Distilling Multi-modal Large Language Models for Autonomous Driving

Deepti Hegde[1†] **Rajeev Yasarla**[1†] **Hong Cai**[1] **Shizhong Han**[1] **Apratim Bhattacharyya**[1]
Shweta Mahajan[1] **Litian Liu**[1] **Risheek Garrepalli**[1] **Vishal M. Patel**[2] **Fatih Porikli**[1]

[1]Qualcomm AI Research* [2]Johns Hopkins University
[†]Equal contribution

Abstract

Autonomous driving demands safe motion planning, especially in critical "long-tail" scenarios. Recent end-to-end autonomous driving systems leverage large language models (LLMs) as planners to improve generalizability to rare events. However, using LLMs at test time introduces high computational costs. To address this, we propose DiMA, an end-to-end autonomous driving system that maintains the efficiency of an LLM-free (or vision-based) planner while leveraging the world knowledge of an LLM. DiMA distills the information from a multi-modal LLM to a vision-based end-to-end planner through a set of specially designed surrogate tasks. Under a joint training strategy, a scene encoder common to both networks produces structured representations that are semantically grounded as well as aligned to the final planning objective. Notably, the LLM is optional at inference, enabling robust planning without compromising on efficiency. Training with DiMA results in a 44% trajectory error reduction in long-tail scenarios. DiMA also achieves state-of-the-art performance on the nuScenes planning benchmark.

1 Introduction

Multi-task planning systems that are trained in an end-to-end manner [14, 10, 6, 4, 7, 31] demonstrate improved performance over modular systems [16, 33, 21, 28, 5, 18, 19] but struggle with long-tail navigation and perception scenarios. Large language models (LLMs) have emerged as a promising solution to this issue. Trained on vast, internet-scale datasets, LLMs can leverage world knowledge to generalize to unseen or rare scenarios. These models can perform high-level reasoning tasks using mechanisms such as chain-of-thought [29]. Recent end-to-end autonomous driving systems leverage LLMs to achieve superior robustness to long-tail scenarios [22, 24, 27, 23, 13, 26]. To differentiate between these methods, we call end-to-end planners that depend on LLMs to perform trajectory prediction as "LLM-based" planners and ones that do not as "vision-based" planners [14, 12, 30]. In spite of their recent success "LLM-based" planners face significant challenges.

LLM-based planners require a significant amount of computational overhead at test time, limiting their practicality. This work addresses a core question: *how can we harness LLMs' world knowledge while preserving the efficiency of vision-based planners?* Secondly, bridging the visual and language domains is inherently more complex for end-to-end planning tasks compared to general MLLM tasks [15, 17], with the additional hurdle of limited training data.

To address these challenges, we propose **DiMA**, a novel framework for **Di**stilling **M**ulti-modal Large Language Models for **A**utonomous driving. We introduce a joint-training scheme between a vision-based planner and an MLLM that enables the learning of robust, grounded and disentangled scene representations that are aligned to the final objective of planning. Specifically, we use the vision-based

