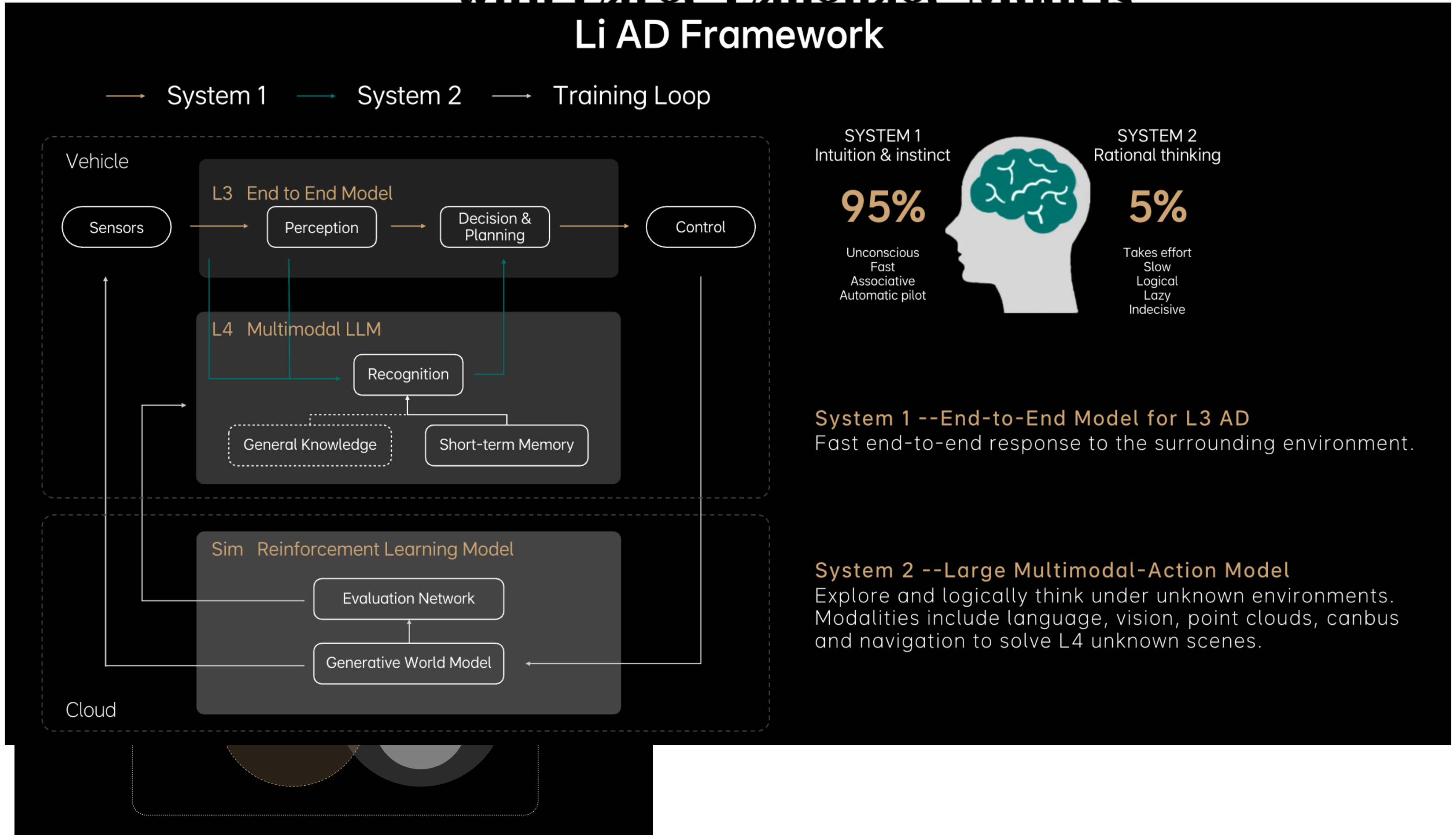
UNIVERSITY OF LIVERPOOL

Empowering Autonomous Driving with LLM and VLM

Sihao Wu

Large Language Model

Drive Like a Human: Rethinking Autonomous Driving with Large Language Models

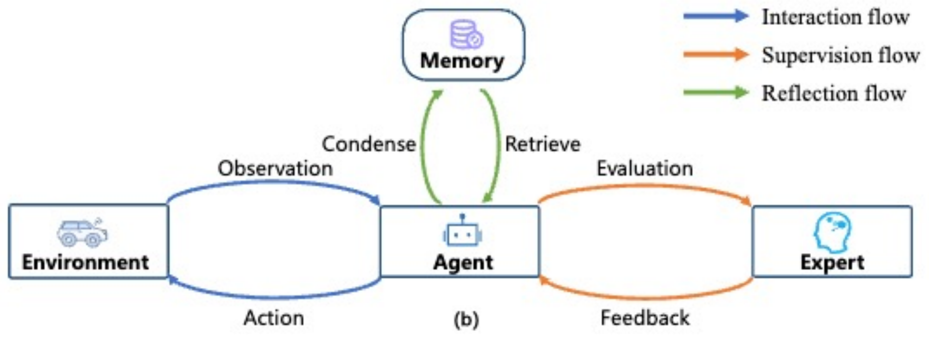


System 1 --End-to-End Model for L3 AD
Fast end-to-end response to the surrounding environment.

System 2 --Large Multimodal-Action Model
Explore and logically think under unknown environments. Modalities include language, vision, point clouds, canbus and navigation to solve L4 unknown scenes.

Towards the goal of driving like a human, we identify three abilities that are necessary:

1. Reasoning
2. Interpretation
3. Memorization



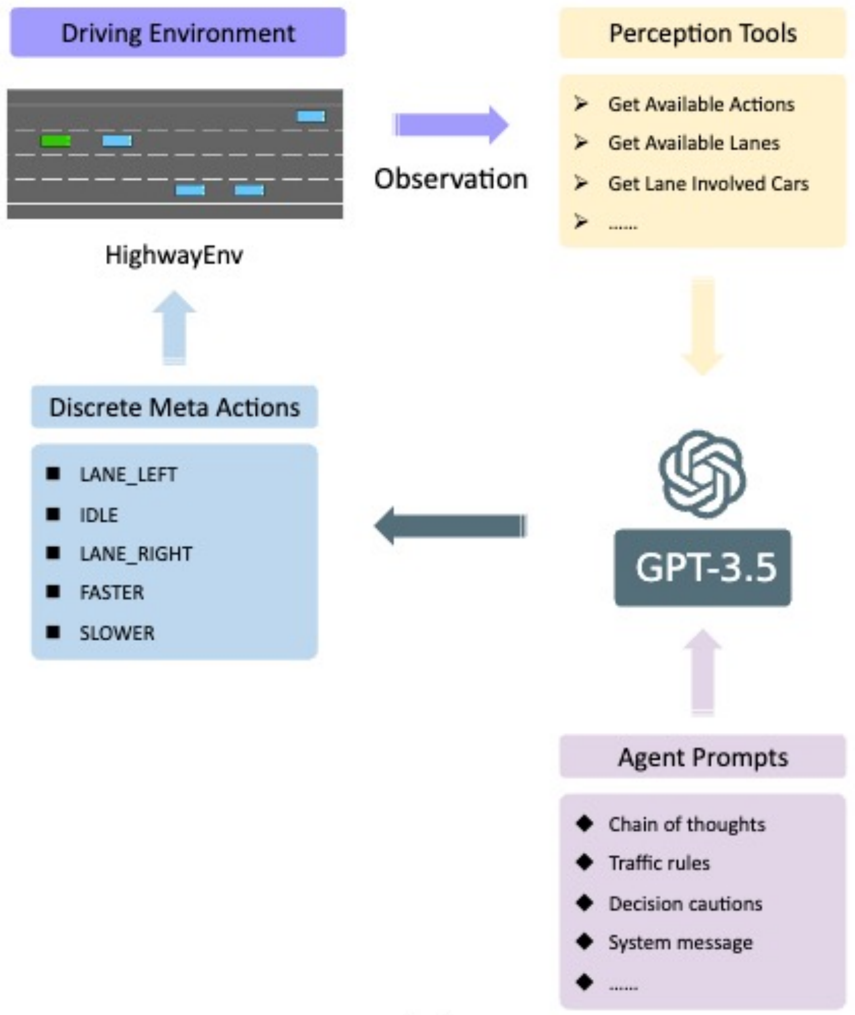
this schema includes four modules:

