arXiv:2310.01957v2 [cs.RO] 13 Oct 2023

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

Abstract

Large Language Models (LLMs) have shown promise in the autonomous driving sector, particularly in generalization and interpretability. We introduce a unique object-level multimodal LLM architecture that merges vectorized numeric modalities with a pre-trained LLM to improve context understanding in driving situations. We also present a new dataset of 160k QA pairs derived from 10k driving scenarios, paired with high quality control commands collected with RL agent and question answer pairs generated by teacher LLM (GPT-3.5). A distinct pretraining strategy is devised to align numeric vector modalities with static LLM representations using vector captioning language data. We also introduce an evaluation metric for Driving QA and demonstrate our LLM-driver's proficiency in interpreting driving scenarios, answering questions, and decision-making. Our findings highlight the potential of LLM-based driving action generation in comparison to traditional behavioral cloning. We make our benchmark, datasets, and model available [1] for further exploration.

1 Introduction

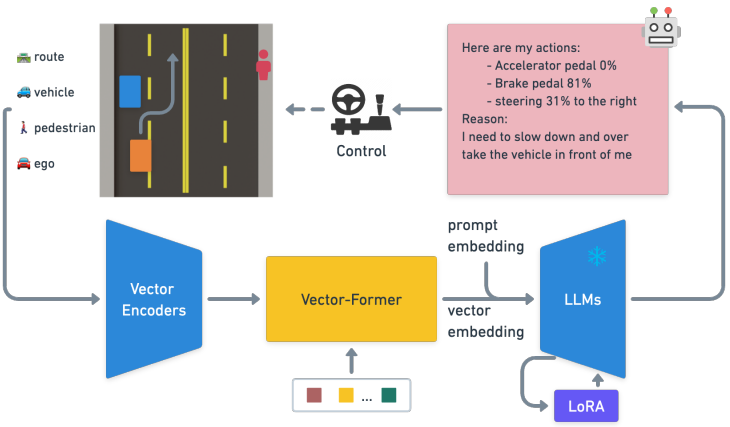

Figure 1: An overview of the architecture for Driving with LLMs, demonstrating how object-level vector input from our driving simulator is employed to predict actions via LLMs

[1] https://github.com/wayveai/Driving-with-LLMs

Preprint. Under review.

Remarkable abilities of Large Language Models (LLMs) demonstrate early signs of artificial general intelligence (AGI) [1], exhibiting capabilities such as out-of-distribution (OOD) reasoning, common sense understanding, knowledge retrieval, and the ability to naturally communicate these aspects with humans. These capabilities align well with the focus areas of autonomous driving and robotics [2] [3].

Modern scalable autonomous driving systems, whether they adopt an end-to-end approach using a single network [4], or a component-based configuration that combines learnable perception and motion planning modules [5] [6], face common challenges. These systems often behave as 'black-boxes' in the decision making process, making it especially difficult to endow them with OOD reasoning and interpretability capabilities. Such issues persist even though there have been some strides towards addressing them [7].

Textual or symbolic modality, with its inherent suitability for logical reasoning, knowledge retrieval, and human communication, serves as an excellent medium for harnessing the capabilities of LLMs [8]. However, its linear sequential nature limits nuanced spatial understanding [1], a crucial aspect of autonomous navigation. Pioneering work in Visual Language Models (VLMs) has begun to bridge this gap by merging visual and text modalities [9], enabling spatial reasoning with the power of pre-trained LLMs. However, effectively incorporating the new modality into the language representation space requires extensive pretraining with a significant volume of labeled image data.

We propose a novel methodology for integrating the numeric vector modality, a type of data that is frequently used in robotics for representing speed, actuator positions and distance measurements, into pre-trained LLMs. Such modality is considerably more compact than vision alleviating some of the VLM scaling challenges. Specifically, we fuse vectorized object-level 2D scene representation, commonly used in autonomous driving, into a pre-trained LLM with adapters [10]. This fusion enables the model to directly interpret and reason about comprehensive driving situations. As a result, the LLMs are empowered to serve as the "brain" of the autonomous driving system, interacting directly with the simulator to facilitate reasoning and action prediction.

To obtain training data in a scalable way, we first use a custom 2D simulator and train a reinforcement learning (RL) agent to solve the driving scenarios, serving as a substitute for a human driving expert. To ground the object-level vector into LLMs, we introduce a language generator that translates this numerical data into textual descriptions for representation pretraining. We further leverage a teacher LLM (GPT) to generate a question-answering dataset conditioned on the language descriptions of 10k different driving scenarios. Our model first undergoes a pretraining phase that enhances the alignment between the numeric vector modality and the latent language representations. Next, we train our novel architecture to establish a robust baseline model, LLM-driver, for the driving action prediction and driving question answering tasks. We provide our datasets, evaluation benchmarks and a pre-trained model[1] for reproducibility and hope to inspire and facilitate further advancements in the field. The subsequent sections of this paper detail the theoretical background, our proposed architecture and experimental setup, preliminary results, potential directions for future research, and implications of our work for the broader field of autonomous driving.

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.

Our work provides the first-of-its-kind baseline approach for integrating LLMs into driving task in simulation. This includes a comprehensive framework encompassing the simulator, automatic data collection, integration of a new object-level vector modality into LLMs, and the GPT-based evaluations approaches.

2 Related Works

2.1 End-to-End Autonomous Driving Systems

There was a significant progress in end-to-end deep learning methods for autonomous systems in recent years [11], [12], [4], with some of the earliest efforts dating back to ALVINN [13] and more recent works such as [14]. However, a fundamental challenge that remains with modern autonomous driving systems is the lack of interpretability in the decision making process [15]. Understanding why a decision is made is crucial for identifying areas of uncertainty, building trust, enabling effective human-AI collaboration and ensuring safety [16]. We continue this line of research by adding vector/textual modality and pretrained LLMs to the end-to-end autonomous driving.