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

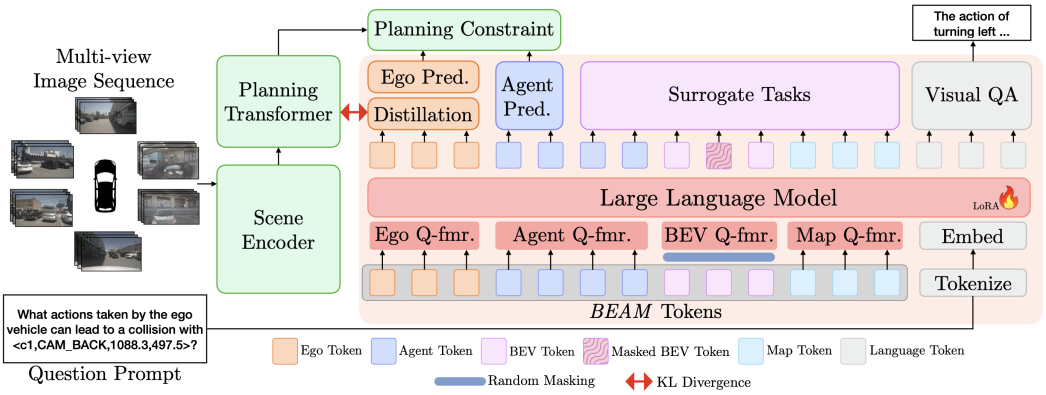

Figure 1: Overview of DiMA. The input to the framework is a multi-view image sequence and a question text prompt. The vision-based end-to-end planner consists of a scene encoder and a planning transformer. The scene encoder learns structured latent representations in the form of **b**ird's-eye-view, **e**go, **a**gent, and **m**ap ($BEAM$) token embeddings and acts as a trainable tokenizer for the multi-modal large language model (MLLM). The planning transformer is trained under standard planning constraints [14, 12]. The MLLM is trained for planning, visual question answering, distillation, and a series of surrogate tasks.

planner as a tokenizer, and pass the learned scene representations to the MLLM, providing a more structured input. The MLLM is then trained for the tasks of visual question-answering, trajectory estimation, and a series of surrogate tasks designed to ground the multi-modal inputs in language. The vision-based planner is simultaneously trained for trajectory estimation, while also distilling features from the MLLM planning head to the planning transformer. Crucially, **the MLLM can be discarded when performing planning inference**, maintaining the efficiency of the vision-based planner while utilizing the knowledge of the language model. Optionally, the MLLM can support language-guided reasoning through visual question answering. Our experiments show that DiMA is more robust to challenging long-tail scenarios compared to baselines and state-of-the-art methods [30, 23]. Our contributions are

- We introduce **DiMA**, an end-to-end autonomous driving framework that distills knowledge from an MLLM to a vision-based planner to ensure robustness to long-tail events while maintaining efficiency. DiMA is capable of planning as well as visual question-answering.
- We propose a distillation task along with a series of surrogate tasks to align the objectives of the vision-based planner and the MLLM.
- DiMA outperforms both vision-based and LLM end-to-end planners, achieving state-of-the-art results on the nuScenes planning benchmark. Training with DiMA results in s a 44% trajectory error reduction in long-tail scenarios.

2 DiMA Framework

We present DiMA, a framework for end-to-end autonomous driving. Given a sequence of multi-view images, the overall objective is to predict the future trajectory of the ego vehicle and answer questions about the scene. An overview of DiMA is shown in Figure 1. The framework has two main components: 1) the vision-based planner; 2) a multi-modal large language model consisting of adapter layers, an LLM, and a series of task-specific decoder heads.

2.1 Vision-based Planner

Vision-based end-to-end planners are trained in a multi-task fashion to perform perception, mapping, motion prediction and planning [14, 12, 30, 25]. Considering standard architecture design from [12, 14, 30] we decompose the planner into the scene encoder and the planning transformer. The scene encoder provides structured latent representations to the planning transformer that performs waypoint prediction. We call these representations as scene token embeddings. In our framework, we leverage the vision-based planner in two ways. First, it acts as the primary network through

which planning is performed, allowing for fast prediction at test time. Second, the scene encoder is shared with the the MLLM, acting as tokenizer to provide it with a highly structured set of inputs. An important distinction of our framework from previous works [13, 23] is that the scene encoder is trained jointly with the MLLM. The scene encoder models the scene as high-dimensional token embeddings that explicitly represent components such as the environment map, the ego vehicle, and the surrounding agents. This is achieved by introducing task specific, learnable query features that are supervised by a series of planning constraints [14].

2.2 Multi-modal LLM

Our objective is to distill knowledge from an MLLM to the vision-based planner. Towards this, we jointly train an MLLM with the vision-based planner. Specifically, the scene encoder is used as a trainable tokenizer for the MLLM to generate the $BEAM$ token embeddings, as well as used as an encoder for the planning transformer. The advantage from this strategy is two-fold. First, the MLLM receives a highly structured input that captures rich spatio-temporal information relevant to the autonomous driving task. Second, the scene encoder learns features that are grounded in language, improving the robustness of vision-based planning. The main components of the MLLM are adapter layers, the large language model, and a series of task-specific decoder heads which we discuss next.

