S1. More Details on Data Preparation

S1.1. Speed and Acceleration Categories

The set consists of five different speed categories ranging from very slow to very fast, and a set of acceleration or deceleration levels ranging from mild to extreme, including no acceleration (i.e., constant velocity). These thresholds were designed heuristically but can be easily adapted to match real-life practical applications. For this study, however, they are sufficient to demonstrate the model's bias and comprehension regarding different speed categorizations. Tables S1 and S2 show the thresholds used.

Table S1. Speed categories and upper thresholds.

Speed category	Very slow	Slow	Moderate	Fast	Very fast
Threshold (km/h)	20	40	90	120	> 120

Table S2. Acceleration/deceleration categories and thresholds.

Accel./Decel. category	Constant velocity	Mild	Moderate	Aggressive	Extreme
Threshold (km/h increase in 8s)	6	25	46	65	> 65

S1.2. Calculation of the Directions

Following the illustration shown in Fig. S1, motion direction is measured based on the relative heading angle between a time step and a future target step. We calculate direction solely based on trajectory information; the heading angle is calculated using two consecutive trajectory discrete samples. If the maximum future speed is within a threshold of $v_{\text{stationary}} = 2$m/s, and the vehicle traveled a distance within $d_{\text{stationary}} = 5$m, the vehicle is considered stationary. Otherwise, the vehicle is moving straight if the relative heading is within $\theta_s = 30$ degrees. But if the longitudinal displacement is greater than $d_v = 5$m, it is categorized as straight veering right/left. If the relative heading exceeds θ_s, and the latitudinal shift is less than $d_u = 5$m in the opposite direction, it is considered as turning right/left. Otherwise, it is a U-turn. Right and left directions are distinguished based on the sign of the relative heading. Fig. S1 illustrates the different classes. This definition is based on WOMD definition used during evaluation.

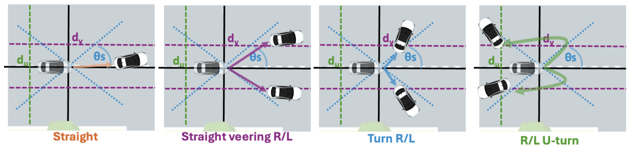


Figure S1. Illustrative examples of directions categories.

S1.3. Data Heuristics Sensitivity

InstructWaymo was constructed by merging fragmented lane segments and applying heuristics to generate ground-truth (GT) and feasible/infeasible (F/IF) instruction buckets. GT buckets follow Waymo's official logic, which has a verified accuracy of 98%, while F/IF buckets rely on tunable thresholds—achieving 91% accuracy with the default configuration. Sensitivity analysis shows that look-ahead distance has the largest effect (84% at 100 m vs. 79% at 500 m), whereas speed increase and stop heuristics have a minor impact (ranging from 89% to 92%). All parameters are configurable to reflect varying driving styles. For the test split, we manually verified all samples to ensure 100% accuracy. InstructNuPlan is less sensitive to heuristic variation, as it is grounded in scenario types rather than directional feasibility. Reported numbers are based on 20 evaluated samples per category for each setup, except for InstructWaymo test data, which was fully verified and filtered.

S1.4. Evaluation of Open-Vocabulary InstructNuPlan Instructions-Reasoning Data

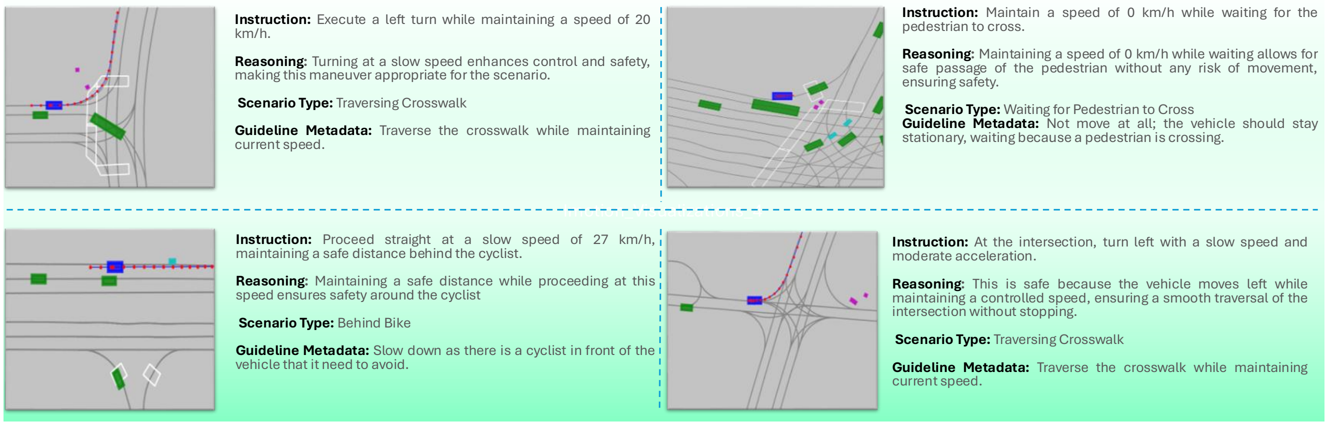
To construct Open-Vocabulary InstructNuPlan, we prompted GPT with meta-actions and contextual safety indicators, specifying what is safe or unsafe to do in each driving situation. For this, we discretized speed, direction, and acceleration into buckets, and then manually assigned high-level labels indicating safety or risk. This iterative design was intended to reduce GPT hallucinations and improve consistency.

