CoLMDriver: LLM-based Negotiation Benefits Cooperative Autonomous Driving

Supplementary Material

8. Model Details

8.1. VLM-based Intention Planner

Dataset and Training. As described in Sec. 4.2.2, we adopted a three-stage training approach for the VLM planner. In the first stage, Driving Knowledge Enhancement Training, we utilized a sampled DriveLM-CARLA dataset containing 64k image-QA pairs focused on driving-related knowledge for perception, prediction, and planning. This stage was completed in a single epoch. In the second stage, Intention Tuning, the VLM was fine-tuned on 50k samples of our collected intention dataset. For each frame, the input query was structured by incorporating the transformed GT perception information, GT navigation instructions and speed into the VLM prompt. The response combined navigation and speed intentions. Finally, in the third stage, Consensus Tuning, we enriched the second-stage dataset by adding negotiation information. The VLM was fine-tuned for five epochs in the second stage and one epoch in the third stage. Key training parameters included LoRA tuning, DeepSpeed ZeRO-3 optimization, a batch size of 1, and learning rates of 1×10^{-4} for stages one and two, and 1×10^{-5} for stage three. For reference, we trained the InternVL2-4B model on 8 NVIDIA 3090 GPUs, with each epoch taking approximately 5 hours.

8.2. Intention-guided Waypoint Planner

Model Structure. The Occupancy Encoder and the Feature Encoder each comprise two convolutional layers with 32 output channels. The MotionNet Encoder includes two Spatial-Temporal Convolution blocks followed by a standard convolutional block. Each Spatial-Temporal block consists of two 2D convolutional layers and one 3D convolutional layer. The MotionNet Encoder generates an output with 256 channels. Both the speed intention and direction intention are embedded into 256-dimensional vectors, matching the output channels of the MotionNet Encoder. Similarly, the target point is transformed into a 256-dimensional vector using a three-layer MLP. Then the MLP Fuser combines the concatenated vector into 256 dimensions. The Transformer Decoder, which includes three layers, applies cross-attention and self-attention mechanisms to BEV Tokens and Command Tokens. Finally, a two-layer MLP decoder predicts 10 waypoints, which are used as control signals.

Dataset. The dataset used for training and testing the generator is derived from CARLA. The training set is constructed from data in CARLA towns 1, 2, 3, 4, and 6, while data from towns 7, 8, and 10 are used for validation, and town 5 is reserved for testing. The original training dataset consists of approximately 25k records. We extend the dataset into four categories—STOP, SLOWER, KEEP, and FASTER. This is done by first polynomial fitting the original trajectory and then sample waypoints according to the intention and environmental information. Then the actual training dataset grows to approximately 93k records. This number is slightly less than four times the original dataset size, as in certain cases, the original trajectory is too short for polynomial fitting. The perception module processes the last five frames of data, and outputs BEV features and BEV occupancy. The BEV occupancy contains six channels with a resolution of 192×96, while the BEV feature comprises 128 channels at the same resolution. Intentions are represented as indexing tensors corresponding to the category of the given extended record. Training the generator on 8 NVIDIA 3090 GPUs takes approximately 9 hours per epoch, with convergence typically achieved after 10 epochs.

Table 4. Detailed information of the 10 scenario types in InterDrive Benchmark.

Scenario Type	Scenario Category	Vehicle Count	Carla Town No.	Route Count
Straight-Straight	IC	2	05, 06, 07	4
Straight-Left	IC	2	05, 06, 07	6
Opposite Lane	IC	3, 4	05	4
Chaos	IC	6, 8	05	4
Straight-Right	LM	2	05, 06, 07	6
Neighbor Lane	LM	2	05, 06, 07	6
Left-Right	LM	3, 4	05	4
Highway-Merge	LM	3, 4	06	4
Right-Straight	LC	3, 4	05	4
Highway-Change	LC	6, 7, 8	06	4

9. InterDrive Benchmark Details

In the current iteration of the InterDrive Benchmark, we have meticulously selected 46 routes from the Town05, Town06, and Town07 scenarios within the CARLA simulator.

Route distribution. In the InterDrive Benchmark, which consists of 46 routes, 10 scenario types and 3 categories. We integrate the characteristics of the scenarios with the specific conditions of each Carla Town to ensure that each route is both challenging to complete and practically valuable. Town05, characterized by an urban environment, is

Table 5. Driving performance in InterDrive Benchmark with traffic participants.