2.2 Interpretability of Autonomous Driving Systems

A variety of explainability methods have been introduced [17] to understand the underlying decision process of deep neural networks. For example [18], [19] and [20] are well-established model-agnostic interpretability methods that generate explanations for individual predictions. Other methods such as gradient based [21], saliency maps [22] and attention maps [23] target the inner operations of models to explain the decision making process. In the field of autonomous vehicles, visual attention maps, which highlight causally influential regions in driving images were proposed in [24]. In [25] the authors combined attention based methods with natural language to create an attention-based vehicle controller that provides natural language action descriptions and explanations based on a series of image frames. This work was further extended in [26], where the authors improved the architecture by integrating part of speech prediction and special token penalties. Others argue that attention is not enough [27], leading to multiple efforts to combine this approach with other explanatory methods. For example [28] proposes to explain transformers by leveraging attentive class activation tokens, encoded features, their gradients, and their attention weights simultaneously. Building on this research, we are proposing to use text modality for explainability in autonomous driving.

2.3 Multi-modal LLMs in Driving Tasks

Recently, there has been a notable trend towards integrating multiple modalities into unified large-scale models. Notable examples include VLMs such as [29], [30], [31], and [32], which effectively combine language and images to accomplish tasks like image captioning, visual question answering, and image-text similarity. Another noteworthy advancement [33] involves the fusion of information from six distinct modalities: text, image/video, audio, depth, thermal, and inertial measurements. This exciting development not only expands the possibilities for generating content using diverse data input and output types but also enables broader multi-modal search capabilities.

With camera sensors being one of the most common sensors used in autonomous driving [34], a natural step to incorporate language has been through VLMs. For example [35] uses images and language directions to train a driving policy. [36] proposes a method for learning vehicle control with human assistance. The system learns to summarize its visual observations in natural language, predict an appropriate action response (e.g. "I see a pedestrian crossing, so I stop"), and predict the controls, accordingly. Using language to explain the inner workings of the model has also been explored in [37], where user-friendly natural language narrations and reasoning are provided for each decision making step of autonomous vehicular control and action.

In robotics, we have seen efforts fusing language with other modalities. Albeit outside of autonomous driving field, the closest work to ours [38] utilizes point clouds with 3D bounding boxes of potential object candidates. It also uses a language utterance referring to a target object in the scene to train a model capable of identifying a target object from a set of potential candidates. Recently, the RT-2 paper [39] demonstrated a similar approach by utilizing LLMs for low-level robotics control tasks, including the joint training of VQA and control tasks. However, their framework is confined to the vision modality, whereas we introduce a novel methodology for grounding vector-based object level modalities into LLMs, facilitating interpretable control and driving QA tasks. In contrast to these existing efforts, the work presented in this paper is, to the best of our knowledge, the first to fuse numeric vector modality with language specifically in the domain of autonomous vehicles.

3 Method

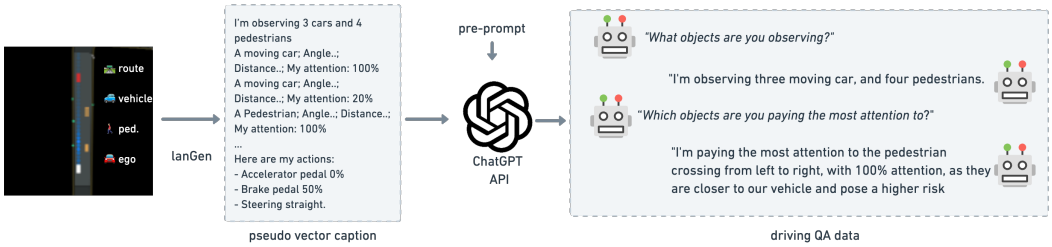

Figure 2: The illustration of our Driving QA Dataset automatic labelling process

3.1 Data Collection using RL experts

To generate language-based grounded driving datasets, we use a custom-built realistic 2D simulator with procedural generation of driving scenarios. We use an RL agent that solves the simulated scenarios using an object-level ground-truth representation of the driving scene. In our approach, we map a vector representation of the environment to an action for the vehicle dynamics with an attention-based neural network architecture. This model is optimized with Proximal Policy Optimization (PPO) [40]. Subsequently, we collect continuous driving data from 15 diverse virtual environments with randomly generated traffic conditions. Our collection includes a 100k dataset for pretraining, a 10k set for QA labeling and fine-tuning, and a 1k set dedicated to evaluation.

3.2 Structured Language Generation for Pseudo Vector Captioning

In our framework, we aim to convert vector representations into language using a structured language generator to facilitate the grounding the vector representation into LLMs. Since our object-level vectors contain semantically significant attributes, such as the number of cars and pedestrians, their respective locations, orientations, speeds, bounding boxes and other attributes, we employ a structured language generator (lanGen) function to craft pseudo-language labels derived from the vector space, as illustrated below:

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

"
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.

The inclusion of o_{rl} is optional, and we generate two different versions of pseudo labels to cater to different requirements: 1) **Without Attention/Action:** Employed during the representation pre-training stage (see Subsection 3.4.1), where the inference of attentions and actions is not required. 2) **With Attention/Action:** Utilized for VQA labeling with GPT during the fine-tuning stage (see Subsection 3.4.2). This equips GPT with the ability to ask specific questions about attentions and actions, thereby empowering the driving LLM agent with the ability to reason about attentions and actions.

3.3 Driving QA Dataset Labeling

Large amount of data is the key to enabling the question answering ability of the language models. This becomes particularly crucial when a new modality is introduced to the LLMs; in such circumstances, it is essential to have a high-quality question-answering dataset relevant to the modality input. Studies have demonstrated that data labelled using ChatGPT surpasses crowdsourced workers in terms of performance for text-annotation tasks [42]. Inspired by Self-Instruct [43] and LLaVA [44], we utilize GPT to generate a Driving QA dataset. This dataset is conditioned on the structured language input, serving as a representative for the vectors.

To be specific, we tasked GPT to emulate the role of a certified professional driving instructor to generate questions and answers for students. We supplied basic driving rules as pre-prompts, covering aspects like traffic signal transitions and the driving side as we are operating in a left-hand driving country. We tasked GPT with creating 16 sets of random questions and answers per scenario, taking into account the current driving scenario provided by language description generated by the lanGen. We also included some instructions to guide the generation of action-related questions, so that they can be easily parsed with regular expressions for evaluating action prediction tasks.