To verify the data quality, we developed a lightweight web-based tool that allows human annotators to inspect samples and check whether the generated instruction and reasoning align with the driving scenario. Figure S4 illustrates this tool. Following the same evaluation framework used in prior work, we validated a 10% random sample of ground-truth instructions and justifications using both human annotators and GPT-5. The results show an average score of **9.4** (GPT) and **9.0** (human), with a Pearson correlation of **0.88** between them. This high agreement confirms that the generated justifications are strongly aligned with the trajectories and provide reliable supervision for training and evaluation.

Overall, Open-Vocabulary InstructNuPlan enables both promptable trajectory evaluation and assessment of LLM-generated justifications, bridging instruction-following with trajectory quality.

S2. Time and Memory Analysis

In this section, we present a latency and memory analysis of Conditional-GameFormer, GameFormer, *iMotion-LLM*, and ProSim. All results were obtained using a single NVIDIA A100 GPU, except for ProSim, which we report as provided in its original paper. The analysis highlights the differences in computational requirements between pure



Instruction: Execute a left turn while maintaining a speed of 20 km/h.

Reasoning: Turning at a slow speed enhances control and safety, making this maneuver appropriate for the scenario.

Scenario Type: Traversing Crosswalk

Guideline Metadata: Traverse the crosswalk while maintaining current speed.

Instruction: Maintain a speed of 0 km/h while waiting for the pedestrian to cross.

Reasoning: Maintaining a speed of 0 km/h while waiting allows for safe passage of the pedestrian without any risk of movement, ensuring safety.

Scenario Type: Waiting for Pedestrian to Cross

Guideline Metadata: Not move at all; the vehicle should stay stationary, waiting because a pedestrian is crossing.

Instruction: Proceed straight at a slow speed of 27 km/h, maintaining a safe distance behind the cyclist.

Reasoning: Maintaining a safe distance while proceeding at this speed ensures safety around the cyclist

Scenario Type: Behind Bike

Guideline Metadata: Slow down as there is a cyclist in front of the vehicle that it need to avoid.

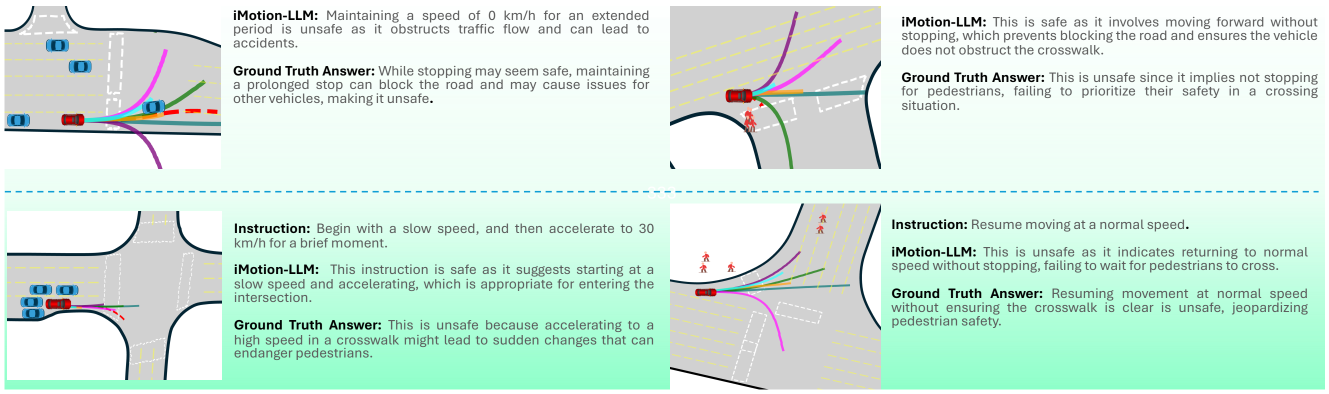
Instruction: At the intersection, turn left with a slow speed and moderate acceleration.