(1) **Environment** creates a stage that the agent can interact with by the interaction flow;

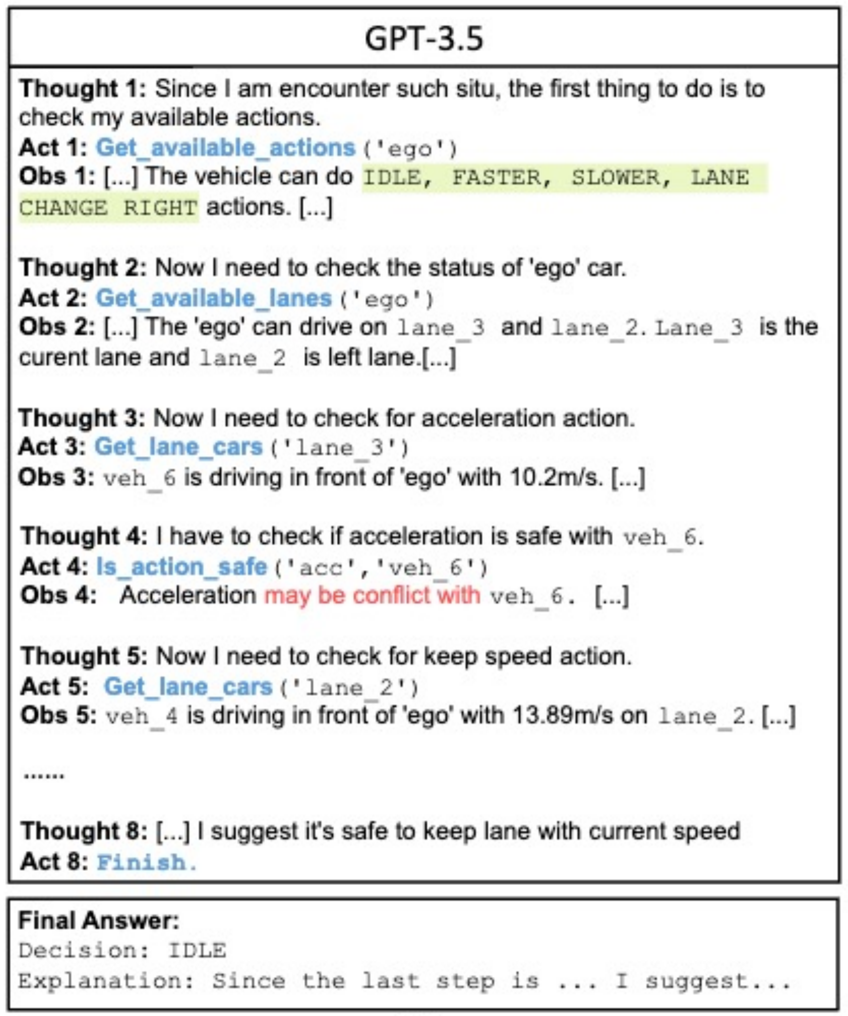
(2) **Agent** stands for a driver that can perceive the environment and make decisions utilizing its memory and learning from expert advice;

(3) **Memory** allows the agent to accumulate experience and perform actions with it via the reflection flow;

(4) **Expert** provides advice on agent training and gives feedback when it acts inconsistently, which forms the supervision flow.



(a)



GPT-3.5

Thought 1: Since I am encounter such situ, the first thing to do is to check my available actions.
Act 1: Get_available_actions `('ego')`
Obs 1: [...] The vehicle can do `IDLE, FASTER, SLOWER, LANE CHANGE RIGHT` actions. [...]

Thought 2: Now I need to check the status of 'ego' car.
Act 2: Get_available_lanes `('ego')`
Obs 2: [...] The 'ego' can drive on `lane_3` and `lane_2`. `Lane_3` is the curent lane and `lane_2` is left lane.[...]

Thought 3: Now I need to check for acceleration action.
Act 3: Get_lane_cars `('lane_3')`
Obs 3: `veh_6` is driving in front of 'ego' with 10.2m/s. [...]

Thought 4: I have to check if acceleration is safe with `veh_6`.
Act 4: Is_action_safe `('acc','veh_6')`
Obs 4: Acceleration may be conflict with `veh_6`. [...]

Thought 5: Now I need to check for keep speed action.
Act 5: Get_lane_cars `('lane_2')`
Obs 5: `veh_4` is driving in front of 'ego' with 13.89m/s on `lane_2`. [...]

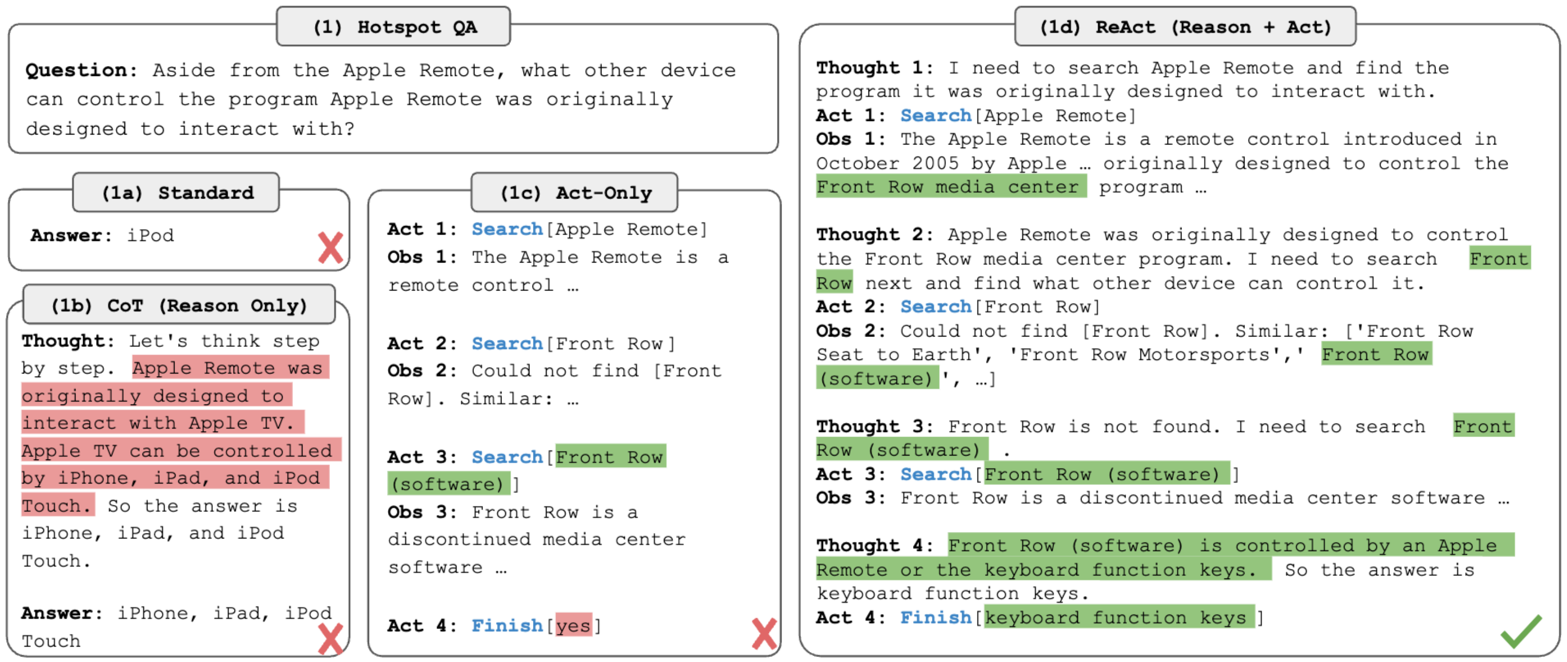
......

