickering: Multiple spurious lanes at intersections. Solution: Lane filtering with angle constraints (40° max) and width bounds (200-1600px).

Steering Oscillation: Pure lane-following caused zigzag motion. Solution: Threshold-based source selection (≤5° lane, >5° waypoint).

V. CONCLUSION

This work presented a complete autonomous driving system for the CARLA simulator, integrating custom-trained perception models with an adaptive control strategy. The key contributions include: (1) automated data generation pipelines leveraging CARLA's semantic segmentation for YOLO and UFLD training data, (2) a lane-following control strategy with waypoint fallback that eliminates steering oscillation while maintaining precision, and (3) comprehensive traffic light and obstacle handling.

Future work will explore LiDAR integration for 3D perception, sensor fusion approaches, and advanced planning algorithms such as A* or RRT for complex route planning. The modular architecture facilitates these extensions while maintaining the validated perception and control components.

REFERENCES

[1] Z. Qin, H. Wang, and X. Li, "Ultra Fast Structure-aware Deep Lane Detection," in *Proc. European Conference on Computer Vision (ECCV)*, 2020. [Online]. Available: https://github.com/cfzd/Ultra-Fast-Lane-Detection

[2] Ultralytics, "YOLO11: State-of-the-art Real-Time Object Detection," 2024. [Online]. Available: https://docs.ultralytics.com/models/yolo11/

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proc. Conference on Robot Learning (CoRL)*, 2017.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.