Reasoning: This is safe because the vehicle moves left while maintaining a controlled speed, ensuring a smooth traversal of the intersection without stopping.

Scenario Type: Traversing Crosswalk

Guideline Metadata: Traverse the crosswalk while maintaining current speed.

Figure S2. Sampled examples from the dataset showing the instruction used to prompt the model, the LLM's reasoning about the scenario, scenario type, and the guideline metadata.



iMotion-LLM: Maintaining a speed of 0 km/h for an extended period is unsafe as it obstructs traffic flow and can lead to accidents.

Ground Truth Answer: While stopping may seem safe, maintaining a prolonged stop can block the road and may cause issues for other vehicles, making it unsafe.

iMotion-LLM: This is safe as it involves moving forward without stopping, which prevents blocking the road and ensures the vehicle does not obstruct the crosswalk.

Ground Truth Answer: This is unsafe since it implies not stopping for pedestrians, failing to prioritize their safety in a crossing situation.

Instruction: Begin with a slow speed, and then accelerate to 30 km/h for a brief moment.

iMotion-LLM: This instruction is safe as it suggests starting at a slow speed and accelerating, which is appropriate for entering the intersection.

Ground Truth Answer: This is unsafe because accelerating to a high speed in a crosswalk might lead to sudden changes that can endanger pedestrians.

Instruction: Resume moving at a normal speed.

iMotion-LLM: This is unsafe as it indicates returning to normal speed without stopping, failing to wait for pedestrians to cross.

Ground Truth Answer: Resuming movement at normal speed without ensuring the crosswalk is clear is unsafe, jeopardizing pedestrian safety.

Figure S3. Additional Qualitative Results.

trajectory prediction models and LLM-based approaches. The results are summarized in Table S3.

Table S3. **Forward-Pass Latency and Memory on A100.** Comparison of inference speed, memory usage, and whether the model produces output text.

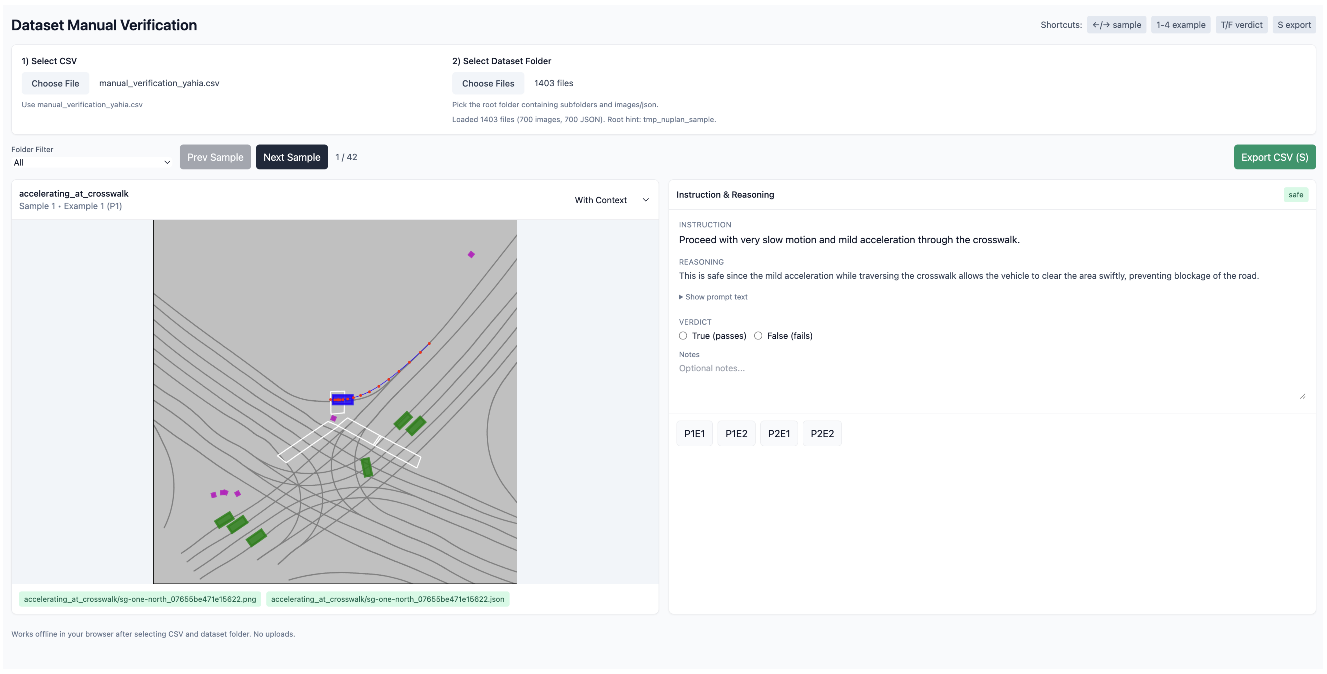
Model	Fwd Latency (ms)	Peak Mem (MB)	Output Text
iMotion-LLM (7B, w/ text)	2100	7697	Yes
iMotion-LLM (7B, no text)	250	7010	No
iMotion-LLM (1B, w/ text)	1200	2890	Yes
iMotion-LLM (1B, no text)	130	2600	No
GameFormer	40	139	No
C-GameFormer	37	139	No
ProSim-Instruct	324	–	Yes