Thought 8: [...] I suggest it's safe to keep lane with current speed
Act 8: `Finish.`

Final Answer:
```
Decision: IDLE
Explanation: Since the last step is ... I suggest...
```

(b)

ReAct: Synergizing Reasoning and Acting in Language Models



(1) Hotspot QA

Question: Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

(1a) Standard

Answer: iPod ✗

(1b) CoT (Reason Only)

Thought: Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch.

Answer: iPhone, iPad, iPod Touch ✗

(1c) Act-Only

Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control …

Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: …

Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Act 4: Finish[yes] ✗

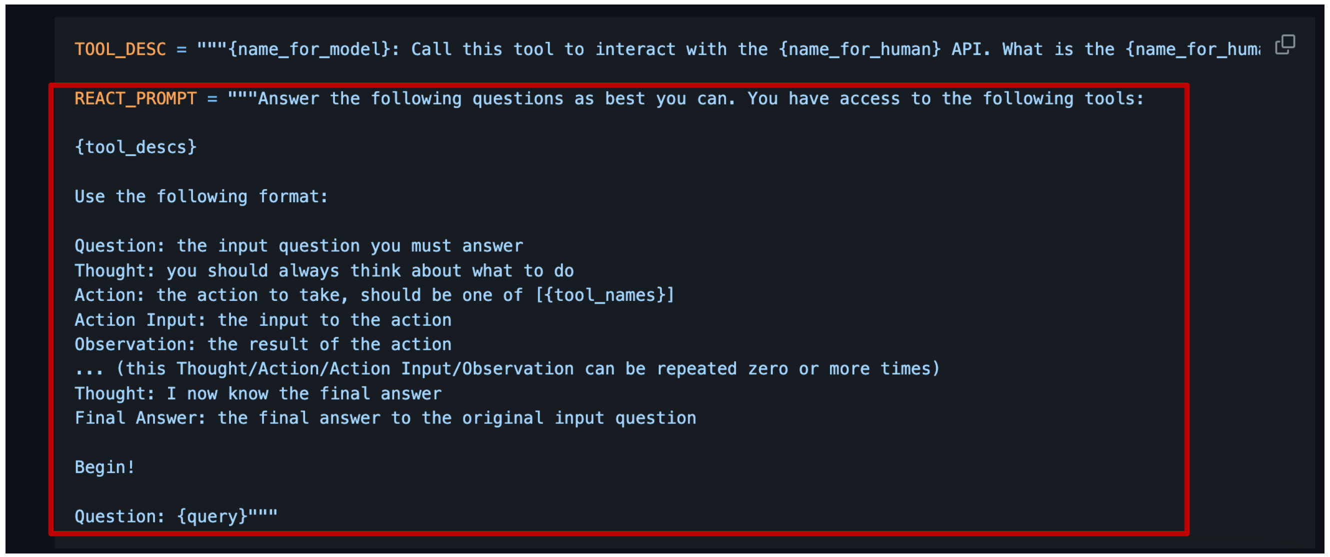
(1d) ReAct (Reason + Act)

Thought 1: I need to search Apple Remote and find the program it was originally designed to interact with.
Act 1: Search[Apple Remote]
Obs 1: The Apple Remote is a remote control introduced in October 2005 by Apple … originally designed to control the Front Row media center program …

Thought 2: Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.
Act 2: Search[Front Row]
Obs 2: Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports',' Front Row (software)', …]

Thought 3: Front Row is not found. I need to search Front Row (software) .
Act 3: Search[Front Row (software)]
Obs 3: Front Row is a discontinued media center software …

Thought 4: Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.
Act 4: Finish[keyboard function keys] ✓



```
TOOL_DESC = """{name_for_model}: Call this tool to interact with the {name_for_human} API. What is the {name_for_hum

REACT_PROMPT = """Answer the following questions as best you can. You have access to the following tools:

{tool_descs}

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can be repeated zero or more times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: {query}"""
```

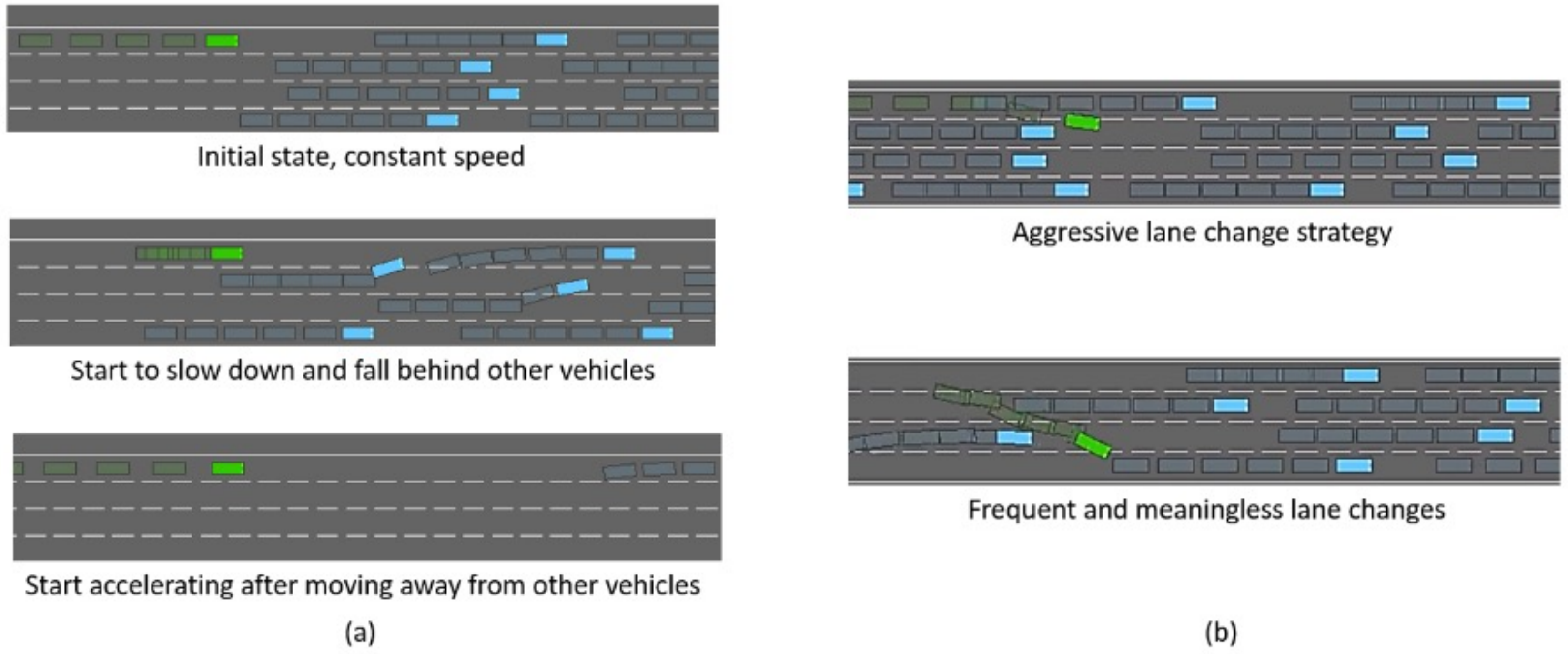
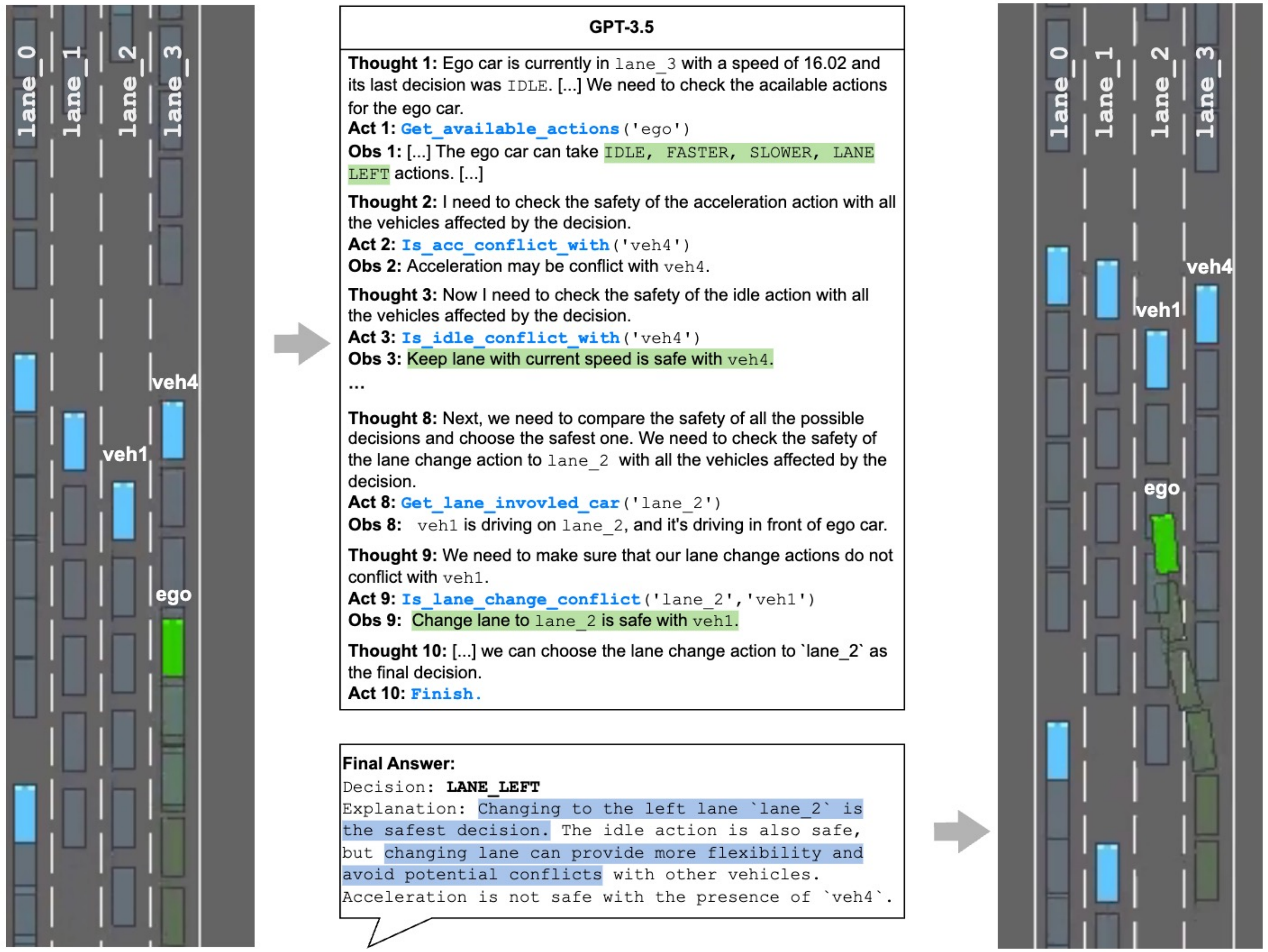