Method	InterDrive-total				InterDrive-IC				InterDrive-LM				InterDrive-LC			
	DS↑	**RC↑**	**IS↑**	**SR↑**	**DS↑**	**RC↑**	**IS↑**	**SR↑**	**DS↑**	**RC↑**	**IS↑**	**SR↑**	**DS↑**	**RC↑**	**IS↑**	**SR↑**
VAD	25.18	75.66	0.31	0.02	22.13	61.31	0.32	0.00	35.10	88.23	0.39	0.05	7.24	76.56	0.09	0.00
UniAD	37.13	88.71	0.41	0.11	37.45	83.52	0.44	0.06	48.29	91.33	0.52	0.20	8.48	93.82	0.09	0.00
TCP	74.18	91.21	0.82	0.48	**76.26**	84.62	**0.91**	0.44	86.59	95.00	0.91	0.65	38.50	96.56	0.40	0.13
LMDrive	49.95	61.61	**0.84**	0.13	47.65	59.34	0.81	0.00	54.12	67.10	0.85	0.20	44.69	53.02	**0.87**	0.25
CoDriving	64.50	**93.58**	0.67	0.47	54.64	88.08	0.59	0.33	88.07	96.55	0.91	0.73	27.78	**98.51**	0.28	0.13
Rule-based	69.71	87.35	0.75	0.57	66.05	**88.48**	0.72	**0.50**	90.72	97.92	0.93	0.80	25.44	58.38	0.38	0.13
CoLMDriver	**77.09**	92.02	0.80	**0.63**	63.06	82.55	0.70	0.44	**94.00**	**100.00**	**0.94**	**0.85**	**66.38**	93.41	0.68	**0.50**

the most representative of the model's target application environment, hence its higher number of routes. Town06 is distinguished by its multi-lane highways, whereas Town07 primarily features rural scenarios with narrow roads. We have designed a variety of scenarios by varying the number of vehicles and the surrounding environments, which include different towns and diverse intersections, as shown in Tab. 4. Their inclusion in the benchmark is crucial for enhancing its diversity and significantly raises the complexity of driving tasks, particularly in terms of vehicle-vehicle interactions.

Scenario Settings. In order to enhance the fidelity of the simulation environment to real-world scenarios, we have introduced a certain number of additional traffic participants. Specifically, we set the number of vehicles, pedestrians, and cyclists in the environment to 50 each. This allows them to create a certain level of disturbance without completely blocking the routes and interfering with the predefined vehicle interactions. Moreover, this numerical value is also objectively close to the actual traffic conditions in real-world scenarios.

The result of the simulation with these participants are shown in Tab. 5. By comparing with Tab. 1, it can be observed that the inclusion of traffic participants has a certain impact on the methods primarily based on cooperation. In contrast, the scores of non-cooperative methods remain essentially unchanged or even slightly improve. This is because these participants still prevent the originally designed vehicle conflicts in specific scenarios.

10. Inference Latency Details

Fig 6 shows the driving scores with/without considering inference latency, using GPU: NVIDIA RTX 3090 and CPU: AMD EPYC 7542. Given that the RTX 3090 (284.7 TOPS) provides similar compute capability to the car-grade AI chip Orin-X (254 TOPS), our method shows feasibility for real-world deployment. Tab. 6 shows how latency affects performance. Tab. 7 reports compared methods latency.

The inference latency increases rapidly with the growth of vehicle numbers. We tackle this with two techniques: (i) agent grouping to limit negotiation scope, (ii) a dual system to manage driving during latency. Tab. 8 presents statistics from latency-aware experiments, showing that negotiation time grows slower than linearly by dynamic grouping as vehicle numbers increase. Our method handles multi-agent negotiation and clearly outperforms non-negotiation approaches.

Table 6. Driving Score↑ across various cooperation latency.

Latency / s	0.5	1	2	No negotiation
CoLMDriver	79.42	76.87	71.35	47.64
CoDriving	71.53	69.68	62.31	/

Table 7. Methods' inference latency on RTX 3090

Method	VAD	UniAD	TCP	LMDrive	Codriving	CoLMDriver	
						negotiation	low-level planning
Latency	0.22s	0.89s	0.12s	0.26s	0.34s	0.76s/round(2 car)	0.12s

Table 8. Effect of vehicle number.

