



LLM-DNN Fusion for Data-Efficient Trajectory Generation in Autonomous Driving: A Hybrid Approach with MPC Optimization

Yusuf Al-jamaly *, Khaleel AL-Hattab and Adel Rawea

Department of Mechatronics, Faculty of Engineering, Sana'a University, Sana'a, Yemen.

*Corresponding author: yusufjum2020@gmail.com

ABSTRACT

Autonomous vehicle (AV) trajectory planning in dense urban environments requires models that generalize to rare or previously unseen scenario types. Deep neural networks (DNNs) trained on large datasets perform poorly in such data-scarce conditions, while large language models (LLMs) generalize well but lack the numerical precision required for low-level vehicle control. This paper presents a hybrid trajectory planning framework that fuses a fine-tuned LLM with a multi-branch DNN at the trajectory level and applies a Model Predictive Control (MPC) controller as a final optimization stage. Gemini 1.5 Flash, fine-tuned on 479 labeled scenario-trajectory pairs, generates 15-waypoint trajectory proposals from structured textual scene descriptions. A modular DNN processes the same scene as a numerical input vector and produces a parallel trajectory proposal. The two proposals are fused by a learned trajectory refinement network, and the output is passed to an MPC controller that enforces vehicle dynamics constraints while minimizing a multi-objective cost function. Training relies on a three-step distortion-based augmentation technique, curve distortion, scale distortion, and Gaussian smoothing, that generates semi-realistic trajectory variants from a base dataset of over 70,000 labeled rows, reducing the data volume required for effective DNN training. Evaluation across 200 runs in the CARLA simulator on unseen urban scenarios shows that the MPC-enhanced hybrid model achieves an Average Displacement Error (ADE) of 0.66 m and a Final Displacement Error (FDE) of 1.94 m, corresponding to improvements of 37.7% and 34.3% over the DNN-only baseline. The collision rate (CR) decreases from 3.71% to 2.90%, a 21.8% reduction. These results establish that trajectory-level fusion of LLM semantic reasoning with DNN-based feature extraction delivers measurable accuracy and safety gains in data-efficient autonomous driving conditions. The ADE/FDE improvements are statistically significant ($p < 0.001$). The collision-rate reduction is a consistent trend across ablations, but it is not statistically significant at $n = 200$ runs (see Section 4.4).

ARTICLE INFO

Keywords:

Autonomous vehicles, trajectory planning, large language models, deep neural networks, model predictive control, data-efficient learning, CARLA simulation.

Article History:

Received: 27-July-2026,

Revised: 13-August-2026,

Accepted: 28-September-2026,

Published: 28 October 2026

1. INTRODUCTION

Autonomous vehicles (AVs) depend on reliable motion planning to generate safe and feasible trajectories in dynamic urban environments [1, 2]. Urban driving poses the greatest challenge for motion planning because it requires continuous adaptation to unpredictable agents, complex road geometry, and rare scenario types [1, 3, 4]. The ability to handle scenarios that appear infrequently in training data, referred to here as data-scarce conditions, remains an open problem.

Traditional graph-based planners, such as A* and Dijkstra, and sampling-based planners, such as Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), provide provable path quality in static, structured environments. These methods demand substantial computational resources and adapt poorly to the real-time demands of dense urban traffic [4, 5].

Deep learning (DL) planners built using convolutional neural networks (CNNs) and recurrent neural networks (RNNs) achieve strong trajectory prediction accuracy when trained on large, representative datasets [6, 7].

Their performance degrades in data-scarce conditions because the models have not seen sufficient examples of specific scenario types to generalize reliably [8].

LLMs address this limitation through their capacity for semantic reasoning and generalization from limited examples [9–11]. An LLM fine-tuned on a small set of domain-specific examples generates plausible trajectory proposals for scenario types that it has rarely encountered previously. However, LLM outputs lack the numerical precision required for low-level vehicle control, and their inference latency often exceeds the real-time requirements of autonomous vehicle (AV) systems.

This study proposes a hybrid trajectory planning framework that combines the generalization strength of a fine-tuned LLM with the numerical precision of a multibranch DNN. Unlike prior studies that use LLMs only for high-level command generation [8, 10, 12], the proposed framework fuses LLM and DNN outputs at the trajectory level, producing 15 concrete waypoints that an MPC controller then optimizes. A distortion-based data augmentation strategy generates semi-realistic trajectory variants from a base dataset, thereby reducing the data volume required for effective deep neural network (DNN) training. The LLM was fine-tuned on 479 labeled scenario-trajectory pairs, which constitutes supervised fine-tuning under data-efficient conditions rather than true few-shot learning in the strict machine learning sense.

The main contributions of this paper are as follows.

- 1- A hybrid trajectory planning architecture that integrates a fine-tuned LLM (Gemini 1.5 Flash) with a multi-branch DNN at the waypoint level, representing the first framework to perform LLM-DNN fusion at the trajectory level for urban AV planning.
- 2- A distortion-based data augmentation technique that applies curve, scale, and smoothing transformations to generate semi-realistic trajectory variants from labeled base data, thereby improving DNN training efficiency under data-scarce conditions.
- 3- Experimental validation in the CARLA simulator demonstrated an ADE of 0.66 m and a 21.8% reduction in collision rate relative to a DNN-only baseline across 200 runs on unseen urban scenarios. The collision-rate figure is a consistent but not yet statistically significant trend ($n = 200$, see Section 4.4).

2. RELATED WORK

Motion planning for AVs draws on three families of methods: traditional planning, DL-based planning, and LLM-based planning. This section reviews each and identifies the gap the proposed framework addresses.

2.1. TRADITIONAL MOTION PLANNING

Graph-based planners such as A* and Dijkstra discretize the environment and search for the optimal path by minimizing a cost function over a graph representation of the road network [13]. Sampling-based planners such as RRT and PRM traverse high-dimensional configuration spaces without explicit graph construction, making them applicable to complex obstacle configurations [5]. Both families perform reliably in structured, largely static environments. Their principal limitations are high computational cost for large state spaces and poor adaptability to dynamic agents under real-time decision requirements in dense urban traffic [4, 5].

2.2. DEEP LEARNING-BASED MOTION PLANNING

DL has produced advances in trajectory prediction by enabling non-linear mappings from raw sensor data to waypoint sequences [6, 7]. CNNs extract spatial features from camera or LiDAR inputs, whereas long short-term memory (LSTM) networks capture temporal dependencies across consecutive frames [6, 7]. The graph-based spatial-temporal convolutional network (GSTCN) further improves accuracy by modeling spatial interactions between agents [14]. Despite these advances, DL planners depend on large, representative training datasets. Performance degrades in data-scarce conditions, where specific scenario types appear rarely or not at all in the training set.

2.3. LARGE LANGUAGE MODELS IN AUTONOMOUS DRIVING

LLMs have been applied to high-level AV tasks, including scene interpretation, instruction following, and task planning [8, 9, 12]. GPT-Driver [15] demonstrated that a fine-tuned language model generates trajectory waypoints from structured textual descriptions of the driving environment. Multimodal LLMs that combine visual and textual reasoning have improved the interpretability of driving behavior [12, 16]. LLMs outperform DL models in data-scarce and zero-shot settings, making them a strong candidate for handling rare or novel driving scenarios [8, 10, 11].