Initial state, constant speed
Start to slow down and fall behind other vehicles
Start accelerating after moving away from other vehicles
(a)
Aggressive lane change strategy
Frequent and meaningless lane changes
(b)

Figure 3: Driving behavior of RL-based and Search-based methods: (a) RL-based agents focus solely on achieving the final reward, disregarding the intermediate steps. This allows them to take unconventional actions, such as slowing down to fall behind other vehicles and then driving on an open road to avoid collisions. (b) Search-based methods make decisions by optimizing objective functions. They may pursue aggressive behavior by seeking max efficiency while ensuring safety.



GPT-3.5

Thought 1: Ego car is currently in `lane_3` with a speed of 16.02 and its last decision was `IDLE`. [...] We need to check the acailable actions for the ego car.
Act 1: `Get_available_actions('ego')`
Obs 1: [...] The ego car can take `IDLE, FASTER, SLOWER, LANE LEFT` actions. [...]

Thought 2: I need to check the safety of the acceleration action with all the vehicles affected by the decision.
Act 2: `Is_acc_conflict_with('veh4')`
Obs 2: Acceleration may be conflict with `veh4`.

Thought 3: Now I need to check the safety of the idle action with all the vehicles affected by the decision.
Act 3: `Is_idle_conflict_with('veh4')`
Obs 3: Keep lane with current speed is safe with `veh4`.
...

Thought 8: Next, we need to compare the safety of all the possible decisions and choose the safest one. We need to check the safety of the lane change action to `lane_2` with all the vehicles affected by the decision.
Act 8: `Get_lane_invovled_car('lane_2')`
Obs 8: `veh1` is driving on `lane_2`, and it's driving in front of ego car.

Thought 9: We need to make sure that our lane change actions do not conflict with `veh1`.
Act 9: `Is_lane_change_conflict('lane_2','veh1')`
Obs 9: Change lane to `lane_2` is safe with `veh1`.

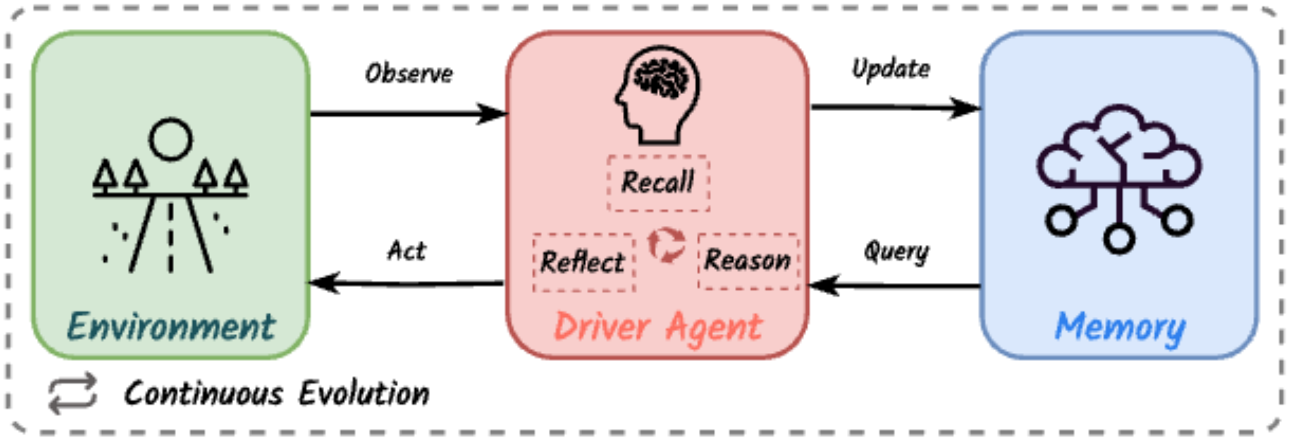
Thought 10: [...] we can choose the lane change action to `lane_2` as the final decision.
Act 10: `Finish.`

Final Answer:

```
Decision: LANE_LEFT
Explanation: Changing to the left lane `lane_2` is
the safest decision. The idle action is also safe,
but changing lane can provide more flexibility and
avoid potential conflicts with other vehicles.
Acceleration is not safe with the presence of `veh4`.
```

Figure 4: The lane-change decision-making process by GPT-3.5

DiLu: A Knowledge-Driven Approach to Autonomous Driving with Large Language Models