Metrics	Negotiation	Vehicle number		
		2	3	4
Negotiation period (s/round)	✓	0.76	0.78	1.22
Driving Score↑	✓	100.0	97.5	86.7
	✗	64.0	63.2	41.7

11. Robustness Experiments

CoLMDriver is feasible to real-world application. The system inputs sensor data and outputs executable steer, throttle, and brake, conforming to real-world situation. Tab. 9 shows robustness under pose noise (follow coalign[1]) compared with non-negotiation approach and cooperative driving baseline CoDriving.

Table 9. Driving Score↑ under pose noise situation.
gray: the improvement over the non negotiation approach.

Methods	pose noise std ($m/°$)		No negotiation
	0.0	0.6	
CoLMDriver	88.53(+40.89)	83.96(+36.32)	47.64
CoDriving	74.13	65.42	/

[1]https://doi.org/10.48550/arXiv.2211.07214

12. Prompt Details and Example

12.1. Prompt for VLM

To better harness the knowledge and reasoning capabilities of the VLM, as well as to standardize its output format, we designed the VLM prompt based on the following structure: role definition, task description, logical guidance, additional rules, real-time input, and output format. The specific prompt design is detailed in Lst. 1. The content in '{}' will be replaced by real-time input.

12.2. Prompt for LLM

According to the design of our negotiation module, the prompts designed for the LLM consist of three types: vehicle-to-vehicle communication, sum actions, and consensus scoring.

In each round of negotiation, each vehicle broadcasts messages based on the prompts required for communication, and subsequently, one vehicle acts as a critic to sum actions and score. The prompts required for vehicle-to-vehicle communication are shown at Lst. 2, where the environmental information and message records are denoted by '{}' and will change in real-time based on the scenario. The prompts for action-summing are presented in Lst. 3, with the output being a JSON-formatted behavior request. The consensus scoring is then conducted using the prompts designed for evaluation, as shown in Lst. 4, to complete one round of negotiation. Herein, the placeholder '-conv-' will be dynamically replaced with the current message record.

13. Autonomous Vehicle Details

The autonomous vehicle in CoLMDriver processes sensor data and produces control signals as its final output. This section offers a detailed introduction to the sensor setup and controller configuration.

Sensor configurations. In CoLMDriver, we use the front-facing image with a resolution of 3000×1500 and a horizontal field of view (FoV) of 100° as an input for the VLM-based intention planner. For 3D detection, we rely on point clouds generated by a 64-channel LiDAR mounted at a height of 1.9 meters, with an upper FoV of 10° and a lower FoV of -30°.

Controller configurations. The controller generates executable driving actions, including steering, throttle, and braking, based on the predicted waypoints. To achieve this, we employ two PID controllers: a lateral controller and a longitudinal controller, which produce the corresponding control signals. The lateral signal (turn signal) is calculated using the angle between the last two predicted waypoints, while the longitudinal signal (speed signal) is calculated using the average displacement in the predicted waypoints. Subsequently, we use the PID controller to generate a relatively smooth output. Mathematically, let $E \in \mathbb{R}^N$ be the historical signal with a time length of N, each PID controller takes the current signal x as input and outputs $x' = K_P * x + K_I * \text{MEAN}(E) + K_D * (E[-1] - E[-2])$, where [K_P, K_I, K_D, N] forms a set of hyper-parameters for a PID controller. Specifically, the lateral controller is configured with [1, 0.2, 0.1, 5], while the longitudinal controller uses [5, 1, 0.1, 20].

```
Suppose you are an autopilot assistant driving a car on the road. You will receive images from the car'
s front camera and are expected to provide driving intentions. There are other traffic participants in
the scenario, and you may have communication with them. Your analysis logic chain should be as follows:
1. Understand the direction of the road and your own position.
2. Perceive surrounding objects.
3. Pay attention to key objects and dangerous situations.
4. Follow the rules listed below.
5. Check communication decision.
6. Finally, conclude the situation and provide driving intentions.

### Rules
1. If the environment is safe and clear, drive fast
2. Maintain a safe distance from the car in front.
3. Stop to avoid pedestrians preparing to cross the road.
4. Slow down or stop when other vehicles change lanes, merge or turn.
5. Slow down or stop when there is obstacle on the road ahead.
6. When establishing communication with other vehicles, take the communication decision as important
reference.

### Perception Results
{perception}
### Real-time Inputs
Negotiation suggestion: {negotiation message}
Target direction: {navigation instruction}
Current Speed: {speed} m/s

### Output Requirements
Provide the navigation and speed intention. Navigation intention include 'turn left at intersection', '
turn right at intersection', 'go straight at intersection', 'follow the lane', 'left lane change', '
right lane change'. Speed intention include STOP, FASTER, SLOWER, KEEP.
```