To reduce hallucinations, we established simple guardrail rules. We instructed GPT to include a driving-related question that could not be inferred from the observation input. If faced with such a question, the model's response should be akin to "I'm unable to answer this question based on the observations I have," preventing the model from imagining objects not present in the provided vector observation.

Additionally, we included a random question unrelated to driving, instructing the model to respond with "As an AI Driver, the question you asked is out of my scope, but I can attempt to answer it." The model would then proceed to answer the question normally, thereby preserving the language model's versatility as a general-purpose assistant.

Detailed instructions can be found in our open-sourced code[1]. Below are some examples of the question-answer pairs generated by GPT:

- **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."

3.4 Training the Driving LLM Agent

Training the LLM-Driver involves formulating it as a Driving Question Answering (DQA) problem within the context of a language model. The key to this formulation is the integration of an object-level vector modality into the pre-trained LLMs, creating a multi-modal system capable of interpreting and interacting with both language and vector inputs.

We use a two-stage process to train our model for effectively fusing object-level vector modality into the LLM-Driver. In the first stage, we ground the vector representation into an embedding that can

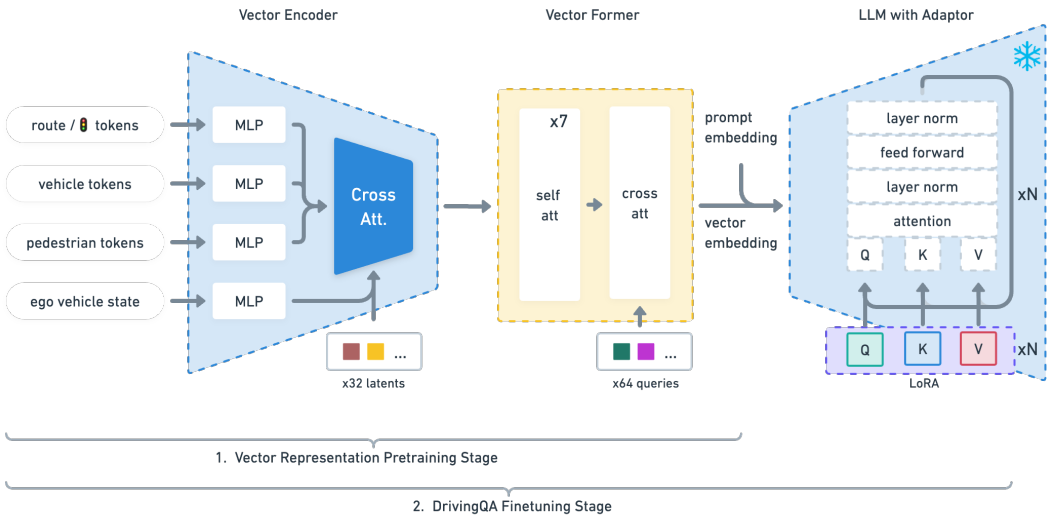

Figure 3: The architecture of the Driving LLM Agent

be decoded by the LLMs. This is accomplished by freezing the language model and optimizing the weights of the vector encoders and the vector transformer. In the second stage, we finetune the model to the DQA task, training it to answer driving-related questions and take appropriate actions based on its current understanding of the environment.

As can be seen in the Figure 3, our model is built on three key components: the Vector Encoder, Vector Former, and a frozen LLM with a Low-Rank Adaptation (LoRA) [10] module.

- **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

Integrating a new modality into pre-trained Large Language Models (LLMs) poses significant challenges due to the need for extensive data and computational resources. In this study, we propose a novel approach that leverages structured language to bridge the gap between the vector space and language embeddings, particularly focusing on numerical tokens.

During the pretraining phase, we freeze the language model while training the entire framework end-to-end to optimize the weights of the vector encoders and the vector transformer (V-former). Such an optimization process enables effective grounding of the vector representation into an embedding that can be directly decoded by the LLMs. 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, such as action prediction (vehicle control commands) and agent attention prediction (where the expert focuses spatial attention). This is because our focus at this stage is solely on representation training, and we aim to avoid prematurely integrating any reasoning components into the V-former.

The pretraining process was conducted using 100k question-answer pairs, which were collected from a simulator. Additionally, in each epoch, we sampled 200k vectors with uniformly-distributed random values, contributed to robust representation learning by comprehensively exploring the vector space and its associated semantic meanings. We employed the lanGen method, as described in Sections 3.2,

to automatically label the pseudo vector-captioning data. During the pretraining phase, we penalized errors in vector captioning results to optimize the vector encoder and V-former weights, thereby transforming the vector space into language embeddings understandable by LLM.

Through this approach, we are able to effectively incorporate object-level vector modality into pre-trained LLMs, which serves as a good starting point for the finetuning stage.

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.

In order to train the LLM-Driver to output accurate actions for driving, we added certain action-triggering questions to the VQA dataset. These are questions that, when asked, require the agent to generate actions in specific format. For example, a question like "How are you going to drive in this case and why?" would require the agent to infer the action based on the vector input. We then use a simple language-action grounding strategy with regular expression matching to extract the action required to control the car in our simulator. To ensure the model pays sufficient attention to these important questions, we upsample the action-triggering questions with different expressions such as:

- "How are you going to drive in this situation?"
- "What actions are you taking?"
- "How are you driving in this situation?"
- "What are your planned actions for this situation?"

We refine the pre-existing model on 10,000 scenarios, each incorporating 16 driving QA entities, along with an additional vector for captioning data and an action prediction QA pair. The outcome is a driving LLM agent that is capable of reasoning about its actions and responding to any driving-related questions given the object-level vectors of the current driving scenario.

4 Evaluation and Results

This section aims to thoroughly assess our model's performance using a separate evaluation set comprising 1,000 distinct driving scenarios. The vector captioning and vector QA data for these scenarios were labeled following the same methodologies discussed in Sections 3.2 and 3.3.