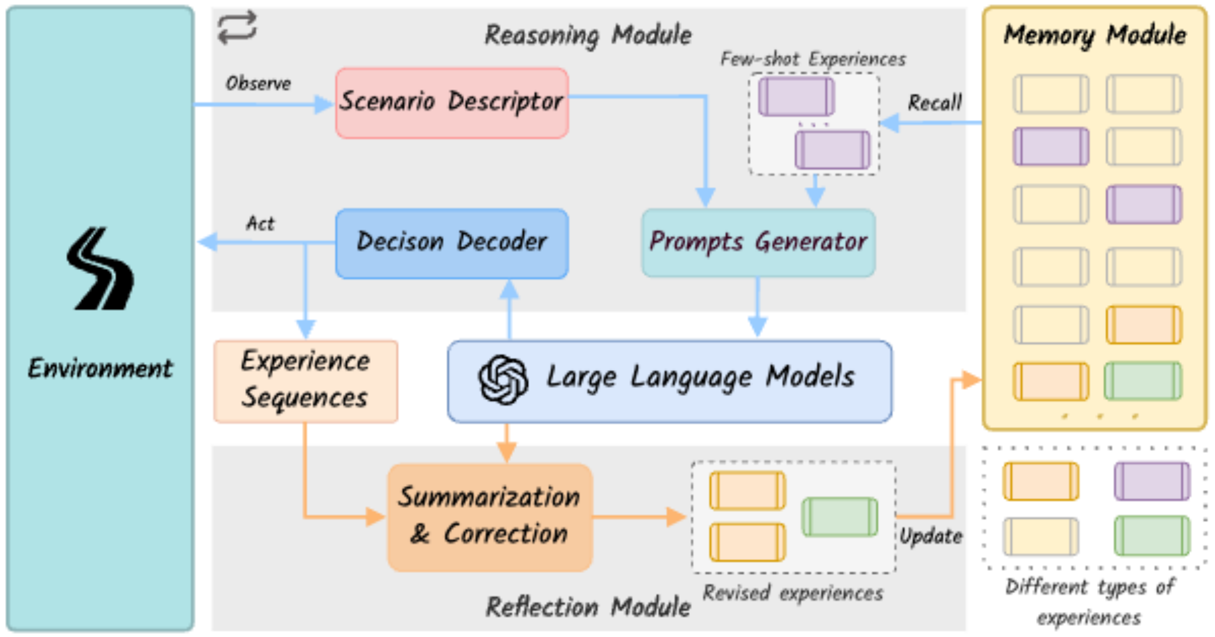
Driver agent continuously evolves to observe the environment, query, update experiences from the memory module, and make decisions to control the ego vehicle.

The contributions of our work are summarized as follows:

- To the best of our knowledge, we are the first to leverage knowledge-driven capability in decision-making for autonomous vehicles from the perspective of how humans drive. We summarize the knowledge-driven paradigm that involves an interactive environment, a driver agent, as well as a memory component.

- We propose a novel framework called DiLu which implements the above paradigm to address the closed-loop driving tasks. The framework incorporates a Memory Module to record the experiences, and leverages LLM to facilitate reasoning and reflection processes.

- Extensive experimental results highlight DiLu's capability to continuously accumulate experience by interacting with the environment. Moreover, DiLu exhibits stronger generalization ability than RL-based methods and demonstrates the potential to be applied in practical autonomous driving systems.

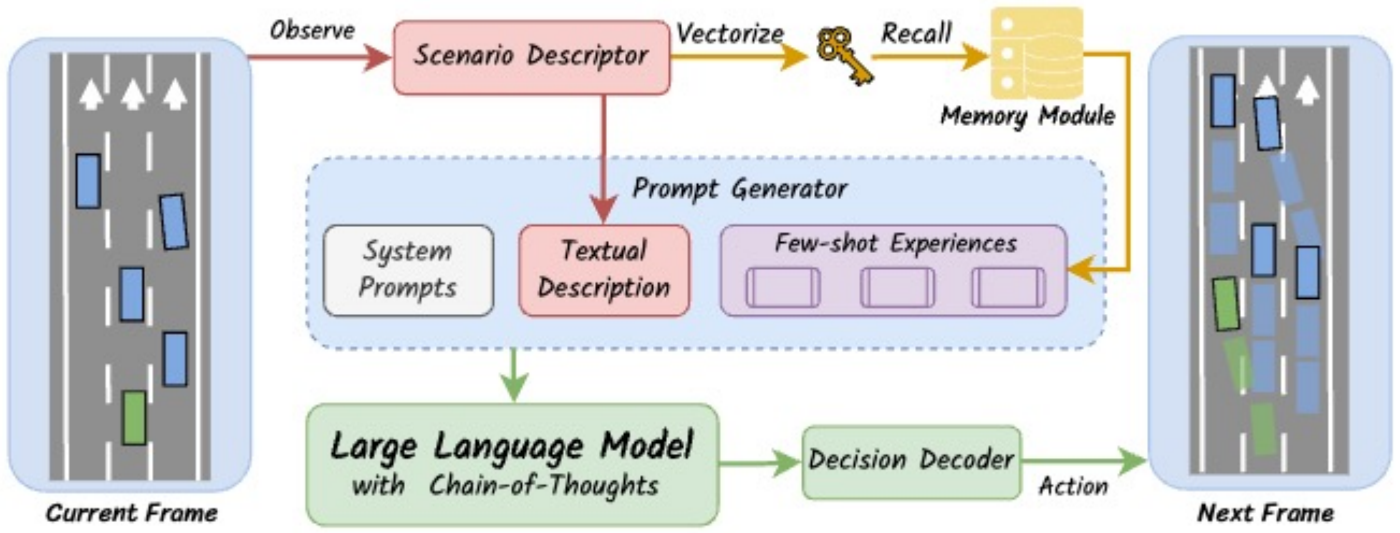


Specifically, the driver agent utilizes the Reasoning Module to query experiences from the Memory Module and leverage the common-sense knowledge of the LLM to generate decisions based on current scenarios.

It then employs the Reflection Module to identify safe and unsafe decisions produced by the Reasoning Module, subsequently refining them into correct decisions using the knowledge embedded in the LLM.

These safe or revised decisions are then updated into the Memory Module.

Without few-shot experiences, the out-of-the-box LLMs fail to perform precise reasoning when tackling the complex closed-loop driving tasks.



(1) encode the scenario by a descriptor;
(2) recall several experience from the Memory module;
(3) generate the prompt;
(4) feed the prompt into the LLM;
(5) decode the action from the LLM's response.

Figure 3: Reasoning module. We leverage the LLM's common-sense knowledge and query the experiences from Memory module to make decisions based on the scenario observation.

Memory recall: At each decision frame, the agent receives a textual description of the driving scenario. Before making a decision, the current driving scenario is embedded into a vector, which serves as the memory key. This key is then clustered and searched to find the closest scenarios (Johnson et al., 2019) in the memory module and their corresponding reasoning processes, or memories. These recalled memories are provided to the agent in a few-shot format to assist in making accurate reasoning and decisions for the current scenario.