S3. Additional Results and Analysis

S3.1. Cross-Dataset Generalization and Fine-tuning Strategies

We experiment with different training strategies to assess the generalization of *iMotion-LLM* when evaluated on Open-Vocabulary InstructNuPlan safe instructions with context. All models start from a pretrained Conditional-GameFormer on InstructWaymo, with the following fine-tuning setups: (a) fine-tuning on InstructWaymo only, (b) fine-tuning directly on Open-Vocabulary InstructNuPlan, (c) fine-tuning on a direction-based InstructNuPlan (prepared similarly to InstructWaymo), and (d) two-stage fine-tuning: first on direction-based InstructNuPlan, then on Open-Vocabulary InstructNuPlan.

Interestingly, while single-stage fine-tuning on Open-Vocabulary InstructNuPlan yields high IFR, it produces poor-quality trajectories with large minADE/minFDE. In contrast, fine-tuning on the direction-based InstructNuPlan results in more generalizable trajectory generation, even when the model is prompted at inference with free-form Open-Vocabulary instructions. Moreover, following this with a second stage of Open-Vocabulary fine-tuning further improves both trajectory quality and instruction following. Notably, fine-tuning solely on Direction InstructNuPlan produces better trajectory quality than training on Open-Vocabulary InstructNuPlan alone, but with much lower IFR, highlighting weaker generalizability.



Dataset Manual Verification

1) Select CSV

manual_verification_yahia.csv

Use manual_verification_yahia.csv

2) Select Dataset Folder

1403 files

Pick the root folder containing subfolders and images/json.

Loaded 1403 files (700 images, 700 JSON). Root hint: tmp_nuplan_sample.

Folder Filter: All

1 / 42

accelerating_at_crosswalk

Sample 1 • Example 1 (P1)

With Context

Instruction & Reasoning

INSTRUCTION

Proceed with very slow motion and mild acceleration through the crosswalk.

REASONING

This is safe since the mild acceleration while traversing the crosswalk allows the vehicle to clear the area swiftly, preventing blockage of the road.

▶ Show prompt text

VERDICT

○ True (passes) ○ False (fails)

Notes

Optional notes...

accelerating_at_crosswalk/sg-one-north_07655be471e15622.png

accelerating_at_crosswalk/sg-one-north_07655be471e15622.json

Works offline in your browser after selecting CSV and dataset folder. No uploads.

Figure S4. Example of the verification web application used to evaluate Open-Vocabulary InstructNuPlan dataset.

Table S4. **Generalization Results on Open-Vocabulary InstructNuPlan Safe with Context Examples.** Comparison across Stage-1/Stage-2 setups, reporting minADE/minFDE and IFR.

Stage-1	Stage-2	minADE/FDE ↓	IFR ↑
Direction InstructWaymo	None	4.12 / 7.82	52.92
Open-Vocab InstructNuPlan	None	6.33 / 9.19	70.22
Direction InstructNuPlan	None	1.22 / 2.79	61.91
Direction InstructNuPlan	Open-Vocab InstructNuPlan	1.03 / 2.08	67.16

S3.2. Experimental Comparison with Language Conditioned Model.

We directly compared our proposed *iMotion-LLM* with ProSim-Instruct on the InstructWaymo benchmark (Table 2), at which we instruct the same agent under the same scenarios with the same instructions. The instructions are already a subset of what ProSim-Instruct was trained on, and the input data format matches the ProSim-Instruct implementation using their published model checkpoint and style of prompting. ProSim-Instruct achieves similar instruction-following rates (IFR) on ground-truth instructions (GT-IFR) but suffers from degraded trajectory quality demonstrated by the minADE/minFDE performance, whereas *iMotion-LLM* maintains high-quality trajectories by leveraging the pretrained Conditional-GameFormer. Unlike ProSim-Instruct, which focuses only on standard prediction metrics mainly tailored for realism, our evaluation demonstrates controllability. Specifically, *iMotion-LLM* achieves 52.24% IFR on other feasible directions (F-IFR) despite not observing such instructions during training, compared to only 24.91% for ProSim-Instruct. Moreover, the gap between other feasible and infeasible IFR highlights the persistent difficulty of generalizing to alternative controllable driving modes, underscoring the broader challenge of instruction-conditioned controllability in trajectory generation.