Adaptation of scene tokens. In order to project the $BEAM$ token embeddings efficiently while maintaining their distinctiveness, we leverage the query-transformer (Q-former) module [11] to compress the visual tokens before being input to the LLM. We implement component-specific Q-former layers, namely a Map Q-former, a Bird's-Eye-View (BEV) Q-former, an Ego Q-former, and an Agent Q-former. Each of these adapters transforms the representing different modalities into a sequence of fixed-length, higher-dimensional tokens, preparing them for processing by the MLLM.

MLLM supervision. We design tasks for the MLLM with the objectives of, a) enriching the intermediate scene representations; b) grounding the scene token embeddings in language; and c) training the LLM for planning-related reasoning. In detail, the MLLM is trained for visual question answering, trajectory estimation, feature distillation, and a set of surrogate tasks [2, 32].

Table 1: Long-tail performance comparison of L2 trajectory error and collision rate on nuScenes [3] validation set using standardized evaluation [30]. The performance of vision planner DiMA model variants are in shades of purple. The performance of MLLM-branch DiMA model variants are in shades of blue. We summarize results by averaging over at $t = \{1, 2, 3\}s$ as well as at all time steps. The best average performance in each setting is in bold.

Method	Traj L2 (m)↓					Coll. (%) ↓
	1s	2s	3s	$\text{Ave}_{1,2,3s}$	Ave_{all}	Ave_{all}
3-point turn (zero-shot)						
VAD-Tiny	0.76	1.68	3.04	1.83	1.55	0.00
VAD-Base	0.71	1.66	3.24	1.87	1.57	0.00
PARA-Drive [30]	0.50	1.38	2.76	1.55	1.29	5.33
TOKEN [23]	0.39	1.29	2.60	1.43	1.18	4.00
DiMA (VAD-Tiny)	0.47	1.27	2.55	1.43	1.17	0.00
DiMA (VAD-Base)	0.36	1.18	2.37	1.30	1.05	0.00
DiMA (MLLM)	0.38	1.12	2.35	1.28	**1.04**	0.00
DiMA-Dual (VAD-Tiny)	0.38	1.08	2.35	**1.27**	**1.04**	0.00
Resume from stop						
VAD-Tiny	0.64	1.63	2.99	1.75	1.49	0.00
VAD-Base	0.60	1.72	2.83	1.72	1.42	0.00
PARA-Drive	0.14	0.79	2.30	1.08	0.85	0.00
TOKEN	0.13	0.70	1.58	0.80	**0.65**	0.00
DiMA (VAD-Tiny)	0.26	1.02	2.40	1.23	0.99	0.00
DiMA (VAD-Base)	0.15	0.65	1.34	**0.71**	0.66	0.00
DiMA (MLLM)	0.24	1.02	2.16	1.14	0.93	0.00
DiMA-Dual (VAD-Tiny)	0.24	0.98	2.13	1.11	0.91	0.00
Overtake						
VAD-Tiny	0.58	1.27	2.12	1.32	1.14	2.42
VAD-Base	0.46	1.16	2.17	1.26	1.06	2.49
PARA-Drive	0.27	0.89	1.94	1.03	0.85	2.30
TOKEN	0.29	0.77	1.63	0.90	0.74	**0.00**
DiMA (VAD-Tiny)	0.24	0.75	1.49	0.83	0.69	1.32
DiMA (VAD-Base)	0.24	0.72	1.50	**0.82**	**0.66**	1.29
DiMA (MLLM)	0.24	0.73	1.50	**0.82**	0.67	1.30
DiMA-Dual (VAD-Tiny)	0.24	0.73	1.50	**0.82**	0.67	1.30

Visual question answering. The MLLM is trained for visual question answering (VQA) on question-answer pairs of 4 types: perception of the scene, prediction of the agent behavior, prediction of ego-vehicle behavior, and future planning steps [22]. The answer is predicted based on a multi-modal prompt consisting of the question embeddings and the projected $BEAM$ token embeddings. This is constructed as a standard next-token prediction task and the VQA branch is supervised with a cross-entropy loss as in [15]. We denote the loss from this task head as $\mathcal{L}_{LLM}$.