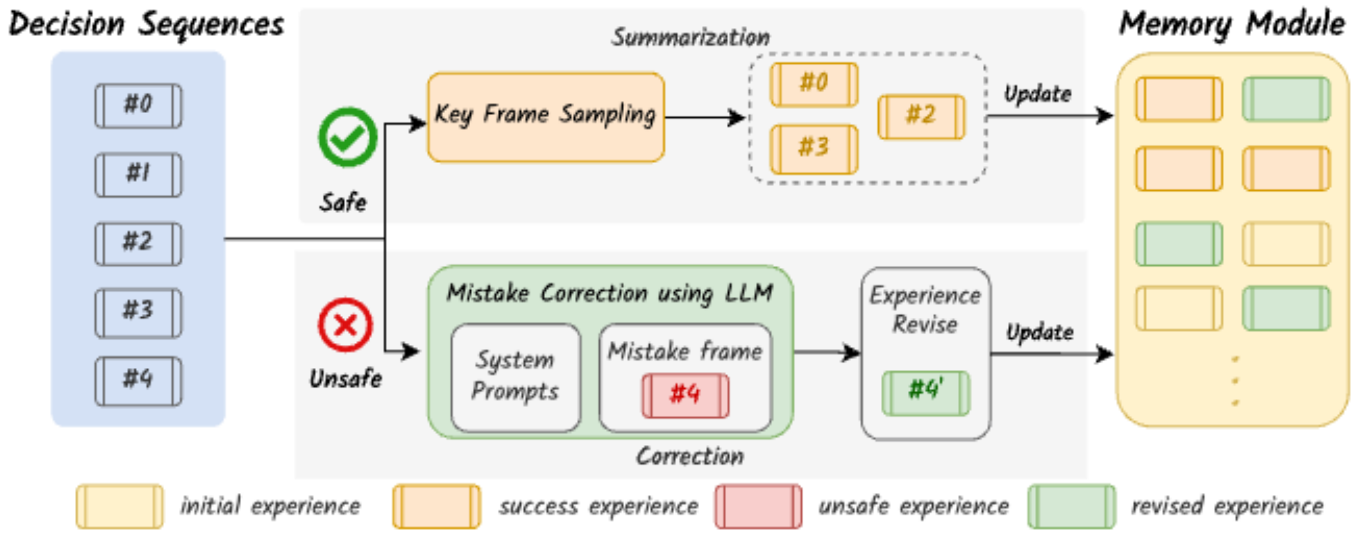
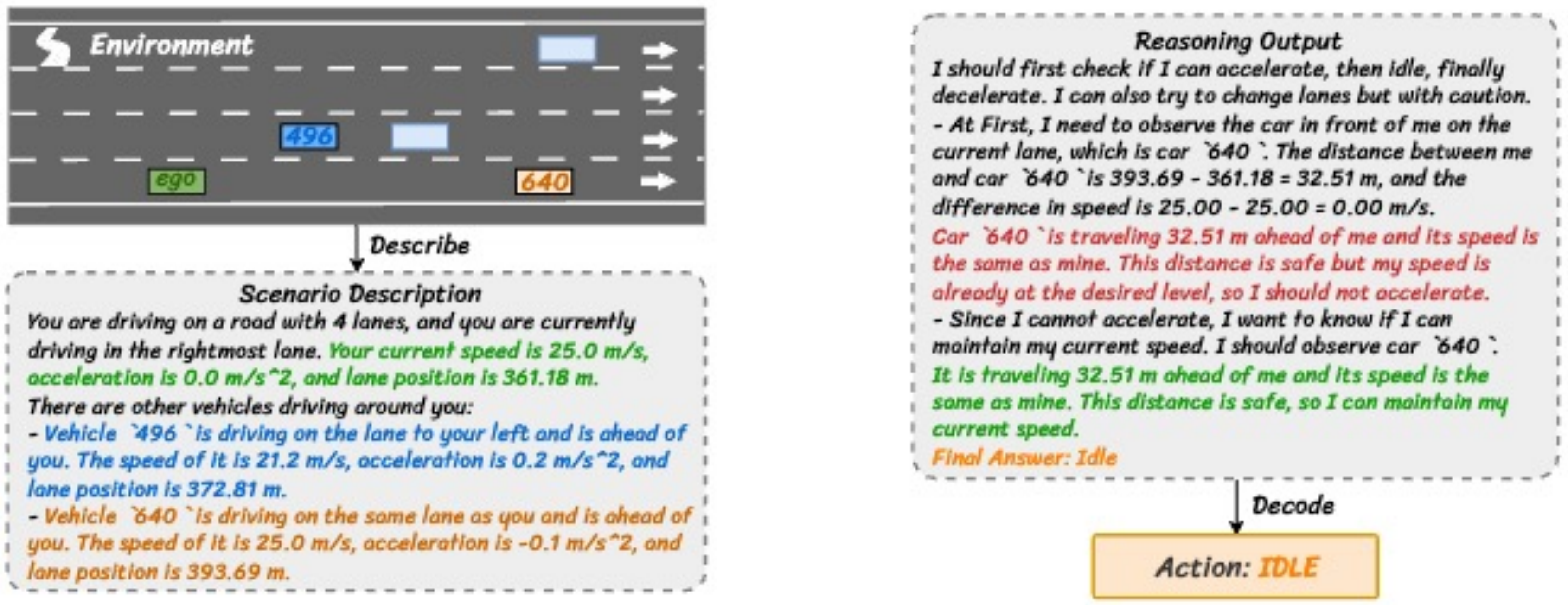


Figure 5: Reflection module. The Reflection module takes recorded decisions from closed-loop driving tasks as input, it utilizes a summarization and correction module to identify safe and unsafe decisions, subsequently revising them into correct decisions through the human knowledge embedded in LLM. Finally, these safe or revised decisions are updated into Memory module.



Scenario Description

You are driving on a road with 4 lanes, and you are currently driving in the rightmost lane. Your current speed is 25.0 m/s, acceleration is 0.0 m/s^2, and lane position is 361.18 m.

There are other vehicles driving around you:

- Vehicle `496 `is driving on the lane to your left and is ahead of you. The speed of it is 21.2 m/s, acceleration is 0.2 m/s^2, and lane position is 372.81 m.
- Vehicle `640 `is driving on the same lane as you and is ahead of you. The speed of it is 25.0 m/s, acceleration is -0.1 m/s^2, and lane position is 393.69 m.

Reasoning Output

I should first check if I can accelerate, then idle, finally decelerate. I can also try to change lanes but with caution.

- At First, I need to observe the car in front of me on the current lane, which is car `640 `. The distance between me and car `640 `is 393.69 - 361.18 = 32.51 m, and the difference in speed is 25.00 - 25.00 = 0.00 m/s.

Car `640 `is traveling 32.51 m ahead of me and its speed is the same as mine. This distance is safe but my speed is already at the desired level, so I should not accelerate.

- Since I cannot accelerate, I want to know if I can maintain my current speed. I should observe car `640 `.

It is traveling 32.51 m ahead of me and its speed is the same as mine. This distance is safe, so I can maintain my current speed.