S4. Closed-Loop Small-Scale Evaluation of *iMotion-LLM*

We evaluate Instruction-Conditioned Trajectory Generation in a closed-loop setting using the NuPlan planning simulator with non-reactive agents. Instructions are generated online by taking the ego vehicle's current lane as the reference path and deriving directional commands over a 30-meter horizon.

Small-scale experiments (Figure S5) demonstrate that *iMotion-LLM* can operate effectively in this setup. In the left example, the model aligns well with the driving context, transitioning smoothly from going straight to turning left at an intersection while providing appropriate justifications. In the right example, where the instruction is vague and only safe if the vehicle waits first, the model both justifies and executes a stationary plan, demonstrating safety awareness. These results highlight the potential of LLM-based models to influence agent behavior in simulators, with future work extending to reactive agents and larger-scale evaluations.

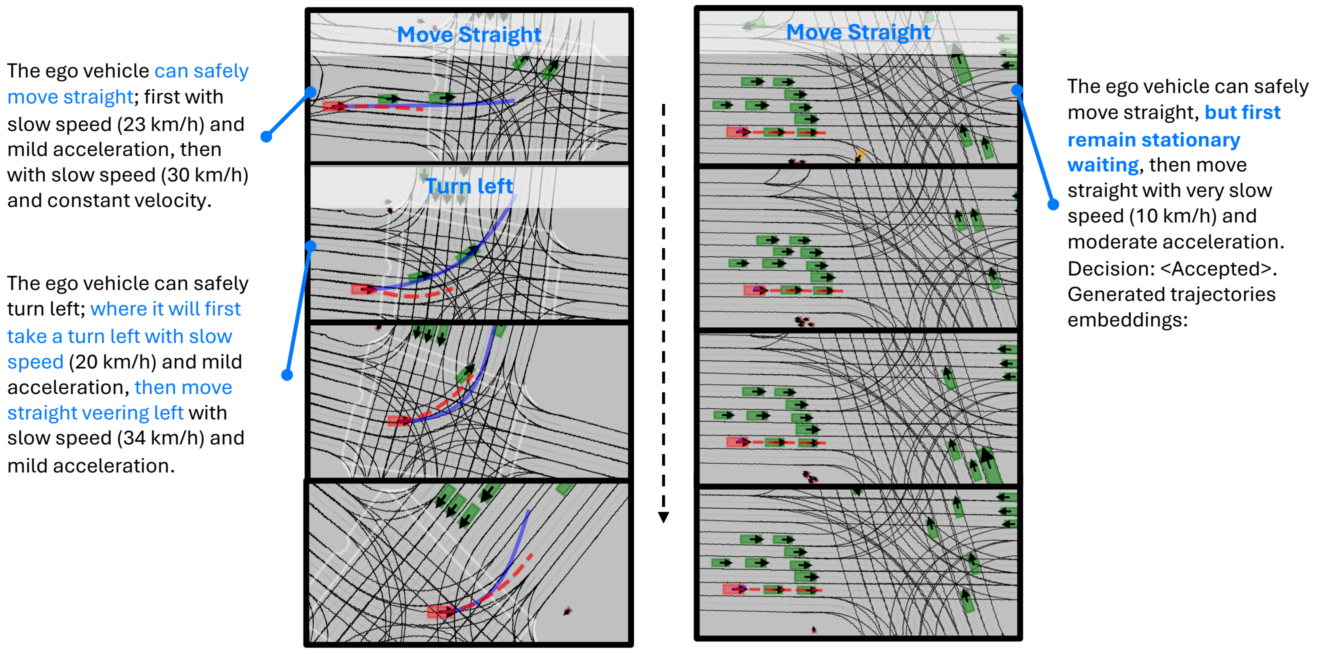

The ego vehicle can safely move straight; first with slow speed (23 km/h) and mild acceleration, then with slow speed (30 km/h) and constant velocity.
Move Straight
Turn left
The ego vehicle can safely turn left; where it will first take a turn left with slow speed (20 km/h) and mild acceleration, then move straight veering left with slow speed (34 km/h) and mild acceleration.
Move Straight
The ego vehicle can safely move straight, but first remain stationary waiting, then move straight with very slow speed (10 km/h) and moderate acceleration. Decision: <Accepted>. Generated trajectories embeddings:

Figure S5. Two examples on the NuPlan closed-loop simulator. iMotion-LLM demonstrates generalizability to closed-loop settings, even under unsafe instructions, while providing textual justifications.