4.1 Evaluation of Perception and Action Prediction

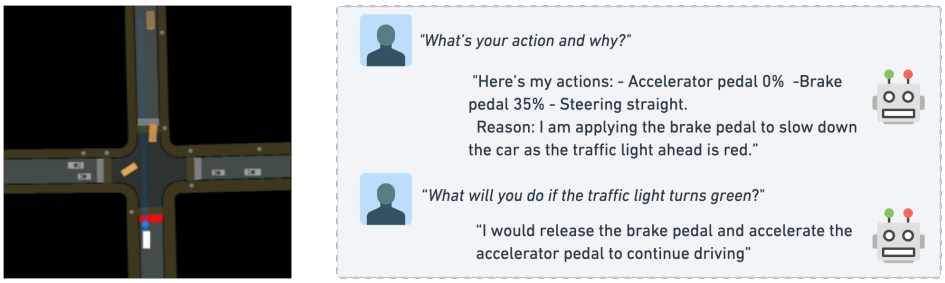

Figure 4: Demonstration of the interaction with the pretrained LLM-Driver. Notably, the driving LLM agent has the ability to anticipate future actions when prompted. Please refer to Appendix B for additional results, including some failure cases.

To obtain the model's perception and action predictions, we prompt the model with the same vector captioning and add the action-predicting triggering question from the driving QA data. We then parse the language output using regular expressions to extract the numerical predictions.

We reported the results on the model trained using a two stage approach (with pretraining), and the model training only on driving QA dataset (without pretraining). We also include a baseline

model, Perceiver-BC, which replaces the LLM with a non-pretrained Perceiver model for comparison purposes. This aims to evaluate how effectively pretrained LLMs can be utilized for reasoning in driving-related tasks, such as action prediction. The Perceiver-BC model includes the identical Vector Encoder and V-former modules as those in the LLM agent model. However, it differs in the fact that it employs a transformer-based policy module in place of the LLM with adapters (please refer to Appendix A for more details). The Perceiver-BC model also outputs actions with perception auxiliary tasks of agent and traffic light detection. To maintain an equitable comparison, we've calibrated the BC model to have a similar number of trainable parameters as in the LLM agent, totaling to approximately 25 million trainable parameters. The Perceiver-BC model was trained on perception and action prediction tasks using the same 10k dataset that we used for the LLM Agent, but without VQA data.

For the reported metrics, we calculate the Mean Absolute Error (MAE) for the predictions of the number of cars and pedestrians, denoted as E_{car} and E_{ped} respectively. Additionally, we measure the accuracy of traffic light detection as well as the mean absolute distance error in meters for traffic light distance prediction, represented as Acc_{TL} and D_{TL}. Furthermore, we compute the MAE for normalized acceleration and brake pressure denoted as $E_{lon.}$, and normalized steering wheel angle denoted as $E_{lat.}$. Lastly, we report the weighted cross-entropy loss for the token prediction on the evaluation set, indicated as L_{token}.

Table 1: The evaluation result of perception and action prediction

	agents count		*traffic light*		*action*		*loss*
	$E_{car} \downarrow$	$E_{ped} \downarrow$	$Acc_{TL} \uparrow$	$D_{TL} \downarrow$	$E_{lon.} \downarrow$	$E_{lat.} \downarrow$	$L_{token} \downarrow$
Perceiver-BC[46]	0.869	0.684	**0.900**	**0.410**	0.180	0.111	n/a
LLM-Driver$_{\text{w/o pretrain}}$	0.101	1.668	0.758	7.475	0.094	0.014[a]	0.644
LLM-Driver$_{\text{w/ pretrain}}$	**0.066**	**0.313**	0.718	6.624	**0.066**	**0.014**[b]	**0.502**

[a] Exact value: 0.01441
[b] Exact value: 0.01437

As can be seen in Table 1, the results clearly demonstrate that the pretraining stage significantly enhances both the model's perception and action prediction capabilities. This suggests that the pretrained model exhibits a higher level of accuracy in perceiving and quantifying the number of cars and pedestrians in its environment. The pretrained model also shows a lower loss value, L_{token}, which indicates an improvement in the overall effectiveness of the model's token predictions.

Note that we filter out agents that fall outside the 30m range from the ego vehicle, which needs to be calculated using the x, y, z vector. This setting makes the "direct decoding" of agent detection from the vector much more difficult. For simpler regression tasks (e.g., traffic light distance), Perceiver-BC performs much better than LLMs.

For the action prediction task requiring in-depth reasoning, we found that LLM-based policies outperform the Perceiver-BC approach when given the same amount of training data and trainable parameters. This indicates that LLMs serve as effective action predictors, harnessing knowledge acquired during the general pretraining phase**—such as stopping at a red light or decelerating when vehicles or pedestrians are ahead—**to inform decisions based on their grounded observations.

However, it's important to note the distinction in training methodologies: Perceiver-BC is trained mostly using regression on perception and control outputs, while LLMs are trained via cross-entropy token loss and benefit from an extra 16x pairs of driving questions and answers, which will reinforces the learning of perception and action prediction. Thus, the comparison might not be entirely equitable and should be taken as merely a point of reference.

4.2 Evaluation of Driving QA

To assess the quality of answers to open-ended questions about the driving environment, we use GPT-3.5 to grade our model's responses. This is a recently emerged technique for grading natural

language answers [47] [48] [49]. This approach allows us to quickly and consistently evaluate our model's capabilities for questions that don't have fixed answers.

For evaluation, we prompt GPT-3.5 with the language-based observation description used during dataset generation (section 3.3), the question from the test set, and the model's answer. GPT's task is to write a one-line assessment, followed by a score ranging from 0 to 10 (where 0 is worst and 10 is best) for each response, based on the given observation details. The final score of the model is the average of all question scores. We note that GPT evaluations can sometimes be overly lenient with answers that are incorrect but semantically close, as well as other issues with GPT-based grading as reported by [49]. To validate these findings, we hand-scored 230 randomly sampled QA pairs and obtained comparable results.

Table 2: Grading of the Driving QA outputs

	GPT Grading ↑	Human Grading ↑
Constant answer "I don't know"	2.92	0.0
Randomly shuffled answers	3.88	0.26
LLM-Driver$_{\text{w/o pretrain}}$	7.48	6.63
LLM-Driver$_{\text{w pretrain}}$	**8.39**	**7.71**
Answers generated by GPT	9.47	9.24

Our results in Table 2 demonstrate that pre-training improves the grading score of the model by 9.1% and 10.8% over those models without pre-training, according to both GPT grading and human grading. For reference, we also provide the scores obtained when running our evaluation procedure with artificially incorrect answers (by either constantly answering "I don't know" or by randomly shuffling all answers), as well as the "ground-truth" answer labels provided by GPT. From these results, we can see that our model, which utilizes only vectorized representations as input, can achieve a much higher score than either a constant or random answer approach.