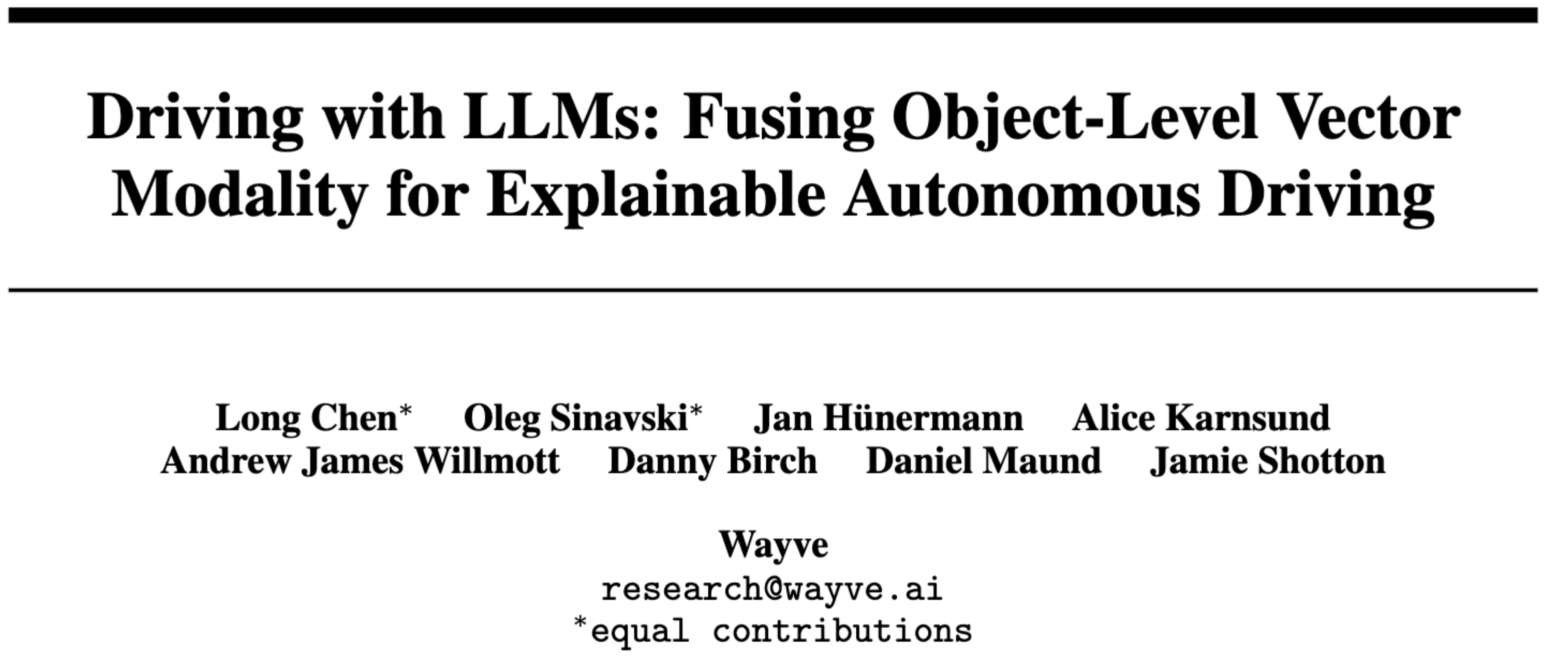
Final Answer: Idle

Action: IDLE

(a) (b)

Figure 4: (a) The textual scenario description generated by the scenario descriptor, (b) The decision decoder decodes the action with the output of LLM reasoning.

Visual Language Model



Driving with LLMs: Fusing Object-Level Vector Modality for Explainable Autonomous Driving

Long Chen*   Oleg Sinavski*   Jan Hünermann   Alice Karnsund
Andrew James Willmott   Danny Birch   Daniel Maund   Jamie Shotton

Wayve
`research@wayve.ai`
*equal contributions

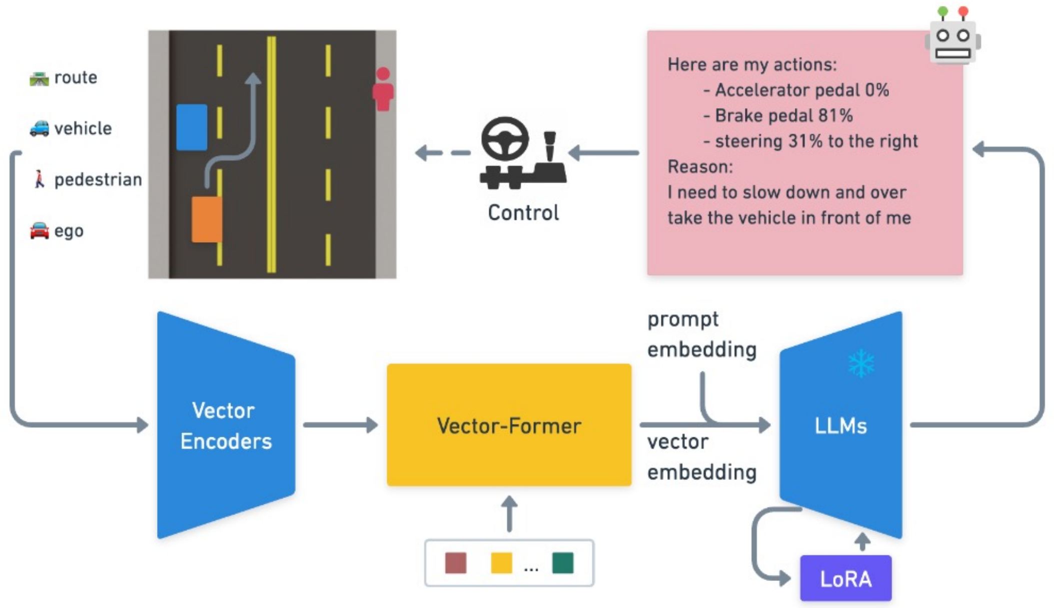


Figure 1: An overview of the architecture for Driving with LLMs, demonstrating how object-level vector input from our driving simulator is employed to predict actions via LLMs

In this paper, we have made the following contributions:

1. **Novel object-level multimodal LLM architecture:** We propose a novel architecture that fuses an object-level vectorized numeric modality into any LLMs with a two-stage pretraining and finetuning method.
2. **Driving scenario QA task and a dataset:** We provide a 160k question-answer pairs dataset on 10k driving situations with control commands, collected with RL expert driving agents and an expert LLM-based question answer generator. Additionally, we also outline the methodology for further data collection.
3. **Novel Driving QA (DQA) evaluation and a pretrained baseline:** We present a novel way to evaluate Driving QA performance using the same expert LLM grader. We provide initial evaluation results and a baseline using our end-to-end multimodal architecture.

What is your action and why?

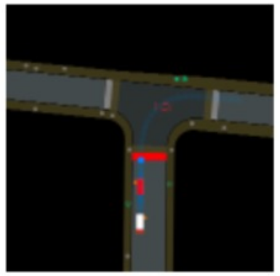
Here are my actions: - Accelerator pedal 84% - Brake pedal 0% - Steering 16% to the right. Reason: Because I need to maintain my current speed and direction.

What's the color of the traffic light?

There is no traffic light.

Can you press the accelerator now? why?

I'm unable to press the accelerator pedal now because the traffic light ahead is red.



What would you do if the car in front start moving, but the traffic light is still red?

I would wait for the green signal to start driving.

Transfer pseudo image to language

$\text{lanGen}(v_{\text{car}}, v_{\text{ped}}, v_{\text{ego}}, v_{\text{route}}, [o_{rl}]) \rightarrow$

“
A moving car; Angle in degrees: 1.19; Distance: 9.98m; [My attention: 78%]
A pedestrian; Angle in degrees: -41.90; Distance: 11.94m; [My attention: 22%]
My current speed is 11.96 mph.
There is a traffic light and it is red. It is 12.63m ahead.
The next turn is 58 degrees right in 14.51m.

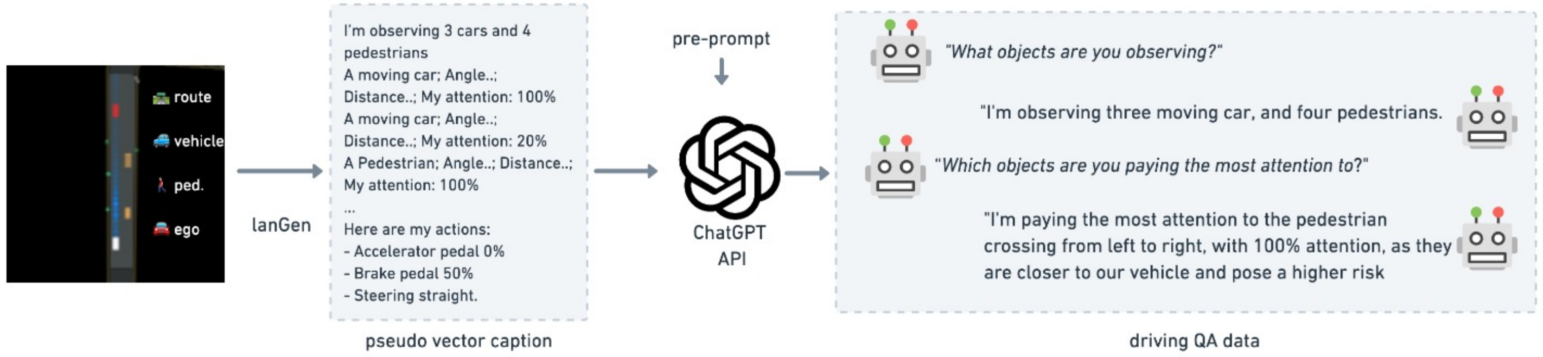
[Here are my actions:]
[- Accelerator pedal 0%]
[- Brake pedal 80%]
[- Steering straight]
”

In this function, variables v_{car}, v_{ped}, v_{ego}, and v_{route} denote vector information corresponding to cars, pedestrians, ego vehicle, and route, respectively. The optional term o_{rl} represents the output from the RL agent, consisting of additional attention and action labels for guiding the action reasoning process. Attention labels are collected from RL policy attention layers similar to [41].