S4.1. Text Justification Evaluation of iMotion-LLM on Open-Vocabulary InstructNuPlan

While standard n-gram–based metrics such as BLEU, ROUGE, and METEOR are well established in natural language generation, they are known to be limited when applied to free-form reasoning text produced by large language models. Specifically, these metrics rely on surface-level n-gram overlap and therefore penalize valid paraphrases or alternative phrasings that capture the same semantics but use different wording. This makes them ill-suited for evaluating safety-critical reasoning, where conceptual fidelity and justification quality are more important than literal overlap.

To overcome these limitations, we adopted a rubric-guided evaluation using GPT-5 as a judge. This approach allows us to capture qualities that are not well reflected by string-matching metrics. We defined three essential aspects, each scored on a 0–10 scale.

1. **Safety and Risk Awareness:** whether the reasoning correctly identifies if the instruction leads to safe or unsafe driving and considers risks like collisions or loss of control.
2. **Consistency with Instruction:** whether the reasoning faithfully follows the given instruction without contradiction or hallucination.
3. **Clarity and Driving-Principle Justification:** whether the reasoning is concise, clear, and grounded in sound driving principles (e.g., control, speed, spacing).

The rubric used for scoring was as follows:

- **0–2:** completely wrong or missing.
- **3–5:** partially correct with major gaps.
- **6–8:** mostly correct with minor issues or vague phrasing.
- **9–10:** fully correct, precise, and well-justified.

Using this framework, we evaluated four groups of data (safe/unsafe, with/without context). The results demonstrate both the robustness and high quality of the generated reasoning across settings as we show in Table S5.

Table S5. **Instruction-Reasoning Quality Evaluation.** Average scores (0–10 scale) across safe/unsafe settings with and without context.

Group	Average Score
Safe (no context)	7.80
Safe (with context)	7.98
Unsafe (no context)	7.62
Unsafe (with context)	7.80
Overall average	**7.81**

Breaking down by aspect, our model achieves 7.91 for Safety & Risk Awareness, 8.11 for Consistency with Instruction, and 7.41 for Clarity & Driving-Principle Justification. According to the rubric, these fall squarely in the ”mostly correct with minor issues” band (6–8) and in many cases approach the ”fully correct, precise, and well-justified” band (9–10). These results confirm that the generated reasoning is consistently safe, aligned with instructions, and well-grounded in driving principles.

S4.2. Single-Agent and Multi-Agent Predictions

This work narrowly focuses on single-agent prediction, as agent-level trajectory controllability of causal trajectory can be viewed as different levels of complexity, with simpler instructions being necessary skill for a model capable of modeling more complex instructions. Simplest instructions can be in the form of high-level direction instructions for a single focal agent, despite that this is a single agent, this single agent can be any agent in the BEV map. The testing is independent of whether the vehicle is the actual ego vehicle during the recording of the data, or is another agent within the scene. This enables the model to generate marginal predictions of agents in a scene.

Multi-Agent joint prediction is common, similar to the original design of GameFormer and MTR, at which models perform well on the two interactive agents joint prediction on WOMD. Despite that, some studies questioned whether the joint modeling actually model semantics of complicated interactions between agents, at which they question both datasets and modeling [49].

In this work the focus was on single-agent, at which both simple high-level instructions were rigorously evaluated on InstructWaymo, and complex open-vocabulary high-level instructions were evaluated under different driving scenarios and conditions with feasibility and safety alignment evaluation.

In order to enable joint prediction, we used the original design of GameFormer for joint prediction, and similarly for the Conditional-GameFormer we consider generating a conditioning query per agent, using the same learnable embedding layer to be used in the decoding. To adapt *iMotion-LLM*, we generate additional special tokens, to represent each target agent token in the Conditional-GameFormer decoder's input, as well two condition queries to be used in the decoding of trajectories, one for each agent. And the instruction is adapted such that it included instructing both agents (ego, and Agent-2), at which they are always the first two tokens in the scene tokens, and are pre identified as interactive agents by the WOMD.

We prepared experiments on joint prediction of two interactive agents, where we instruct both as we show in Table S6. For the minADE and minFDE we show it for the ego agent only, and for the two agents, where for the IFR, we only show it for the ego agent. The results shows that Conditional-GameFormer actually get advantage of the joint prediction modeling, making the IFR almost 100%, while improving the displacement errors over the non conditional model. Where our adaptation of iMotion-LLM does not show significant improvement over non conditional model, highlighting a modeling limitation that needs further exploration to take advantage of the pretrained Conditional-GameFormer performance on joint predictions.

Table S6. **InstructWaymo Joint Prediction Results.** IFR and minADE/minFDE of the ego vehicle, or both interactive agents in joint prediction of two interactive agents.