5 Conclusion and Limitations

While our approach exhibits considerable novelty and potential, the preliminary results indicate that there is still progress to be made for the LLM to fully navigate in simulation. We are aware of possible discrepancies in open-loop vs closed-loop results [50], and our future work will focus on addressing the challenge of efficiently evaluating the model in a closed-loop system. This includes addressing the lengthy inference time of LLMs and the extensive steps needed to thoroughly test the system. Moreover, we anticipate the need to improve the precision of the direct driving commands produced by our baseline in order to operate effectively in closed-loop settings. Factors contributing to this include the intricacy of the task, potential enhancements to the model architecture, and the need for improving the scale and quality of our training dataset. We have observed that numeric inaccuracies and the early developmental stage of our system can lead to discrepancies in explanations and reasoning, preventing our ideas from being fully implemented in closed-loop settings. These observations will guide our future research as we continue to refine our approach.

Nevertheless, our work establishes a foundation for future research in this direction. We believe that our proposed architecture, combined with the novel way for grounding a new modality into LLMs, a data auto-labelling pipeline, and an LLM-based evaluation pipeline, can serve as a starting point for researchers interested in exploring the integration of numeric vector modality with LLMs in the context of autonomous driving.

In terms of future research, improving the architecture to better handle nuances in the numeric vector modality could offer a promising direction. Our approach holds potential for application beyond simulated environments. Given sufficient real-world perception labels for pretraining and fine-tuning a VLM, our methodology could also be adapted to real-world driving scenarios.

Overall, while our results are preliminary we believe our work is a significant step forward in the integration of vector modality with LLMs in the context of autonomous driving.

6 Ethical Implications and Broader Impact

By introducing LLMs into autonomous driving we are inheriting their ethical implications [51]. We believe that these problems will be addressed by further research from the LLM community. As for autonomous driving, it's crucial to ensure that the system can handle all possible driving scenarios safely. By improving the interpretability of autonomous driving systems, we can help build trust in this technology, thereby accelerating its adoption and ultimately leading to safer and more efficient transportation systems.

References

[1] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with gpt-4," 2023.

[2] L. Wells and T. Bednarz, "Explainable ai and reinforcement learning—a systematic review of current approaches and trends," *Frontiers in artificial intelligence*, vol. 4, p. 550030, 2021.

[3] K. Lu, S. Zhang, P. Stone, and X. Chen, "Robot representation and reasoning with knowledge from reinforcement learning," 2018.

[4] J. Hawke, V. Badrinarayanan, A. Kendall, *et al.*, "Reimagining an autonomous vehicle," *arXiv preprint arXiv:2108.05805*, 2021.

[5] L. Chen, L. Platinsky, S. Speichert, B. Osiński, O. Scheel, Y. Ye, H. Grimmett, L. Del Pero, and P. Ondruska, "What data do we need for training an av motion planner?" in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1066–1072.

[6] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21918

[7] D. Omeiza, H. Webb, M. Jirotka, and L. Kunze, "Explanations in autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 142–10 162, aug 2022. [Online]. Available: https://doi.org/10.1109%2Ftits.2021.3122865

[8] N. F. Rajani, B. McCann, C. Xiong, and R. Socher, "Explain yourself! leveraging language models for commonsense reasoning," 2019.

[9] X. Wang, G. Chen, G. Qian, P. Gao, X.-Y. Wei, Y. Wang, Y. Tian, and W. Gao, "Large-scale multi-modal pre-trained models: A comprehensive survey," 2023.

[10] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[11] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[12] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[13] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[14] A. Hu, G. Corrado, N. Griffiths, Z. Murez, C. Gurau, H. Yeo, A. Kendall, R. Cipolla, and J. Shotton, "Model-based imitation learning for urban driving," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 20 703–20 716. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/827cb489449ea216e4a257c47e407d18-Paper-Conference.pdf

[15] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519308103

[16] W. Xu, "From automation to autonomy and autonomous vehicles: Challenges and opportunities for human-computer interaction," *Interactions*, vol. 28, no. 1, p. 48–53, dec 2020. [Online]. Available: https://doi.org/10.1145/3434580

[17] A. Holzinger, A. Saranti, C. Molnar, P. Biecek, and W. Samek, *Explainable AI Methods - A Brief Overview*. Cham: Springer International Publishing, 2022, pp. 13–38. [Online]. Available: https://doi.org/10.1007/978-3-031-04083-2_2

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[19] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[20] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International conference on machine learning*. PMLR, 2017, pp. 3145–3153.

[21] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[22] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[23] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*. PMLR, 2015, pp. 2048–2057.

[24] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," 2017.

[25] J. Kim, A. Rohrbach, T. Darrell, J. Canny, and Z. Akata, "Textual explanations for self-driving vehicles," 2018.

[26] M. A. Kühn, D. Omeiza, and L. Kunze, "Textual explanations for automated commentary driving," *arXiv preprint arXiv:2304.08178*, 2023.

[27] S. Jain and B. C. Wallace, "Attention is not explanation," *arXiv preprint arXiv:1902.10186*, 2019.

[28] Y. Qiang, D. Pan, C. Li, X. Li, R. Jang, and D. Zhu, "Attcat: Explaining transformers via attentive class activation tokens," in *Advances in Neural Information Processing Systems*, 2022.

[29] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021.

[30] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, "Flamingo: a visual language model for few-shot learning," 2022.

[31] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," 2023.

[32] OpenAI, "Gpt-4 technical report," 2023.

[33] R. Girdhar, A. El-Nouby, Z. Liu, M. Singh, K. V. Alwala, A. Joulin, and I. Misra, "Imagebind: One embedding space to bind them all," 2023.