This lanGen enables the transformation of vector representations into human-readable language captions. It crafts a comprehensive narrative of the current driving scenario, which includes of the agent’s observations, the agent’s current state, and its planned actions. This comprehensive contextual foundation enables the LLMs to conduct reasoning and construct appropriate responses in a manner that humans can interpret and understand.

3.3 Driving QA Dataset Labeling

we utilize GPT to generate a Driving QA dataset.



I'm observing 3 cars and 4 pedestrians
A moving car; Angle..; Distance..; My attention: 100%
A moving car; Angle..; Distance..; My attention: 20%
A Pedestrian; Angle..; Distance..; My attention: 100%
...
Here are my actions:
- Accelerator pedal 0%
- Brake pedal 50%
- Steering straight.

"What objects are you observing?"

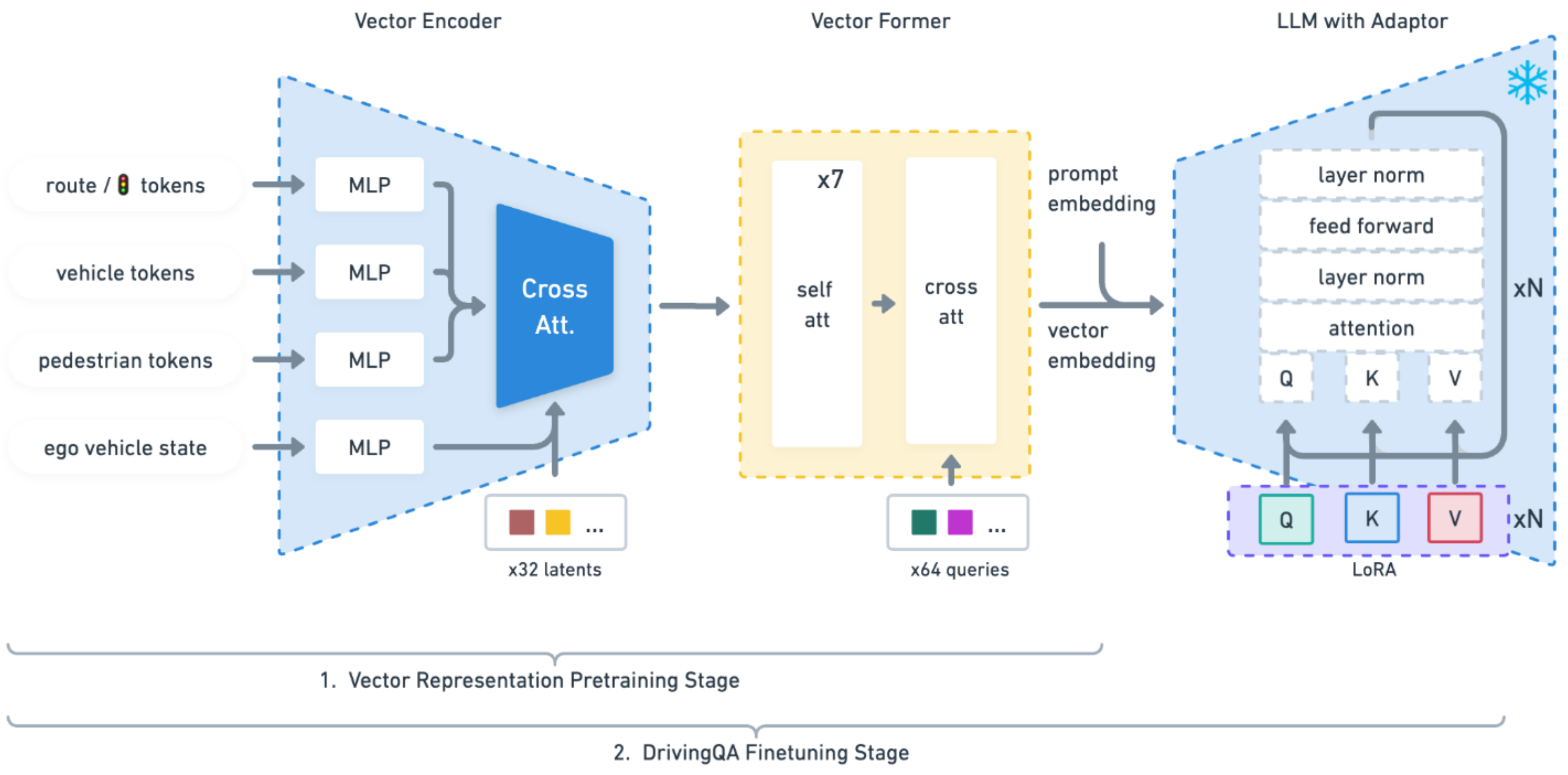
"I'm observing three moving car, and four pedestrians.

"Which objects are you paying the most attention to?"

"I'm paying the most attention to the pedestrian crossing from left to right, with 100% attention, as they are closer to our vehicle and pose a higher risk

Figure 2: The illustration of our Driving QA Dataset automatic labelling process

- **Question:** "What objects are you observing?" **Answer:** "I'm observing a moving car, and two pedestrians. The car is coming from the opposite direction, and the pedestrians are crossing the street from both the right and left sides."
- **Question:** "How are you going to drive in this situation and why?" **Answer:** "Here are my actions: - Accelerator pedal 0% - Brake pedal 7% - Steering straight. Reason: Because I'm maintaining a slow speed due to the pedestrians crossing, and I'm steering straight to keep our vehicle properly aligned with the road."
- **Question:** "What are the best tourist spots in London?" **Answer:** "As an AI Driver, the question you asked is out of my scope, but I can try to answer it. Some famous tourist spots in London include the Tower of London, Buckingham Palace, The British Museum, The Shard, and the London Eye."



- **Vector Encoder**: The four input vectors are passed through the Multilayer Perceptron (MLP) layers. They're then processed by a cross-attention layer to move them into a latent space. We add the ego feature to each learned input latent vector to emphasize the ego states.
- **Vector Former**: This part contains self-attention layers and a cross-attention layer that work with the latent space and question tokens. This transforms the latent vectors into an embedding that the LLM can decode.
- **LLM with Adaptor**: Here, we inject trainable rank decomposition matrices (LoRA) into the linear layers of the pretrained LLMs for parameter-efficient finetuning. We utilize LLaMA-7b [45] as the pretrained LLM for our experiments.

3.4.1 Vector Representation Pre-training

It is important to note that during this pretraining phase, we use only perception structured-language labels and avoid training on tasks that involve reasoning

3.4.2 Driving QA Finetuning

After the pre-training stage, we integrate the trainable LoRA module into the LLM, and optimize the weights of the Vector Encoder, Vector Former and LoRA module in an end-to-end fashion on the Driving QA data that we collected in Section 3.3.

LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS

We propose Low-Rank Adaptation, or LoRA, which freezes the pretrained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks

$$h = W_0x + \Delta W x = W_0 x + BAx$$

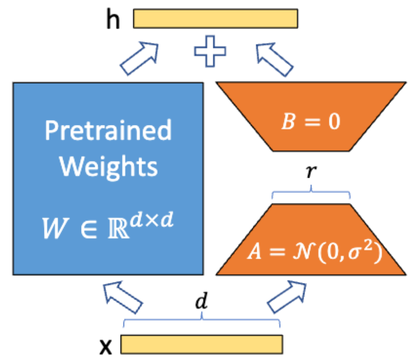


Figure 1: Our reparametrization. We only train A and B.