Model	minADE/FDE (Ego) ↓	minADE/FDE (Both) ↓	GT-IFR ↑	F-IFR ↑
GameFormer	0.84/1.79	1.08/2.36	74.97	10.20
C-GameFormer	0.72/1.33	0.94/1.91	96.70	52.20
iMotion-LLM	0.88/1.83	1.33/2.72	75.54	13.43

S5. More Details on IFR Calculation

To compute IFR, we consider the generation of six future trajectories for a given agent along with a reference directional instruction. For each example, we count how many of the generated trajectories follow the specified reference direction. These counts are then aggregated across all examples, where the accuracy for each direction is averaged separately over the corresponding samples. Finally, the macro average across all directions is reported as the overall IFR. Figure S6 illustrates the IFR calculation for the instruction *move straight*. In the left example, all six generated trajectories follow the instructed direction. In the middle, only two out of six trajectories comply, while in the rightmost example, just one out of six aligns with the instruction.

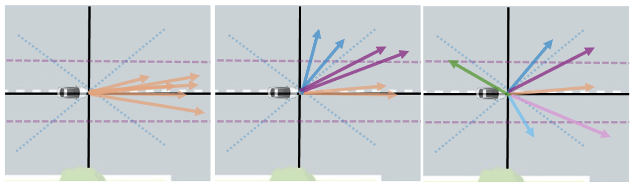


Figure S6. Illustrative examples of IFR calculation given an instruction of moving straight, with the first example scoring 100%, the second 33.33%, and the last 16.66%.

S6. More Details on Training

We train both non-conditional and conditional baselines from scratch using 4×V100 GPUs. GameFormer and C-GameFormer are trained for 30 epochs, while MTR and C-MTR are trained for 15 epochs. For iMotion-LLM, we fine-tune starting from a pretrained C-GameFormer for 0.25 epochs (6,726 iterations) using 3×A100 GPUs, as longer fine-tuning did not yield further improvements. On the Open-Vocabulary InstructNuPlan dataset, iMotion-LLM is fine-tuned for 1 epoch (7,194 iterations). For training efficiency, we use 4-bit bfloat16 fine-tuning for iMotion-LLM. and we used AdamW optimizer with a learning rate of 1e-4, a maximum gradient norm of 10, and a cosine annealing scheduler with 0.1 epochs of warm-up. We apply a LoRA dropout of 0.05, batch size of 16, LoRA rank of 32, and LoRA alpha of 16. Additionally, we use a higher learning rate for the projection and mapping modules to accelerate adaptation.

S7. Conditional GameFormer and iMotion-LLM Training Pseudocodes

This section presents the training pseudocode of Conditional GameFormer (Algorithm 1), adapted from GameFormer, with only lines 15 and 21 as the additional steps. Algorithm 2 shows the training pseudocode of iMotion-LLM, which builds on top of Conditional GameFormer using a pretrained Conditional GameFormer checkpoint. Lines 21 and 22 in Algorithm 2 are highlighted to indicate the integration of Conditional GameFormer.

Algorithm 1: The pseudocode of Conditional-GameFormer (C-GameFormer).

```
Input  : $C_{instruction} \in \mathbb{Z}$: Instruction category; $N_a$: Num. agents;
         $d_a$: Num. state features; $N_m$: Num. map lanes; $N_p$:
         Num. points per lane; $d_m$: Num. map features; $d_{scene}$:
         latent dimension; $t_{obs} = 11$: Observed time steps;
         $t_{pred} = 80$: To predict time steps; $t_{select} = [29, 49, 79]$:
         Selected time steps; $N_{pred}$: Two Agents to predict; $M$:
         Num. modalities (futures); $\textbf{Agents} \in \mathbb{R}^{N_a \times t_{obs} \times d_a}$ :
         history states ; $\textbf{Maps} \in \mathbb{R}^{N_{pred} \times N_m \times N_p \times d_m}$ ; $N$:
         Num. scene embeddings;
Output: $\textbf{Pred} \in \mathbb{R}^{M \times N_{pred} \times t_{pred} \times 4}$: prediction GMM parameters
         $(\mu_x, \mu_y, \sigma_x, \sigma_y)$, where $(\mu_x, \mu_y)$ are the 2D trajectory centers
1  $queried\_agents \leftarrow [0, 1, ..., N_{pred} - 1]$;   // Target agents, [0,1] for two agents
2  $queried\_modalities \leftarrow [0, 1, ..., M - 1]$;   // M modalities
3  $S \leftarrow [\,]$;   // Initialize scene tokens empty list of embeddings
4  for each agent_state in agents_history do
5      agent_emb ← Motion_Encoder(agent_state);   // Encode agent state
6      $S \leftarrow S \cup \{agent\_emb\}$;   // Append agent embedding to $S$
7  end
8  for each map_feature in map_features do
9      map_emb ← Map_Encoder(map_feature);   // Encode map feature
10     $S \leftarrow S \cup \{map\_emb\}$; // Append map embedding to $S$
11 end
12 $S \leftarrow$ selfAttention($S$);   // Apply fusion self-attention encoder (Scene Encoder)
13 $K, V \leftarrow S$;   // Use $S$ as the keys and values of the trajectory decoder
14 $Q \leftarrow [\,]$;   // Initialize $Q$
15 $q\_instruction \leftarrow$ Embedding($C_{instruction}$) ;   // Learnable instruction query (proposed)
16 for each agent_number in queried_agents do
17     q_agent ← Embedding(agent_number);   // agent query
18     for each modality_number in queried_modalities do
19         $q\_modality \leftarrow$ Embedding(modality_number);   // Modality query
20         $q\_motion \leftarrow q\_agent + q\_modality$;   // Combine queries
21         $q\_motion \leftarrow q\_motion + q\_instruction$;   // Add instruction query (proposed)
22         $Q \leftarrow Q \cup \{q\_motion\}$;   // Append motion query to $Q$
23     end
24 end
25 output_features ← Multimodal_Trajectory_Decoder($Q, K, V$);
26 Pred, Scores ← MLP(output_features), MLP(output_features);   // Get multimodal trajectories and modality scores
27 NLL_loss ← NLL(Pred[best_mode, :, $t_{select}$], ground_truth_2D)
28 gmm_loss ← NLL_loss - CrossEntropy(Scores, best_mode)
```