[34] M. Hasanujjaman, M. Z. Chowdhury, and Y. M. Jang, "Sensor fusion in autonomous vehicle with traffic surveillance camera system: Detection, localization, and ai networking," *Sensors*, vol. 23, no. 6, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/6/3335

[35] J. Roh, C. Paxton, A. Pronobis, A. Farhadi, and D. Fox, "Conditional driving from natural language instructions," in *Conference on Robot Learning*. PMLR, 2020, pp. 540–551.

[36] J. Kim, S. Moon, A. Rohrbach, T. Darrell, and J. Canny, "Advisable learning for self-driving vehicles by internalizing observation-to-action rules," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[37] B. Jin, X. Liu, Y. Zheng, P. Li, H. Zhao, T. Zhang, Y. Zheng, G. Zhou, and J. Liu, "Adapt: Action-aware driving caption transformer," 2023.

[38] J. Roh, K. Desingh, A. Farhadi, and D. Fox, "Languagerefer: Spatial-language model for 3d visual grounding," in *Conference on Robot Learning*. PMLR, 2022, pp. 1046–1056.

[39] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *arXiv preprint arXiv:2307.15818*, 2023.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[41] K. Renz, K. Chitta, O.-B. Mercea, A. S. Koepke, Z. Akata, and A. Geiger, "Plant: Explainable planning transformers via object-level representations," 2022.

[42] F. Gilardi, M. Alizadeh, and M. Kubli, "Chatgpt outperforms crowd-workers for text-annotation tasks," 2023.

[43] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi, "Self-instruct: Aligning language model with self generated instructions," 2022.

[44] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," 2023.

[45] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[46] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[47] J. Fu, S.-K. Ng, Z. Jiang, and P. Liu, "Gptscore: Evaluate as you desire," 2023.

[48] J. Wang, Y. Liang, F. Meng, Z. Sun, H. Shi, Z. Li, J. Xu, J. Qu, and J. Zhou, "Is chatgpt a good nlg evaluator? a preliminary study," 2023.

[49] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: Nlg evaluation using gpt-4 with better human alignment," 2023.

[50] F. Codevilla, A. M. López, V. Koltun, and A. Dosovitskiy, "On offline evaluation of vision-based driving models," 2018.

[51] T. Y. Zhuo, Y. Huang, C. Chen, and Z. Xing, "Exploring ai ethics of chatgpt: A diagnostic analysis," 2023.

[52] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 7153–7162.

Appendix A Perceiver-BC model

In order to assess the proficiency of leveraging the reasoning ability of pretrained Large Language Models (LLMs) in driving tasks, such as action prediction, we decided to conduct additional experiments using a simple Perceiver-BC model for comparison purposes. We've detailed the implementation process in Figure 5.

The Perceiver-BC model includes the identical Vector Encoder and Vector Former modules as those in the LLM agent model. However, it differs in that it employs a perceiver-based policy module instead of the LLM with adapters. To maintain an equitable comparison, we've calibrated the BC model to possess a similar number of trainable parameters to the LLM agent, totaling approximately 25 million trainable parameters.

The Perceiver-BC model was trained using the same 10k dataset that we used for the LLM Agent. This dataset included only captioning and action data. We trained the Perceiver-BC model for 5 epochs, just like the LLM Agent. We have reported the results of these evaluations in Table 1.

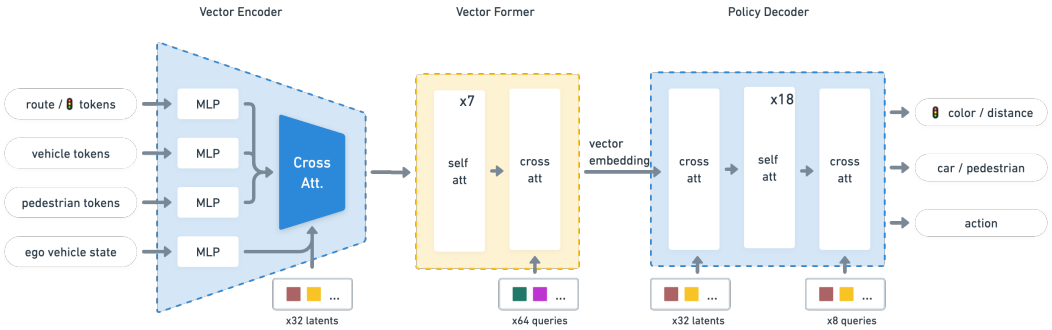

Figure 5: The architecture of the behaviour cloning baseline

Appendix B Additional Results

We include here more results DrivingQA results in Figure 6 and Figure 7, including some failure cases.