Listing 1. VLM intention generation prompt

```
## Role
You are a driving assistant of a car (Vehicle ID: {i}). Given a scenario where multiple vehicles are in
 conflict, you need to negotiate with other vehicles to reach a consensus and ensure the safety and
efficiency of all vehicles involved.

## Scenario
- Ego Vehicle (ID: {info['ego_id']}): Intention = {info['ego_intention']}, Speed = {round(info['
ego_speed'], 1)}m/s
- Surrounding Vehicles:
{veh_string}

## Traffic Rules
0. In emergency situations, allow vehicles with special circumstances to pass through first.
1. Merging cars slow down to yield to straight car.
2. Left-turn cars slow down to yield to straight/right-turn car.
3. The car being yielded to go faster.
4. Cars behind decrease speed during emergency braking.
5. Following cars maintain a safe distance.

## Task
Based on the scenario info and conversation history, analyze the situation considering the **speed,
direction, distance and intention of each vehicle**. Make sure you understand the situation before
making any decisions. Pay attention to the traffic rules and critic suggestion. Identify any potential
conflicts and propose actions that ensure the safety and efficiency of all vehicles involved. Remember
to consider others' actions and requests from previous conversations. When conflicts occur, either
request others to yield or yield to others.
Your message may contain the action you will take and requests for other vehicles. **The actions and
requests are speed intentions**

## Negotiation Tips
- Your actions should be logically consistent with your requests. No need for both sides to yield.
- Clearly specify which vehicle is responsible for each request or action.
- Focus your message on speed rather than navigation.

## Conversation History
{previous_conv}{sug_str}

## Output
You are vehicle {info['ego_id']}, you need to send a message to other cars. Please output the message
only, within 18 words. Please do not provide specific speed values; instead, describe the trend of
speed changes.
Sample output: I will [speed intention]; [requested speed intention].
```

Listing 2. LLM negotiation prompt - ego vehicle communication

```
## Task
Given a conversation of multiple cars negotiating to reach consensus, classify each vehicle’s speed
change into [STOP, SLOWER, KEEP, FASTER] and output the result as a string in format: {’id’: car_id, ’
speed’: category}.

## Classification rules
- STOP: Come to a complete stop.
- SLOWER: Decrease speed.
- KEEP: Maintain current speed.
- FASTER: Increase speed.

## Additional rules:
- If a car request others to yield, it should go faster
- If a car yield to others, it should stop

## Input conversation:
{conv}
Your task is to analyze the given conversations for each vehicle and output the classification as a
string in the specified format. DO NOT output other content other than the required actions. Ensure the
 output matches the required structure exactly.

## Output example:
{"0": {"speed": "STOP"}, "1":{"speed": "SLOWER"}, "2":{"speed": "SLOWER"}...}
```

Listing 3. LLM negotiation prompt - sum actions

```
Task Description:
Please analyze the following conversation and determine whether the characters have reached a consensus
 in the given scenario. Your response should include two parts: the first part is a brief explanation
of whether a consensus was reached; the second part is a score indicating the degree of consensus,
ranging from 0 to 100, where 0 means no consensus at all, and 100 means complete consensus.

Scoring Criteria:
0-20: There are significant disagreements with almost no common ground.
21-40: While there are some disagreements, there are one or two points where both parties can accept
each other’s views.
41-60: There is a moderate level of compromise and understanding on most discussed topics, but
important disagreements remain unresolved.
61-80: Consensus has been reached on most issues, with only minor differences of opinion on a few
details.
81-100: Almost all issues have been agreed upon by all parties, with only negligible objections
remaining.

Scenario: On the road, multiple cars may have driving conflicts now. They negotiate with each other to
avoid conflict.
Conversation:
{conv}

Your output format:
Short analysis: very short sentence to sum the consensus situation of the conversation.
Consensus score: int
```

Listing 4. LLM negotiation prompt - consensus score evaluation