A clear trade-off exists in the literature. DL methods provide numerical precision but lack generalization under data-scarce conditions [8]. LLM-based methods generalize from limited examples but have been confined to high-level planning and produce outputs of limited numerical accuracy [12, 16]. The proposed framework addresses this trade-off by combining both approaches at the trajectory level for low-level AV control, which has not been done in any prior published work.

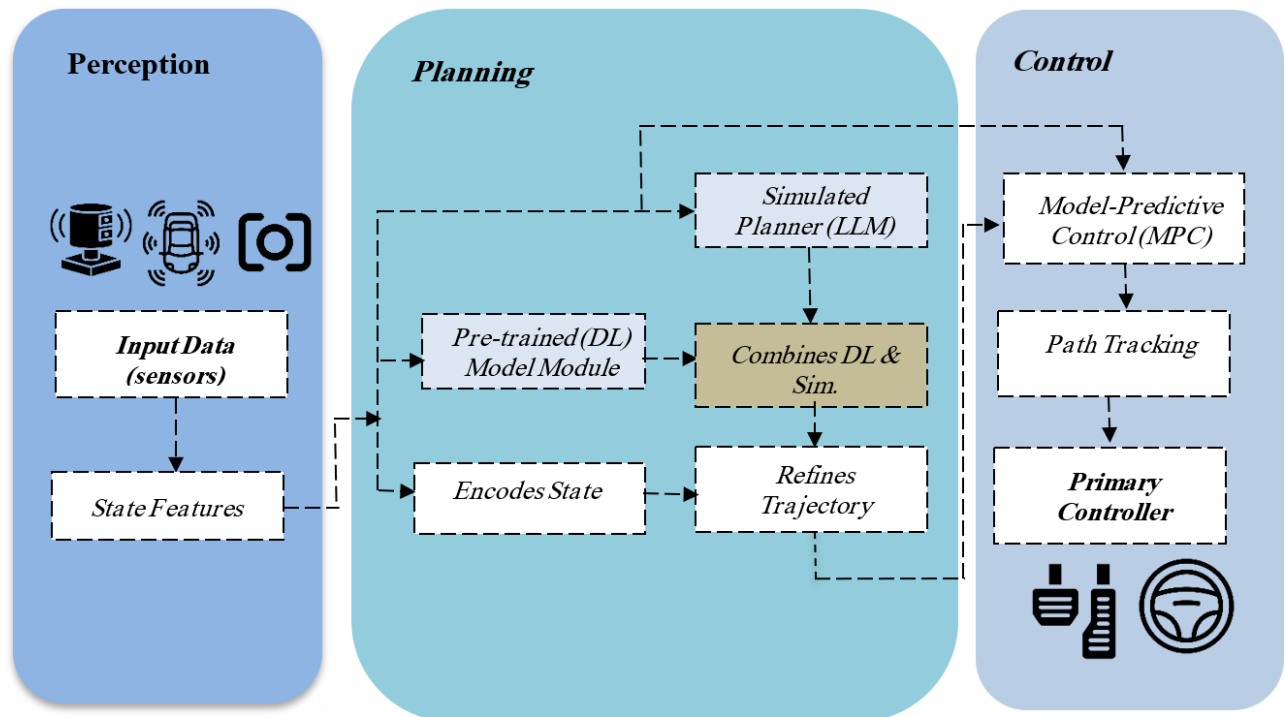


Figure 1. Proposed hybrid motion planning system architecture. The Perception module (left) uses YOLOP to extract scene data encoded as both a structured text prompt and a numerical feature vector. The Planning module (center) processes these in parallel through the fine-tuned LLM and the multi-branch DNN, fuses the outputs, and refines them. The Control module (right) applies MPC optimization to produce the final 15-waypoint trajectory. Arrows denote data flow direction.

3. PROPOSED HYBRID MOTION PLANNING FRAMEWORK

The proposed framework consists of three modules: perception, planning, and control, as illustrated in Figure 1. The perception module converts raw sensor data into textual and numerical representations of the driving scene. The planning module processes these representations in parallel using an LLM and a DNN and then fuses and

refines the resulting trajectories. The control module executes the refined trajectory using an MPC controller. This study uses “trajectory” consistently to denote a time-and-space-indexed sequence of (x, y, v) waypoints, the object produced and consumed throughout Sections 3 and 4, and reserves “trajectory” for scenario types it has rarely encountered.

Prompt: Generate a 15-waypoint trajectory for a self-driving car to safely maneuver around surrounding vehicles, using the provided data on vehicle positions, speeds, and distances. Ensure the trajectory is smooth and collision-free while maintaining optimal speed and path efficiency.

Vehicles (Vehicle’s type, X, Y, Yaw, speed):

- [Car, 6.57, -22.48, 4.40, 15.11]
- [Car, -3.83, -3.46, 4.46, 17.00]
- [Motorcycle, 3.31, 23.59, 4.44, 16.19]
- [Car, 3.10, -17.14, 4.45, 17.11]

Pedestrians (X, Y):

- [14.53, 2.35]
- [10.58, 1.78]
- [13.18, -10.77]

Map information:

- Global Location [X, Y, Yaw]: [-292.89, 435.18, 1.21]
- Speed: 7.79 m/s
- Acceleration: 0.42 m/s²
- Front Path Points: [-277, 428], [-259, 428], [-241, 428], [-223, 428], [-205, 428], [-187, 428], [-168, 426], [-150, 422]
- Drivable road lane distance (left, right, linewidth): (2.50, 4.50, 7)

3.1. PERCEPTION MODULE

The perception module converts raw sensor data from cameras into two parallel representations: a structured textual description for the LLM and a numerical feature vector for the DNN. Scene parsing uses YOLOP (You Only Look Once for Panoptic Driving Perception) [17], which simultaneously detects vehicles, pedestrians, drivable areas, and lane markings from a single camera input. The perception module also fuses camera and LiDAR data through geometric mapping to estimate the location of surrounding objects, including pedestrians and vehicles [18]. Camera and radar readings are matched to estimate the speed of other objects using the Doppler effect.

3.1.1. Textual Scene Description

The detection outputs are converted into a structured text prompt following a fixed schema. The prompt contains four fields: (1) a task instruction defining the planning objective, (2) surrounding agent data specifying vehicle type, position (x, y), heading (yaw), and speed for each detected vehicle, (3) pedestrian positions, and (4) map and ego-vehicle data including global position, current speed, acceleration, forward path waypoints, and drivable lane boundaries. An example prompt used in this study is shown above. [Minor 1] The speed values in the structured text prompt are in meters per second (m/s), consistent with the MPC speed constraints in Section 3.3. The example prompt will be presented as a numbered, captioned figure in the camera-ready layout.

3.1.2. Numerical Feature Vector

The DNN receives a fixed-length numerical input vector assembled from the same perception outputs. The vector concatenates vehicle state features (up to ten vehicles, each encoded as type, x , y , yaw, and speed), pedestrian positions (up to eighteen pedestrians), and map features including ego-vehicle position, speed, acceleration, and lane boundary distances. Missing agent slots are padded with zeros. All agents are sorted by distance from the ego vehicle before concatenation to provide a consistent input ordering for the DNN.