Surrogate tasks: The main goal of joint training is to enrich the $BEAM$ scene representations learned by the scene encoder. We design MLLM surrogate tasks whose objectives are in line with that of future trajectory prediction. We point the reader to the appendix for more details on these tasks.

Distillation In order to further align the representations learned by the vision planner and the MLLM, we perform knowledge transfer between the penultimate layers of the language model and the planning transformer. Concretely, we minimize the KL-divergence between the distributions of the hidden features of the planning transformer and the LLM hidden features of the ego-token

embeddings:

$$\mathcal{L}_{distill} = D_{KL}(P_{llm}||P_{vis}) \tag{1}$$

where the P_{vis} are the features of penultimate layers of vision planning transformer and P_{llm} are the hidden LLM embeddings features corresponding to the ego-token embeddings of penultimate layers of MLLM model.

Loss functions. We supervise the network with a weighted sum of the losses corresponding to planning, visual question answering, distillation, and each surrogate task.

$$\mathcal{L} = \mathcal{L}_{planning} + \mathcal{L}_{LLM} + \mathcal{L}_{recon} + \mathcal{L}_{future} + \mathcal{L}_{distill} \tag{2}$$

here $\mathcal{L}_{planning}$ comes from the planning objective used to train [12, 14]. Here, the loss weights are chosen to bring each value to the same scale.

3 Experimental Setup

In this section, we present the training strategy and provide information about the datasets used. We refer the reader to the appendix for the architecture and evaluation strategy details.

Training. DiMA training follows a two-stage approach. First the vision-only planner is pre-trained for perception, prediction, and planning for 60 epochs in order to learn informative latent scene representations. Second, we perform joint training of the vision planner and the MLLM for an additional 30 epochs, incorporating all proposed tasks and losses detailed in Section B. In the second stage, the language model of the MLLM is fine-tuned using LoRA [9]. Please refer to the appendix for more details.

Datasets. We use the nuScenes dataset for the task of open-loop planning [3]. For supervising visual question answering, we train with DriveLM [22]. This dataset consists of a 4k subset of samples from the nuScenes dataset annotated with 300k QA pairs related to perception, prediction, planning, and behavior of the ego vehicle. For the samples in nuScenes that do not have text annotations, we create a set of perception, planning, and prediction QA pairs based on the numerical annotations. More details on this method as well as examples of generated QA pairs may be found in the appendix.

4 Experimental Results

We perform a comprehensive evaluation of DiMA on the nuScenes open-loop planning benchmark in a variety of evaluation settings. We compare our performance against that of recent vision-based planners [10, 12, 14, 30] as well as recent MLLM planners [20, 26, 24, 23].To facilitate a fair comparison of DiMA with recent works, we use a standardized evaluation detailed in [30].

Performance in long-tail scenarios. Autonomous driving performance is particularly crucial for rare scenarios. Using the manually selected long-tail events from [23], we demonstrate the robustness of DiMA in cases of novel navigation and perception. In Table 1, we evaluate the performance of DiMA for three long-tail scenarios, "3-point turn", "resume from the stop" and "overtake", selected from the nuScenes validation set. Training with DiMA results in a significant and consistent boost in performance from the baseline vision-based planners. We also consistently outperform PARA-Drive and TOKEN. Notably, the "3-point turn" case is a zero-shot scenario, not present in the training data. We achieve the lowest L2 trajectory error for this case. We also point out that the boost in performance comes with **no additional computational cost** on the base vision-based planner, making it more efficient (17 FPS) than LLM planners like TOKEN. See appendix for a latency comparison, further planning results and qualitative results on visual question answering.