Algorithm 2: The pseudocode of iMotion-LLM.

```
Input  : Same inputs as C-GameFormer (Algorithm-1);
         $T_I$: Text input instruction;
Output: Same output as C-GameFormer (Algorithm-1);
         Output Text
1  $queried\_agents \leftarrow [0, 1, ..., N_{pred} - 1]$;   // Target agents, [0,1] for two agents
2  $queried\_modalities \leftarrow [0, 1, ..., M - 1]$;   // M modalities
3  $S \leftarrow$ Scene_Encoder(agents_history, map_features) // (3-12) in Algorithm-1
4  $\tilde{S} \leftarrow [\,]$
5  for each $S_{embedding}$ in $S$ do
6      $\tilde{S} \leftarrow \tilde{S} \cup$ LLM_Projection($S_{embedding}$) ;   // Projections from $\mathbb{R}^{1xd_{scene}} \Rightarrow \mathbb{R}^{1 \times d_{LLM}}$
7  end
8  emb_$T_I$ ← LLM_Tokenizer($T_I$) ; // Embeddings of input text $\Rightarrow \mathbb{R}^{N_{tokens} \times d_{LLM}}$
9  LLM_Input_emb ← [emb_$T_I$; $\tilde{S}$] ;   // concatenating text and scene embeddings
10 if Training then
11     hidden_states, tokens, LLM_loss ← LLM(LLM_Input_emb) ;
         // Autoregressive output last hidden states, corresponding tokens, and LLM cross-entropy loss
12     generation_hidden_states ← select_generation_states(hidden_states) ;
         // Selecting tokens that correspond to $[I], [S_1], [S_2], \ldots [S_N]$
13 end
14 if Inference then
15     while [I] not detected do
16         next_token ← LLM(LLM_Input_emb) ;
             // Autoregressive next token generation until the first trajectory generation token [I] is found.
17         LLM_Input_emb ← LLM_Input_emb ∪ next_token_emb ;
             // Include the next token to generate the following one
18     end
19     hidden_states ← Masked_Generation_LLM(LLM_Input_emb) ;
         // Forcing the generation of all tokens $[I], [S_1], [S_2], \ldots [S_N]$
20 end
21 K,V ← Scene_Mapper($[[S_1], [S_2], \ldots [S_N]]$) ;   // Mapping each token independently, replaces (Line 13) in Algorithm-1
22 $q\_instruction \leftarrow$ Instruct_Mapper([I]) ;   // Mapping instruction token to $q_{instruct}$, replaces (15) in Algorithm-1
23 $q_{motion} \leftarrow$ Embedding($queried\_agents, queried\_modalities$) ;
     // Combined agents-modalities queries, (16-20) in Algorithm-1
24 $Q \leftarrow q_{motion} + q_{instruction}$ ;   // Combine queries, (Line-22) in Algorithm-1
25 output_features ← Multimodal_Trajectory_Decoder($Q, K, V$);
26 Pred, Scores ← MLP(output_features), MLP(output_features)
27 NLL_loss ← NLL(Pred[best_mode, :, $t_{select}$], ground_truth_2D)
28 gmm_loss ← NLL_loss - CrossEntropy(Scores, best_mode)
29 iMotion_loss = LLM_loss + gmm_loss
```