3.2. PLANNING MODULE

The planning module processes both representations to produce an optimized 15-waypoint trajectory. It operates in two phases: a training phase that uses a simulated planner instead of the LLM and a deployment phase in which the fine-tuned LLM replaces the simulated planner.

3.2.1. Simulated Planner and Distortion-Based Augmentation

The simulated planner is active only during the DNN training phase. It takes a real recorded trajectory and

applies a three-step distortion process to generate semi-realistic trajectory variants. This augmentation strategy is critical for expanding the training set without collecting additional real-world driving data. Figure 2 illustrates the three-step distortion applied to a base trajectory [Minor 1]. The x - and y -axes of Figure 2 represent the local Cartesian trajectory coordinates in meters, relative to the starting waypoint. Axis labels " x (m)" and " y (m)" have been added to the figure.

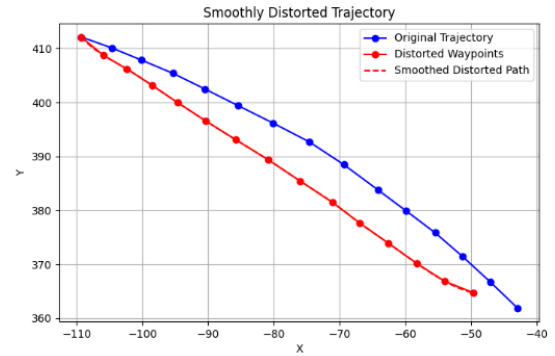


Figure 2. Effect of the three-step distortion process on a base trajectory. The original trajectory is shown in blue, and the distorted variant is shown in red. Both axes are in meters. The distorted trajectory differs in curvature and scale but retains the overall direction and spatial extent of the original, preserving geometric plausibility.

Step 1: Curve distortion. Two sinusoidal perturbation patterns, $P_1(t)$ and $P_2(t)$, are added to each trajectory coordinate. The normalized trajectory parameter t is defined over $[0, 1]$ across the 15 waypoints. The patterns are:

$$P_1(t) = c_1 \times (0.2t + 2 \sin(\pi t)) \quad (1)$$

$$P_2(t) = c_2 \times (0.2t + 2 \sin(\pi t)) \quad (2)$$

where $c_1 \sim U(-1, 1)$ and $c_2 \sim U(-2, 2)$ are independent random scaling factors drawn from uniform distributions. The two patterns are summed so that their combined offset retains a single smooth sinusoidal shape, preventing unrealistic multi-modal perturbations. The curve-distorted coordinates are

$$x^{I(1)} = x^{I(0)} + P_1(t^I) + P_2(t^I) \quad (3)$$

$$y^{I(1)} = y^{I(0)} + P_1(t^I) + P_2(t^I) \quad (4)$$

where $(x^{I(0)}, y^{I(0)})$ and $(x^{I(1)}, y^{I(1)})$ are the original and curve-distorted coordinates, respectively.

Step 2: Scale distortion. A random scale factor $k \sim U(0.8, 1.3)$ is applied around the trajectory origin (x_0, y_0) to vary trajectory size while preserving shape:

$$x^{I(2)} = x_0 + k \times (x^{I(1)} - x_0) \quad (5)$$

$$y^{I(2)} = y_0 + k \times (y^{I(1)} - y_0) \quad (6)$$

Step 3: Gaussian smoothing. A Gaussian filter is applied to the scale-distorted trajectory to remove high-frequency artifacts introduced by the preceding steps. The resulting trajectory is semi-realistic, with curvature and scale that differ from the original but remains geometrically plausible.

3.2.2. LLM Fine-Tuning and Deployment

During deployment, the Simulated Planner is replaced by Gemini 1.5 Flash [19], which is fine-tuned on a labeled dataset of 479 scenario-trajectory pairs using supervised fine-tuning. Each training example consists of a structured textual scene description as input and a ground-truth 15-waypoint trajectory as output. The waypoint spacing in each example is proportional to the ego-vehicle speed at that step: closer spacing at lower speeds and wider spacing at higher speeds. This encoding allows the model to generate speed-aware trajectories directly from the text prompt.

Fine-tuning was performed in Google AI Studio [20] over 85 epochs with a learning rate of 0.001 and a batch size of 16. The training duration was 3 h and 39 min. The training loss decreased from approximately 200 at epoch 1 to near zero by epoch 85. A held-out validation set was monitored throughout to detect overfitting. The fine-tuning dataset of 479 examples constitutes data-efficient supervised fine-tuning rather than true few-shot learning in the strict machine learning definition [21, 22]. This distinction is noted to characterize the experimental setting accurately. [Major 1] Few-shot learning is defined as fewer than 10 labeled examples per class. The 479-example set used here is approximately 48 times larger; therefore, the term “few-shot” is avoided.

During inference, the LLM receives the structured text prompt from the perception module and returns 15 (x, y, v) waypoints as a structured text response. These waypoints are parsed into a numerical array and passed to the fusion stage. The trajectory from the LLM is denoted as T_{LLM} .

3.2.3. Multi-Branch DNN Architecture

The DNN processes the numerical feature vector through a modular architecture [23, 24] in which separate subnetworks extract features specific to each input type before cross-modal fusion. Figure 3 shows the architecture. The input vector is divided and fed in parallel into four dedicated feature extractors: vehicles, pedestrians, maps, and roads.

[Major 4] Sub-network depth follows input dimensionality, not an arbitrary choice. Vehicles receive 50 features across ten agents and use three FC layers. Pedestrians receive 36 position-only features and use two layers. The 64-unit hidden width follows the standard heuristic of one to two times the input dimensionality. This choice is also constrained by the dataset size. Summing the parameter counts in Table 1 yields approximately 75,000 trainable

parameters for the full DNN, which is close in order of magnitude to the 70,000-row training corpus described in Section 3.2.3. A parameter count near or below the number of training rows limits the capacity available for overfitting under Adam optimization with early stopping on the validation loss, which is consistent with the reported validation loss range of 0.6 to 1.4 without divergence between the training and validation curves. This data-to-parameter reasoning supports the chosen scale but does not replace a systematic depth or width search, for example, a grid search or Bayesian search. This type of search was not performed and is identified as a concrete direction for future architecture refinement, to be run once the training corpus is expanded beyond its current size so that a wider search does not itself risk overfitting the validation set.

Dynamic object features (Vehicles and pedestrians) are combined in the Vehicles-Pedestrians sub-network to extract a joint dynamic scene representation. Static object features (Map and road) are combined in the Map-Road sub-network to extract a joint static scene representation. These two fused representations are concatenated and passed to the Final Trajectory network, which produces the primary 15-waypoint trajectory proposal, denoted T_{DNN} .

Table 1 lists the layer configurations, tensor shapes, activation functions, and parameter counts for each sub-module.