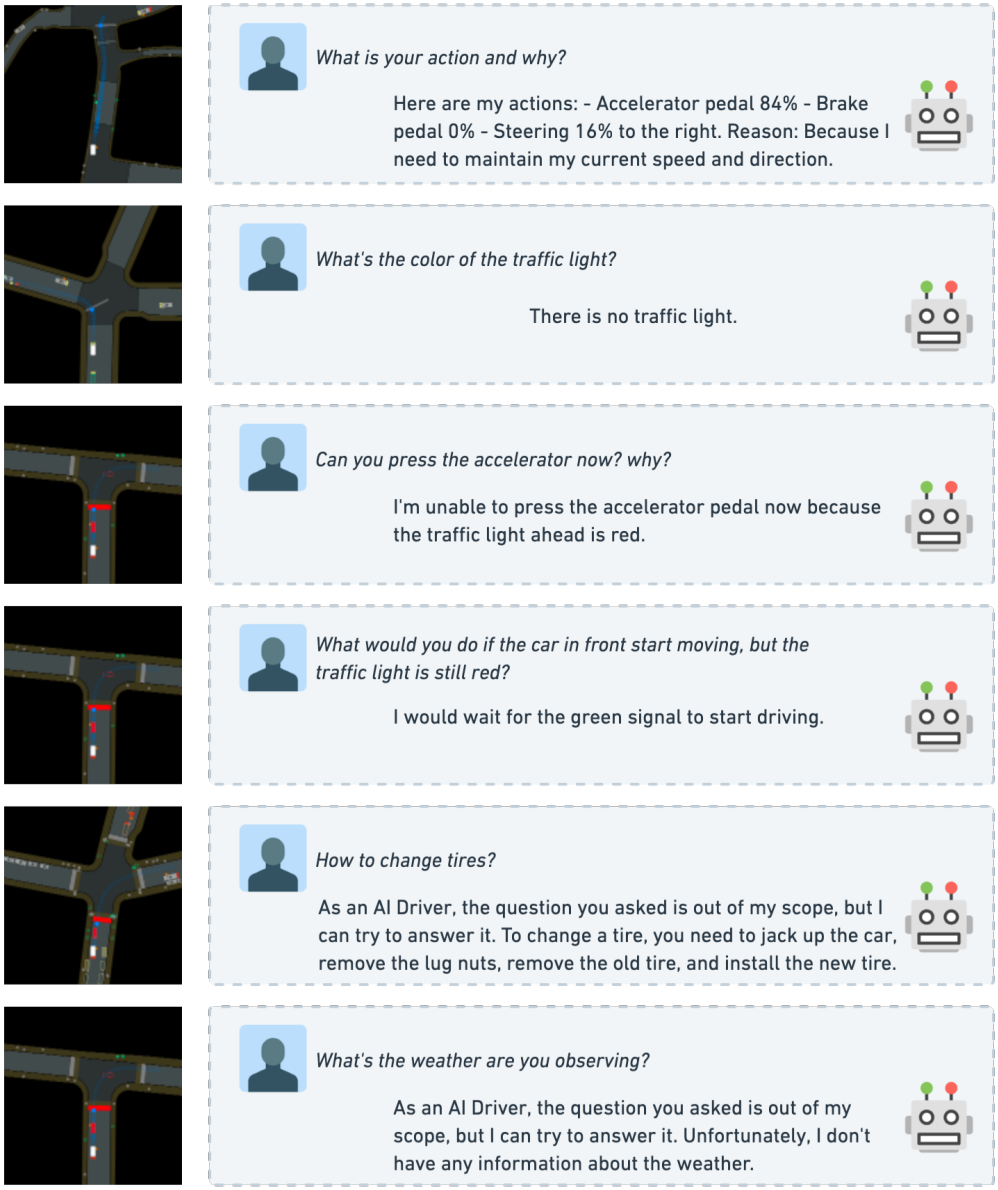

Figure 6: More drivingQA results

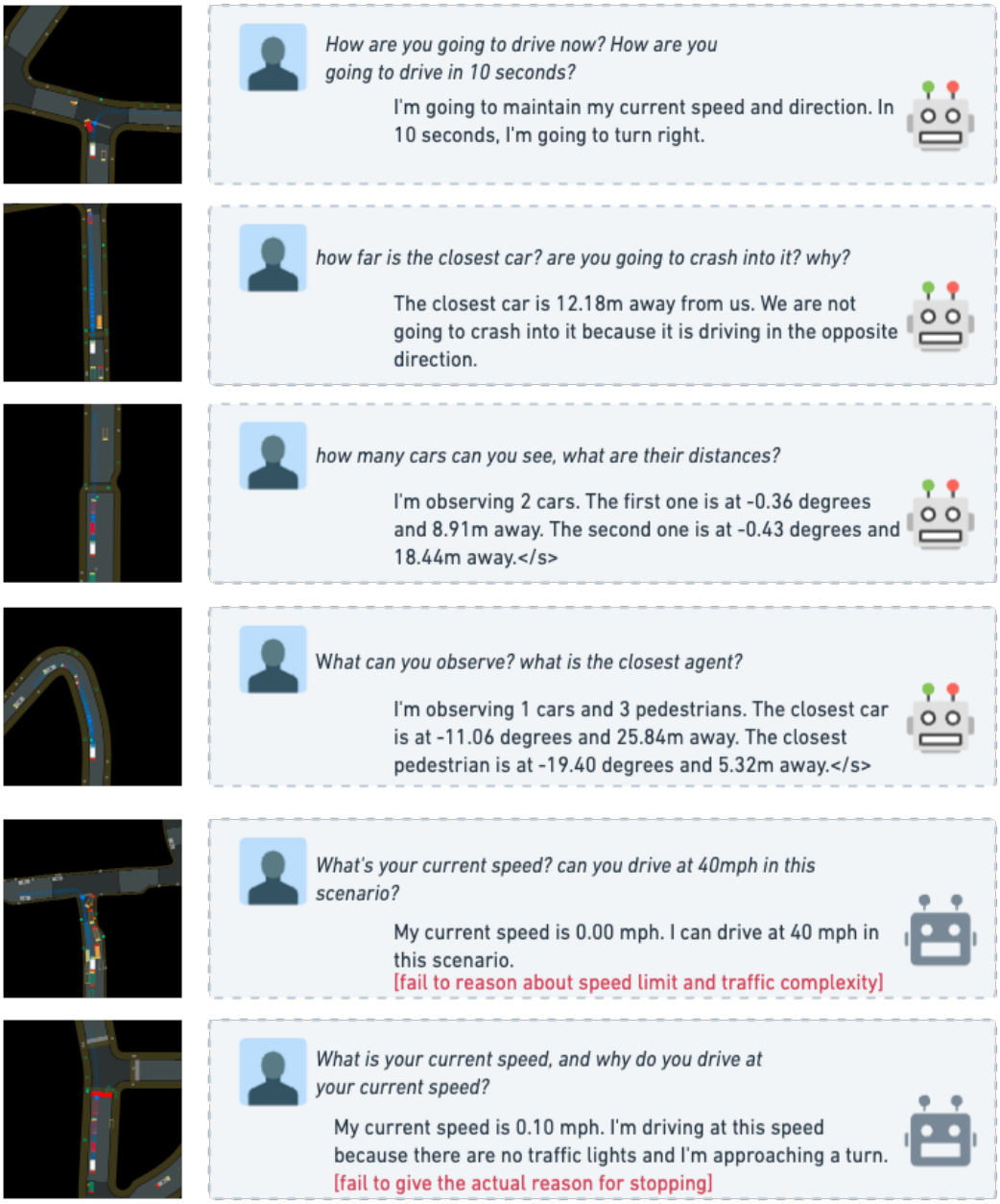

Figure 7: More drivingQA results and failure cases (last two)

B.1 Design of the Object-level Vector Representation

In order to effectively translate the intricacies of autonomous driving into a format that a Large Language Model (LLM) can understand and work with, we have designed an object-level vector representation. This allows us to represent a variety of crucial aspects of driving conditions, such as the intended route, nearby vehicles, pedestrian presence, and the state of the ego vehicle, using a set of compact object-level vectors.