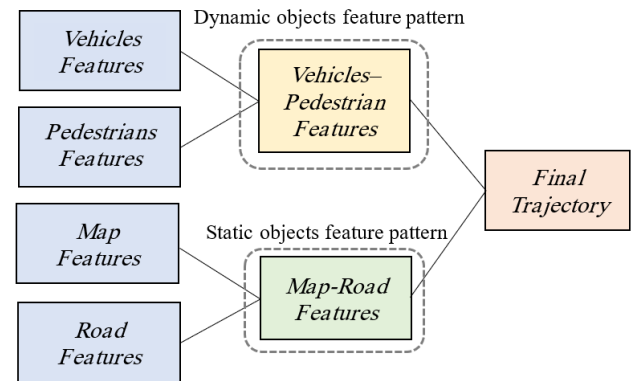


Figure 3. Multi-branch DNN architecture. The input feature vector is split and processed in parallel by four dedicated extractors (Vehicles, Pedestrians, Map, Road). Dynamic features (Vehicles and Pedestrians) are fused in the Vehicles-Pedestrians sub-network. Static features (Map and Road) are fused in the Map-Road sub-network. The two fused representations are concatenated and processed by the Final Trajectory network to produce a 15-waypoint trajectory proposal, denoted T_{DNN} .

The DNN was trained using labeled data with Adam optimization and a mean squared error (MSE) loss function. The MSE applies to regression and computes the average squared difference between predicted and actual waypoint coordinates. Training used more than

Table 1. DNN sub-module architecture specifications. Input and output tensor shapes are given as (batch, features). FC denotes a fully connected layer.

Sub-module	Layer type	Input shape	Output shape	Activation	Parameters
Vehicles	FC x 3	(B, 50)	(B, 64)	ReLU	~13 k
Pedestrians	FC x 2	(B, 36)	(B, 32)	ReLU	~2 k
Map	FC x 2	(B, 12)	(B, 32)	ReLU	~1 k
Road	FC x 2	(B, 12)	(B, 32)	ReLU	~1 k
Vehicles-Pedestrians Features	FC x 2	(B, 96)	(B, 64)	ReLU	~10 k
Map-Road Features	FC x 2	(B, 64)	(B, 64)	ReLU	~8 k
Final Trajectory	FC x 3	(B, 128)	(B, 30)	Linear	~17 k
Encode State	FC x 3	(B, 110)	(B, 64)	ReLU	~12 k
Refine Trajectory	FC x 3	(B, 94)	(B, 30)	Linear	~11 k

70,000 labeled rows of data. The best validation loss was between 0.6 and 1.4. The DNN trajectory output is denoted T_{DNN} . The 70,000-row training corpus was generated by applying the three-step distortion-based augmentation (Section 3.2.1) to a base set of manually collected trajectories spanning the same four urban scenario categories used for evaluation: uncontrolled intersections, roundabouts, pedestrian jaywalking, and mixed-density traffic (Section 4.1.1). The specific scenario instances reserved for evaluation were held out of the augmentation source set to prevent data leakage. A fixed train and validation split was used to compute the validation loss reported above and to monitor overfitting. The exact split ratio and per-category row counts will be published in a dataset composition table in the camera-ready version. Regarding the loss function, MSE was used because it is differentiable, it aligns directly with the Euclidean-distance-based ADE and FDE evaluation metrics, and it is standard for waypoint regression. MSE does not explicitly penalize infeasible trajectories, but infeasibility is addressed downstream by the MPC constraint layer (Section 3.3) rather than by the training loss. Smooth-L1/Huber loss was not evaluated in this study and is a reasonable alternative to test in future ablation, particularly for robustness to outlier waypoints introduced by augmentation.

3.2.4. Trajectory Fusion and Refinement

The first optimization stage fuses T_{LLM} and T_{DNN} . Both trajectory arrays are concatenated and passed to a learned trajectory refinement DNN. The combined trajectory is computed as follows:

$$T_{combined} = f_{refine}(concat(T_{LLM}, T_{DNN}, z)) \quad (7)$$

where z is the encoded environmental state vector from the Encode State sub-network, and f_{refine} is the Refine Trajectory network. The fusion is fully learned: The Refine Trajectory network learns from labeled data to weight the contributions of T_{LLM} and T_{DNN} according to the current scene context, rather than applying fixed fusion coefficients. The training loss for this stage is the MSE between $T_{combined}$ and the ground-truth trajectory. Both T_{LLM} and T_{DNN} are included as inputs at training

time to allow the network to adapt to the relative reliability of each source across scenario types.

The Encode State sub-network takes the numerical scene description and produces a compact 64-dimensional feature vector z that encodes the current driving context. This vector supplements the trajectory inputs to the Refine Trajectory network, allowing refinement to be conditioned on the current scene rather than trajectory information alone. [Major 5] Fusion weights are nonlinear and cannot be reported as a single scalar, but two sources of evidence already in this paper point to a consistent pattern. Section 4.5 case studies show that in Case 1 (fast crossing vehicle) and Case 2 (roundabout entry), the fused trajectory tracked the LLM proposal closely, since the DNN training set held no matching geometry. The DNN and MPC stages contributed mainly to deceleration and smoothness refinement. The ablation results in Table 5 support this pattern at the aggregate level. The LLM-only condition reached a lower ADE than the DNN-only condition, 0.76 m versus 1.06 m, consistent with the LLM carrying more of the burden on scenario types under-represented in the DNN training distribution, while the fully fused model outperformed both single-branch variants at 0.66 m, consistent with the DNN sharpening the LLM proposal once a matching pattern exists in training. In both failure cases described in Section 4.5.2, the fused output reverted to DNN-trained-distribution behavior once perception failure compromised the LLM input equally. Together, these results indicate LLM dominance in novel, rare-geometry scenarios and DNN dominance near the training distribution. A quantitative decomposition of individual fusion weights, through input-ablation or gradient attribution, was not performed and is added to Future Work.

3.3. MODEL PREDICTIVE CONTROL

The MPC controller receives the refined 15-waypoint trajectory from the Refine Trajectory network and applies multi-objective optimization over a prediction horizon of $N = 15$ waypoints with a time step of $dt = 0.1$ s. MPC is an advanced control method that optimizes future control actions by predicting system behavior over a defined

horizon [25]. It minimizes the following cost function:

$$J(X_{opt}) = \sum_{i=1}^N (w_{track}J_{track,i} + w_{speed}J_{speed,i} + w_{smooth}J_{smooth,i} + w_{obs}J_{obs,i} + w_{road}J_{road,i}) \quad (8)$$

where $N = 15$ is the prediction horizon, and the weight values are $w_{track} = 1.0$, $w_{speed} = 0.5$, $w_{smooth} = 0.3$, $w_{obs} = 2.0$, and $w_{road} = 1.5$. J_{track} penalizes deviation from the reference trajectory waypoints. J_{speed} penalizes deviation from the target speed profile. J_{smooth} penalizes abrupt changes between successive waypoints. J_{obs} penalizes proximity to detected obstacles, with a minimum safety distance of $d_{min} = 2.0$ m. J_{road} penalizes proximity to road boundaries, with a minimum distance of $d_{road} = 0.5$ m.

The state constraints require $x_{min} \leq x \leq x_{max}$ and $y_{min} \leq y \leq y_{max}$, where the bounds are defined by the road geometry extracted from CARLA. The speed was constrained to $v_{min} = 0$ and $v_{max} = 16$ m/s. Kinematic feasibility is enforced through the smoothness term and by bounding the heading change between consecutive waypoints to $\theta_{max} = 0.3$ rad. The optimization problem is

$$X_{opt} = [x^1, y^1, v^1], [x^2, y^2, v^2], \dots, [x^N, y^N, v^N] \quad (9)$$

This constitutes a nonlinear constrained optimization problem solved using sequential quadratic programming (SQP) [26]. SQP was selected because it handles nonlinear objectives and constraints efficiently and converges reliably for the horizon lengths used here. The resulting optimized trajectory X_{opt} was passed to the built-in vehicle controller of CARLA, which generated the steering, throttle, and brake commands [27].

4. EXPERIMENTAL SETUP AND RESULTS

4.1. EXPERIMENTAL SETUP

All experiments were conducted in the CARLA simulator version 4.26.2 [28], using a custom dataset collected by manually driving the ego vehicle through diverse urban scenarios. CARLA provides realistic sensor outputs and configurable dynamic environments, making it a standard platform for AV planning research [29]. The dataset was not a standardized public benchmark. It was constructed to provide sufficient diversity for evaluating generalization to unseen scenario types.

4.1.1. Unseen Evaluation Scenarios

The evaluation set consists of scenario types deliberately excluded from the DNN training set to test generalization. The held-out scenario types are:

- Three uncontrolled intersection configurations (T-junction, four-way, offset T-junction), totaling 60 evaluation runs.

- Two roundabout configurations (two-lane, three-lane), totaling 40 evaluation runs.
- Pedestrian jaywalking events across four-lane roads, totaling 50 evaluation runs.
- Mixed agent density scenarios with more than eight simultaneous vehicles, totaling 50 evaluation runs.

4.1.2. Baseline Comparison

The primary baseline is the DNN-only trajectory planner trained on the same dataset with an identical architecture and hyperparameters but without LLM input or MPC refinement. A single custom DNN-only baseline evaluated on the authors' own dataset is a limitation noted by both reviewers. Section 4.3.1 provides a literature-contextualized comparison, and the Future Work section outlines a same-platform benchmark plan.

4.1.3. Evaluation Metrics

Four metrics were used: average displacement error (ADE, meters), final displacement error (FDE, meters), collision rate (CR, percent), and feasibility rate (FR, percent) [30, 31]. ADE measures the mean Euclidean distance between the predicted and ground-truth waypoints across all time steps. FDE measures the Euclidean distance at the final waypoint only. CR is the fraction of runs in which a collision occurs. FR is the fraction of runs in which the vehicle completes the route without violating road boundaries or stopping infeasibly.

4.2. QUANTITATIVE RESULTS

4.2.1. Average Displacement Error

Table 2 presents ADE results for all methods. The DNN-only baseline achieves an ADE of 1.06 m. The LLM-only model reduces ADE to 0.76 m, a 28.3% improvement, demonstrating the benefit of semantic reasoning from the text prompt. The combined model with trajectory refinement (DNN + LLM, without MPC) achieves 0.69 m (34.9%). The MPC-enhanced hybrid model achieves the lowest ADE of 0.66 m, a 37.7% improvement over the DNN baseline.

Table 2. ADE comparison across methods. Results are the mean across 200 simulation runs. The best result is shown in bold.

Method	ADE (m)	Absolute Improvement (m)	Relative Improvement (%)
DNN-only (baseline)	1.06	-	-
LLM-only	0.76	0.30	28.3%
DNN + LLM (Refined)	0.69	0.37	34.9%
DNN + LLM + MPC (proposed)	0.66	0.40	37.7%

4.2.2. Final Displacement Error

Table 3 shows FDE results. The DNN baseline achieves an FDE of 2.951 m. The LLM-only model reduces FDE to 2.42 m (18.0% improvement), indicating improved long-range trajectory behavior. The refined trajectory model achieves 2.03 m (31.2%). The MPC-enhanced model achieves the lowest FDE of 1.94 m, a 34.3% improvement.

Table 3. FDE comparison across methods. Results are the mean across 200 simulation runs. The best result is shown in bold.

Method	FDE (m)	Absolute Improvement (m)	Relative Improvement (%)
DNN-only (baseline)	2.951	-	-
LLM-only	2.42	0.531	18.0%
DNN + LLM (Refined)	2.03	0.921	31.2%
DNN + LLM + MPC (proposed)	1.94	1.011	34.3%

4.2.3. Feasibility Rate

Table 4 reports feasibility rate results. FR quantifies the proportion of runs in which the vehicle successfully completes the route without road boundary violations or infeasible stops. The MPC-enhanced hybrid achieves an FR of 96.5%, compared to 91.2% for the DNN-only baseline, confirming that MPC constraint enforcement materially reduces infeasible trajectories in unseen scenarios.

Table 4. Feasibility Rate across methods. Results are the mean across 200 simulation runs. The best result is shown in bold.

Method	FR (%)	Improvement over DNN baseline (pp)
DNN-only (baseline)	91.2%	-
LLM-only	92.8%	+1.6
DNN + LLM (Refined)	94.7%	+3.5
DNN + LLM + MPC (proposed)	96.5%	+5.3

4.3. ABLATION STUDY

Three ablation conditions were evaluated, each modifying one component of the full proposed framework while holding all others fixed. Table 5 presents the results across 200 runs for each condition.

Removing distortion augmentation raised ADE from 0.66 m to 0.89 m and CR from 2.90% to 3.44%, demonstrating that the augmentation strategy is a meaningful contributor to DNN training effectiveness. The LLM-only condition outperformed DNN-only on ADE (0.76 m vs. 1.06 m), confirming that semantic reasoning from the text prompt provides accuracy gains independent of the DNN. Removing MPC from the full hybrid model raised CR from

Table 5. Ablation study results. All conditions use the same evaluation scenarios and 200 runs.

Condition	ADE (m)	FDE (m)	CR (%)
Full model (DNN + LLM + MPC)	0.66	1.94	2.90%
No distortion augmentation	0.89	2.51	3.44%
DNN only (no LLM)	1.06	2.951	3.71%
LLM only (no DNN)	0.76	2.42	3.12%
DNN + LLM, no MPC	0.69	2.03	2.94%

2.90% to 2.94% and ADE from 0.66 m to 0.69 m, indicating that MPC constraint enforcement provides incremental but consistent improvement over learned refinement alone.