Our object-level vectors contain semantically significant attributes. This allows us to further create structured language labels from the vector representation. This process turns complex driving situations into a language format that the Large Language Model (LLM) can understand. This enables the LLM to make sense of driving conditions, make decisions, and even generate outputs that can be explained, based on its own internal reasoning.

The schema below provides more details about each aspect represented in our vector input system:

```
# A 2d array per ego vehicle describing the route to follow.
# It finds route points for each vehicle. Each point goes into a new row,
# then each point have the following attributes:
# - (x, y, z) of a point
# - its direction
# - pitch
# - speed limit
# - is junction?
# - width of the road
# - is traffic light and its state
# - is give way?
# - is a roundabout?
route_descriptors: torch.FloatTensor

# A 2d array per ego vehicle describing nearby vehicles.
# First, if finds nearby vehicle in the neighbourhood of the car.
# Then allocates an array of zeros a fixed max size (about 30).
# There is a logic that tries to fit dynamic and static vehicles into
# rows of that array.
# For every vehicle:
# - "1" for marking that a vehicle is found in the row
# (empty rows will have this still "0")
# - Is it dynamic or static (parked) vehicle
# - its speed
# - its relative position in the ego coordinates
# - its relative orientation
# - its pitch
# - its size
# - vehicle class
# - positions of its 4 corners
vehicle_descriptors: torch.FloatTensor

# A 2d array per ego vehicle describing pedestrians.
# First, if finds nearby pedestrians in the neighbourhood of the car.
# Then allocates an array of zeros a fixed max size (about 20).
# Then every found pedestrian is described in a row of that array:
# - "1" for marking that a pedestrian is found in the row
# (empty rows will have this still "0")
# - ped. speed
# - its relative position in the ego coordinates (x, y, z)
# - its relative orientation
# - pedestrian type
# - intent of crossing the road
pedestrian_descriptors: torch.FloatTensor
```

```
# A 1D array per ego vehicle describing its state. Specifically,
# - VehicleDynamicsState (acc, steering, pitch ..)
# - Vehicle size
# - Vehicle class
# - Vehicle dynamics type
# - Previous action
# - 2 lidar distance arrays, placed on the front corner of the vehicle
ego_vehicle_descriptor: torch.FloatTensor
```

B.2 RL expert

In this section we describe how we trained the reinforcement learning agent used to collect data in a diverse set of driving scenarios.

The agent maps a set of object-level vectors, described in section B.1, to actions executable by the simulated vehicle dynamics. The actions of the RL agent are modelled by two independent beta distributions, corresponding to acceleration and steering wheel angle. We partition the acceleration value $x \in [0; 1]$, sampled from the beta distribution, into accelerator pedal and brake pressure:

$$\begin{aligned} \text{Accel} &= s_1 \max\{0; 2x - 1\} \\ \text{BrakePressure} &= s_2 \max\{0; 1 - 2x\} \end{aligned}$$

where s_1 and s_2 scale the values to the respective output ranges. The three values accelerator pedal, brake pressure, and steering wheel angle are then fed into our vehicle dynamics simulation that closely matches our real world vehicles.

To learn the action distributions, we chose Proximal Policy Optimization [40] as a reinforcement learning algorithm due to its simplicity, widespread usage, and robustness. In each epoch of training we run the simulation with the agent in closed-loop and obtain a replay buffer. Note that we control multiple vehicles simultaneously, which allows us to generate a replay buffer with 100 to 200 thousand samples very efficiently. We then retrieve batches from that replay buffer and optimize the neural network using the PPO objective. After that, the replay buffer is discarded, and we begin the next epoch. We run this procedure for 10 thousand steps.

As the neural architecture, we use Perceiver IO [46]. The advantage of this architecture lies in its flexibility to deal with different input and output types. Input vectors from the simulator are preprocessed into tokens using multi-layer perceptrons with ReLU non-linearity, which are then fed into the cross-attention mechanism. Two output queries of the Perceiver are used to predict acceleration and steering wheel distributions, respectively. An additional output query predicts a value estimate.

The reward during training is given by the following formula, which is based on the rewards described in [52].

$$R = \underbrace{\beta_1 \left(1 - \frac{|v - v_{\text{desired}}|}{v_s}\right)}_{\text{Desired speed}} - \underbrace{\beta_2 d_{\text{lateral}}}_{\text{Lane distance}} - \underbrace{\beta_3 \alpha_{\text{angle}}}_{\text{Lane angle}} - \underbrace{\beta_4 \frac{(\phi_t - \phi_{t-1})^2}{\phi_s}}_{\text{Erratic steering penalty}} + \underbrace{R_{\text{failed to stop}}}_{\text{Stopping penalty}} \tag{1}$$

where v is the vehicle speed, v_{desired} is a desired speed heuristic (described below), d_{lateral} is the distance from vehicle centre to closest lane centre, α_{angle} is the angle between vehicle and closest lane tangent, ϕ_t and ϕ_{t-1} are steering wheel angle actions from the current and previous time step, and $R_{\text{failed to stop}}$ is another heuristic for yielding the right-of-way. β_1, β_2, β_3, and β_4 are hyper-parameters and v_s and ϕ_s are velocity and steering wheel scales.

The desired speed v_{desired} serves as a heuristic for determining the optimal speed in various situations. We address cases where the agent is in a junction, near another vehicle, at a traffic light, close to a pedestrian, or trailing a lead vehicle. For each of these situations, we establish a heuristic, and the final desired speed is selected as the minimum among them.

The failed to stop heuristic ensures the agent stops appropriately. It is given by

$$R_{\text{failed to stop}} = \begin{cases} -v & \text{if the agent is close to a give way sign and } v > 2m/s \\ 0 & \text{otherwise} \end{cases} \tag{2}$$