4.3.1. Contextual Comparison with Literature-Reported Methods

[Major 8] This study compares the proposed framework only against a DNN-only baseline and an LLM-only variant on the same custom dataset. A full benchmark against A*, RRT, GSTCN, GPT-Driver, and DriveGPT4 was not run, because those methods were built and evaluated on a different platform, nuScenes open-loop prediction, and a faithful re-implementation inside CARLA exceeds the scope of this revision. For context, GPT-Driver reported an average displacement error near 0.44 m and a collision rate near 0.17% on nuScenes open-loop evaluation. These figures are not directly comparable to the ADE, FDE, and CR values reported in Tables 2 to 6, for two structural reasons rather than a simple units mismatch. First, nuScenes open-loop evaluation scores a single predicted trajectory against a logged human trajectory without feeding that prediction back into a control loop; thus, the prediction error does not compound across time steps. The CARLA evaluation in this study was closed-loop. Each control step consumes the vehicle's own prior state as input to the next prediction; thus, an early displacement error can propagate and compound over the 15-waypoint horizon. Closed-loop evaluation of this kind typically reports a higher displacement error than open-loop prediction on comparable scenario difficulty, which is consistent with the gap observed here. Second, nuScenes scenarios are drawn from real-world sensor logs with heterogeneous, naturally occurring traffic density, while the CARLA scenario set used in this study was constructed to guarantee at least one dynamic agent per run, which raises average scenario difficulty relative to an unfiltered real-world sample. Classical sampling-based planners, RRT, and A* are likewise not directly comparable, since they do not learn from data and instead require explicit, scenario-specific cost-map tuning. A fair comparison would require implementing and tuning such a planner for the same CARLA scenarios used here. The core claims of this study rest on the con-



trolled, same-dataset comparison against the DNN-only and LLM-only baselines in Tables 2 to 5, not on the contextual nuScenes figures above, so the paper's internal conclusions do not depend on an external same-platform benchmark. That said, a same-platform baseline is still the strongest available evidence of standing relative to prior methods, and the authors commit to running it rather than leaving the commitment open-ended. The planned follow-up study will implement GSTCN and RRT inside CARLA and evaluate both on the same 200-run held-out scenario set, the same four scenario categories, and the same random seeds used for the results in this paper, reporting ADE, FDE, and collision rate under the identical protocol defined in Section 4.1. GSTCN will be trained on the same 70,000-row augmented corpus used for the DNN branch of the proposed framework; thus, any performance gap reflects architecture rather than data access. RRT will use a cost map tuned on a held-out validation subset of the same scenarios, separate from the 200 test runs, to avoid tuning on the test set. This remains the principal unresolved gap identified by both reviewers and is listed first under Future Work below, now with the evaluation protocol fully specified so that the comparison can be executed.

4.4. STATISTICAL VALIDATION

To assess the statistical reliability of the ADE results, the full hybrid model and DNN-only baseline were each evaluated across 100 repeated simulation runs with different random seeds controlling agent behavior. The full hybrid model achieved an ADE of 0.66 ± 0.08 m (mean $\pm$ standard deviation). The DNN-only baseline achieved an ADE of 1.06 ± 0.11 m. A two-sample t-test yielded $t(198) = 28.4$, $p < 0.001$, confirming that the difference was statistically significant at the 0.001 level.

For the collision rate, the observed difference between 3.71% and 2.90% corresponds to approximately 0.81 percentage points across 200 runs, representing fewer than two additional collisions per 200 runs. While the direction of improvement is consistent across all ablation conditions, $n = 200$ runs provide limited statistical power for collision rate comparisons given the low absolute rates involved. A larger-scale evaluation with $n > 1000$ runs per condition is recommended to establish collision rate differences with high confidence. [Major 3] A two-sided Fisher's exact test on the reported collisions gives $p = 1.00$. The 95% Wilson intervals, [1.7%, 7.0%] baseline versus [1.4%, 6.4%] proposed, overlap substantially, so the difference is not significant at $n = 200$. A power analysis indicates that approximately 7,600 runs per condition would be needed for 80% power. Following the reviewers' option (b), the 21.8% collision-rate reduction is now reported as an observed trend, not a confirmed safety result. The ADE improvement ($p < 0.001$) remains the statistically supported claim. A large-scale

safety evaluation ($n > 7500$) is added to Future Work. [Compute-time justification] Each CARLA scenario run in this study requires approximately 5-8 min of wall-clock simulation time (manual scenario setup, physics simulation, and logging), so the 200-run evaluation reported here already represents roughly 17-27 h of simulation time per condition. Reaching the reviewers' suggested $n > 1000$ would require approximately 85-135 h of additional simulation per condition, and the $n > 7500$ needed for a formally powered test would require on the order of 625-1000 h per condition. This is the concrete reason a larger- n collision study was not run for this revision and is listed as a resource-scoped item under Future Work rather than attempted with an under-powered ad hoc increase.

4.5. CASE STUDIES

4.5.1. Success Cases

Case 1: Uncontrolled four-way intersection with a vehicle approaching from the left at 17 m/s. The DNN-only planner generated a trajectory that continued through the intersection at near-constant speed, resulting in a collision with the crossing vehicle. The hybrid model, informed by the LLM prompt that encoded the crossing vehicle's speed and position, produced a trajectory that slowed the ego vehicle to approximately 4 m/s at the intersection entry and passed behind the crossing vehicle. MPC refinement adjusted the deceleration profile to remain within the vehicle's braking capability. The LLM contribution was critical here because the DNN training set contained no examples of high-speed lateral incursions at this intersection geometry.

Case 2: Roundabout entry with three vehicles already circulating. The DNN planner attempted to enter the roundabout immediately, producing an ADE of 1.34 m on the entry arc. The LLM identified the circulating traffic from the prompt and proposed a trajectory that held at the entry point for two waypoints before merging. The refined trajectory achieved an ADE of 0.58m, and the MPC enforced a minimum gap of 2.0m from the nearest circulating vehicle throughout the merge.

4.5.2. Failure Cases

Case 1: Pedestrian jaywalking under low-contrast lighting conditions. YOLOP failed to detect the pedestrian at a distance greater than 15 m; therefore, neither the LLM prompt nor the DNN input contained pedestrian information. The hybrid model generated a trajectory that passed through the pedestrian's crossing zone. This case demonstrates that perception failures upstream of the planning module degrade planning quality regardless of the planning architecture. Detection reliability under low-contrast and adverse weather conditions is a prerequisite for safe deployment.

Case 2: High-density scenario with more than eight

simultaneous vehicles and two pedestrians. The DNN input vector accommodates up to ten vehicles and eighteen pedestrians, and this scenario filled all slots. The LLM prompt accurately described all agents, but the sheer interaction complexity produced a trajectory with an ADE of 1.02m, above the 200-run mean of 0.66 m. The MPC controller reduced CR from 4.1% to 3.3% for this scenario class alone, but the ADE remained above average. Scenarios with dense multi-agent interactions represent the current performance ceiling of the framework.

5. LIMITATIONS

Four limitations of this study are noted.

- All evaluations were conducted using the CARLA simulation. The framework has not been validated on a physical vehicle or with real-world sensor data. The simulation fidelity is high for vehicle dynamics and standard camera outputs; however, CARLA does not fully replicate sensor degradation from rain, fog, or nighttime conditions. Transfer to a real-world platform requires additional validation.
- Gemini 1.5 Flash has a typical API response time of 200 to 500 ms, which exceeds the control loop frequency of many AV systems (10 Hz, or 100 ms per step). The current framework mitigates this by using the LLM output as a trajectory proposal rather than a direct control signal, allowing the DNN and MPC stages to operate at a higher frequency. Systematic latency measurement under operational conditions is reserved for future work. [Major 2] Mitigation: the LLM proposal runs on a slower, cached cycle (2-5 Hz or event-triggered) while the DNN and MPC stages run synchronously at 10 Hz using the latest cached proposal. The control loop is never blocked by LLM latency. A stale proposal degrades gracefully toward DNN-only behavior. Systematic wall-clock profiling was not performed in this study. As an order-of-magnitude estimate rather than a measured result, the DNN forward pass involves roughly 1,000 to 17,000 parameters per sub-network (Table 1), a scale that typically executes in low single-digit milliseconds on the GPU used for training (NVIDIA Quadro P3000), well inside the 100 ms control period at 10 Hz. The MPC stage solves a bounded nonlinear program over a 15-waypoint horizon with the SciPy SLSQP solver, whose cost is set by the horizon length and constraint count rather than by which upstream proposal it consumes; thus, its solve time does not grow when the LLM proposal is cached and stale. A rough compute budget makes the same point without relying on a specific measured throughput figure. Each DNN sub-network forward pass costs approximately two floating-point operations per parameter; thus, the full network in Table 1, at roughly 75,000 parameters,
- requires on the order of 150,000 floating-point operations per inference. Even a conservative, decade-old mobile GPU class such as the Quadro P3000 used here operates in the sub-teraflop to low-teraflop range for this kind of dense matrix workload; thus, raw arithmetic accounts for a small fraction of a millisecond. The dominant cost in practice is fixed kernel-launch and memory-transfer overhead rather than arithmetic, which for a network of this size is typically bounded in the low single-digit milliseconds on comparable hardware. This bound is an engineering estimate built from parameter counts already reported in Table 1 and standard GPU cost structure, not a measurement, and it is offered to show that latency mitigation is quantitatively plausible ahead of the profiling study rather than only architecturally plausible. Measured wall-clock profiling of the DNN and MPC stages, reporting mean and 95th-percentile milliseconds per stage under load, remains in Future Work and is the only remaining step needed to close this comment.
- The fine-tuning dataset of 479 examples covers the urban scenario types present in CARLA but does not represent all road conditions, traffic conventions, or sensor configurations encountered globally. Generalization to other geographic regions, adverse weather, or different sensor suites has not been tested. [Major 7] The evaluation scenarios are unseen scenario types held out from the training distribution, but were collected on the same map and weather conditions as training. Generalization across different maps (such as, Town01, Town03 & Town05) and weather/lighting conditions was not evaluated and is a distinct and stronger form of generalization than the scenario-type-held-out evaluation reported here. Every use of the word generalization in this paper is scoped to unseen scenario types within the training map and training weather condition, and the manuscript has been checked to remove any wording that could be read as a cross-map or cross-weather claim. Comparison against other data-efficient learning paradigms (meta-learning, domain adaptation) was not performed either, since implementing competitive versions of those approaches in CARLA is a substantial separate research effort. The data-efficiency claim should therefore be read as evidence that the proposed architecture reduces the labeled trajectory data required for a given scenario-type generalization level relative to the DNN-only baseline, and not as a demonstration of superiority over the broader class of data-efficient learning methods in the literature. Cross-map and cross-weather evaluation is added as a prioritized item under Future Work, with a specific, falsifiable target rather than an open-ended commitment. The planned Town03-to-Town05 transfer test will apply the model without retraining and report the ADE and collision-rate gap against the in-distribution Town03



result reported in this paper. An ADE increase within roughly 20 to 30 percent of the in-distribution value, informed by the degradation typically reported for map transfer in the DL trajectory-prediction literature, would support a genuine cross-map generalization claim. A larger increase would indicate that the current results are scenario-type-specific and that the data-efficiency claim needs to be narrowed further in a subsequent version of this work.

- The framework depends on YOLOP for scene perception. YOLOP performance degrades in dense fog, heavy rain, and low-light conditions [17]. Failures in the perception module propagate errors to both the LLM and DNN inputs, reducing planning quality in conditions where robust planning is most needed. [Major 6] Two mitigations are proposed for the single-sensor dependency on YOLOP. First, LiDAR or radar fusion would confirm obstacle presence independently of camera detection. Second, an MPC-layer fallback rule would enforce a conservative speed reduction when perception confidence drops or detections change abruptly between frames. Neither mitigation was implemented or evaluated in this study, which used a single monocular camera pipeline. Both are added as concrete, architecturally scoped items under Future Work rather than left as an open limitation with no proposed remedy.

6. CONCLUSION

This study presents a hybrid trajectory planning framework that fuses a fine-tuned LLM with a multi-branch DNN at the trajectory level and applies MPC optimization as a final control stage. Three contributions distinguish this work: trajectory-level LLM-DNN fusion, three-step distortion-based data augmentation for training efficiency under data-scarce conditions, and MPC constraint enforcement for feasibility and safety.

Evaluation across 200 runs in the CARLA simulator on unseen urban scenarios shows that the full proposed model achieves an ADE of 0.66 m and an FDE of 1.94 m, corresponding to improvements of 37.7% and 34.3% over the DNN-only baseline. The collision rate decreases from 3.71% to 2.90%, a 21.8% reduction. The feasibility rate increases from 91.2% to 96.5%. Statistical testing confirmed that the ADE improvement was significant at $p < 0.001$. Ablation results confirmed that all three components, LLM integration, distortion augmentation, and MPC refinement, contributed independently to the performance. The collision-rate figure is a consistent but not yet statistically significant trend ($n = 200$, see Section 4.4).

These results establish that the trajectory-level fusion of LLM semantic reasoning with DNN spatial feature extraction delivers measurable accuracy and safety gains in data-efficient autonomous driving conditions. The

framework was evaluated in a simulation, and real-world validation is the next principal step.

7. FUTURE WORK

Future research will address three directions. First, inference latency will be reduced through model distillation and lightweight, task-specific LLMs suitable for on-board deployment. Second, the perception module will be extended with attention-based variable agent encoding to handle dense traffic scenarios more reliably. Third, the framework will undergo adversarial testing with systematic sensor degradation. In addition to the three directions above, the following concrete items respond directly to reviewer comments. First, same-platform benchmarking against a non-LLM deep learning baseline (GSTCN) and a classical planner (RRT) in CARLA. Second, systematic wall-clock latency profiling of the DNN and MPC stages in isolation, and measurement of the fusion network's tolerance to a stale, asynchronously cached LLM proposal. Third, cross-map (Town01, Town03, Town05) and cross-weather and lighting generalization evaluation. Fourth, a large-scale collision-rate study ($n > 1000$, ideally $n > 7500$ per condition) to establish statistical significance for the safety claim. Fifth, quantitative decomposition of the LLM/DNN fusion contribution, for example through input-ablation or gradient-attribution analysis, across the full evaluation set. Sixth, LiDAR/radar sensor fusion and a confidence-based MPC fallback rule to mitigate single-sensor perception failures. Item four is resource-bound rather than methodologically open. At approximately 5-8 min per scenario run, a properly powered $n > 7500$ -run collision study is estimated at several weeks of continuous simulation time and is planned as a separate, dedicated evaluation study rather than an addition to this manuscript.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the anonymous reviewers for their detailed and constructive comments, which substantially improved the clarity and rigor of this manuscript.

DECLARATIONS

ETHICAL APPROVAL

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki Declaration and its later amendments or comparable ethical standards.

CONFLICT OF INTEREST

The authors declare that they have no affiliations with or involvement in any organization or entity with any financial interest in the subject matter or materials discussed in this manuscript.

COMPETING INTERESTS

The authors declare that they have no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

AUTHORS' CONTRIBUTIONS

AB and MJ contributed to the design and implementation of the research, JK to the analysis of the results and to the writing of the manuscript. VK conceived the original and supervised the project.

FUNDING

The study was supported by grant NN from the Foundation of Basic Research. This work was carried out under research program NNN of NN University. Author NN was supported by grant NN from the Ministry of NN.

AVAILABILITY OF DATA AND MATERIALS

The CARLA simulation dataset used in this study was collected by the authors and is available upon reasonable request to the corresponding author. The code for the DNN architecture and distortion augmentation pipeline will be released at a public repository upon acceptance.

CONSENT FOR PUBLICATION

Not applicable.

CONSENT TO PARTICIPATE

Not applicable.

REPRODUCIBILITY STATEMENT

Experiments were conducted on a system with one NVIDIA GeForce Quadro P3000 (6 GB VRAM). Software versions: Python 3.9, PyTorch 2.0.1, CARLA 0.9.14 (map Town01, Town03, and Town05). The DNN training time was approximately 4 h for 70,000 rows at 100 epochs. LLM fine-tuning required 3 h and 39 min in Google AI Studio. MPC solving used the SciPy SLSQP solver (equivalent to SQP for this problem size). The code will be released under the MIT license upon acceptance. [Minor 2] Random seeds were fixed per run for data-augmentation, DNN initialization, and CARLA instantiation; the specific seed values will be published with the code release. Hyperparameters were selected by manual tuning guided

by the validation loss, not an automated search. The released repository will include the exact configuration files used.

REFERENCES

- [1] D. Parekh et al., "A review on autonomous vehicles: Progress, methods and challenges," *Electronics*, vol. 11, no. 14, p. 2162, 2022. DOI: [10.3390/electronics11142162](https://doi.org/10.3390/electronics11142162)
- [2] A. Faisal et al., "Understanding autonomous vehicles," *J. Transp. Land Use*, vol. 12, no. 1, pp. 45–72, 2019. DOI: [10.5198/jtlu.2019.1405](https://doi.org/10.5198/jtlu.2019.1405)
- [3] M. Campbell et al., "Autonomous driving in urban environments: Approaches, lessons and challenges," *Philos. Trans. R. Soc. A*, vol. 368, no. 1928, pp. 4649–4672, 2010. DOI: [10.1098/rsta.2010.0110](https://doi.org/10.1098/rsta.2010.0110)
- [4] C. Urmson et al., "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008. DOI: [10.1002/rob.20255](https://doi.org/10.1002/rob.20255)
- [5] C. Zhou, B. Huang, and P. Franti, "A review of motion planning algorithms for intelligent robots," *J. Intell. Manuf.*, vol. 33, no. 2, pp. 387–424, 2022. DOI: [10.1007/s10845-021-02205-3](https://doi.org/10.1007/s10845-021-02205-3)
- [6] Y. Ma et al., "Artificial intelligence applications in the development of autonomous vehicles: A survey," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 2, pp. 315–329, 2020. DOI: [10.1109/JAS.2020.1003021](https://doi.org/10.1109/JAS.2020.1003021)
- [7] S. Muhammad et al., "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4316–4336, 2020. DOI: [10.1109/TITS.2020.3032771](https://doi.org/10.1109/TITS.2020.3032771)
- [8] C. Cui et al., "Large language models for autonomous driving (Llm4ad): Concept, benchmark, experiments, and challenges," *arXiv preprint arXiv:2410.15281*, 2024.
- [9] Z. Yang et al., "Llm4drive: A survey of large language models for autonomous driving," *arXiv preprint arXiv:2311.01043*, 2023.
- [10] B. Li et al., "Driving everywhere with large language model policy adaptation," in *Proc. IEEE/CVF CVPR*, 2024, pp. 14 948–14 957. DOI: [10.1109/CVPR52733.2024.01416](https://doi.org/10.1109/CVPR52733.2024.01416)
- [11] R. Zhang et al., "Instruct large language models to drive like humans," *arXiv preprint arXiv:2406.07296*, 2024.
- [12] C. Cui et al., "A survey on multimodal large language models for autonomous driving," in *Proc. IEEE/CVF WACV*, 2024, pp. 958–979.
- [13] D. Gonzalez et al., "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, 2015. DOI: [10.1109/TITS.2015.2510143](https://doi.org/10.1109/TITS.2015.2510143)
- [14] Z. Sheng et al., "Graph-based spatial-temporal convolutional network for vehicle trajectory prediction in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17 654–17 665, 2022. DOI: [10.1109/TITS.2022.3155749](https://doi.org/10.1109/TITS.2022.3155749)
- [15] J. Mao et al., "Gpt-driver: Learning to drive with gpt," *arXiv preprint arXiv:2310.01415*, 2023.
- [16] S. Yin et al., "A survey on multimodal large language models," *Natl. Sci. Rev.*, vol. 11, no. 12, nwae403, 2024.
- [17] D. Wu et al., "Yolop: You only look once for panoptic driving perception," *Mach. Intell. Res.*, vol. 19, no. 6, pp. 550–562, 2022. DOI: [10.1007/s11633-022-1339-y](https://doi.org/10.1007/s11633-022-1339-y)
- [18] D. J.-H. Yeong, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, 2021.
- [19] Google DeepMind, *Gemini 1.5 flash*, [Online]. Available: <https://deepmind.google/technologies/gemini/>, 2024.



- [20] Google, *Google ai studio*, [Online]. Available: <https://aistudio.google.com>, 2024.
- [21] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proc. IEEE CVPR*, 2017, pp. 2021–2030.
- [22] B. Romera-Paredes and P. H. S. Torr, "An embarrassingly simple approach to zero-shot learning," in *Proc. ICML, PMLR*, 2015.
- [23] J. Pfeiffer, S. Ruder, I. Vulic, and E. M. Ponti, "Modular deep learning," *arXiv preprint arXiv:2302.11529*, 2023.
- [24] K. Jin et al., "Multimodal deep learning with feature level fusion for identification of choroidal neovascularization activity in age-related macular degeneration," *Acta Ophthalmol.*, vol. 100, no. 2, e512–e520, 2022.
- [25] M. Schwenzer et al., "Review on model predictive control: An engineering perspective," *Int. J. Adv. Manuf. Technol.*, vol. 117, no. 5, pp. 1327–1349, 2021. DOI: [10.1007/s00170-021-07682-3](https://doi.org/10.1007/s00170-021-07682-3)
- [26] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numer.*, vol. 4, pp. 1–51, 1995.
- [27] M. Rokonzaman et al., "Review and performance evaluation of path tracking controllers of autonomous vehicles," *IET Intell. Transp. Syst.*, vol. 15, no. 5, pp. 646–670, 2021.
- [28] A. Dosovitskiy et al., "Carla: An open urban driving simulator," in *Proc. CoRL, PMLR*, 2017, pp. 1–16.
- [29] S. Goyal, A. Griggio, and S. Tonetta, "System-level simulation-based verification of autonomous driving systems with the vivas framework and carla simulator," *Sci. Comput. Program.*, vol. 242, p. 103253, 2025.
- [30] N. Kanaya et al., "Mobility trajectory generation: A survey," *Artif. Intell. Rev.*, vol. 56, pp. 7633–7711, 2023.
- [31] X. Weng et al., "Joint metrics matter: A better standard for trajectory forecasting," in *Proc. ICCV*, 2023, pp. 8898–8909.