5 Conclusion

We introduce DiMA, an end-to-end autonomous driving framework for robust and efficient planning. DiMA distills knowledge from an MLLM to a vision-based planner, employing a novel joint training strategy alongside carefully designed surrogate tasks such as masked token reconstruction, future token prediction, and scene editing. DiMA enables the model to learn semantically grounded scene representations, improving its performance in difficult navigation scenarios. We conduct extensive evaluations and ablation studies, showing that training with DiMA results in a planner that excels in long-tail cases while maintaining computational efficiency.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Apratim Bhattacharyya, Sunny Panchal, Reza Pourreza, Mingu Lee, Pulkit Madan, and Roland Memisevic. Look, remember and reason: Grounded reasoning in videos with language models. In *International Conference of Learning Representations*, 2024.

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[4] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021.

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[6] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17222–17231, 2022.

[7] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15793–15803, 2021.

[8] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[9] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[10] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *European Conference on Computer Vision*, pages 533–549. Springer, 2022.

[11] Wenbo Hu, Zi-Yi Dou, Liunian Harold Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka query transformer for large vision-language models, 2024.

[12] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[13] Jyh-Jing Hwang, Runsheng Xu, Hubert Lin, Wei-Chih Hung, Jingwei Ji, Kristy Choi, Di Huang, Tong He, Paul Covington, Benjamin Sapp, et al. Emma: End-to-end multimodal model for autonomous driving. *arXiv preprint arXiv:2410.23262*, 2024.

[14] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8350, 2023.

[15] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[16] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pages 1–18. Springer, 2022.

[17] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[18] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7577–7586, 2021.

[19] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified architecture for predicting multiple agent trajectories. *arXiv preprint arXiv:2106.08417*, 2021.

[20] Chenbin Pan, Burhaneddin Yaman, Tommaso Nesti, Abhirup Mallik, Alessandro G Allievi, Senem Velipasalar, and Liu Ren. Vlp: Vision language planning for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14760–14769, 2024.

[21] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 194–210. Springer, 2020.

[22] Chonghao Sima, Katrin Renz, Kashyap Chitta, Li Chen, Hanxue Zhang, Chengen Xie, Ping Luo, Andreas Geiger, and Hongyang Li. Drivelm: Driving with graph visual question answering. *arXiv preprint arXiv:2312.14150*, 2023.

[23] Ran Tian, Boyi Li, Xinshuo Weng, Yuxiao Chen, Edward Schmerling, Yue Wang, Boris Ivanovic, and Marco Pavone. Tokenize the world into object-level knowledge to address long-tail events in autonomous driving. *arXiv preprint arXiv:2407.00959*, 2024.

[24] Xiaoyu Tian, Junru Gu, Bailin Li, Yicheng Liu, Chenxu Hu, Yang Wang, Kun Zhan, Peng Jia, Xianpeng Lang, and Hang Zhao. Drivevlm: The convergence of autonomous driving and large vision-language models. *arXiv preprint arXiv:2402.12289*, 2024.

[25] Wenwen Tong, Chonghao Sima, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, et al. Scene as occupancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8406–8415, 2023.

[26] Shihao Wang, Zhiding Yu, Xiaohui Jiang, Shiyi Lan, Min Shi, Nadine Chang, Jan Kautz, Ying Li, and Jose M Alvarez. Omnidrive: A holistic llm-agent framework for autonomous driving with 3d perception, reasoning and planning. *arXiv preprint arXiv:2405.01533*, 2024.

[27] Wenhai Wang, Jiangwei Xie, ChuanYang Hu, Haoming Zou, Jianan Fan, Wenwen Tong, Yang Wen, Silei Wu, Hanming Deng, Zhiqi Li, et al. Drivemlm: Aligning multi-modal large language models with behavioral planning states for autonomous driving. *arXiv preprint arXiv:2312.09245*, 2023.

[28] Zitian Wang, Zehao Huang, Yulu Gao, Naiyan Wang, and Si Liu. Mv2dfusion: Leveraging modality-specific object semantics for multi-modal 3d detection. *arXiv preprint arXiv:2408.05945*, 2024.

[29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[30] Xinshuo Weng, Boris Ivanovic, Yan Wang, Yue Wang, and Marco Pavone. Para-drive: Parallelized architecture for real-time autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15449–15458, 2024.

[31] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *Advances in Neural Information Processing Systems*, 35:6119–6132, 2022.

[32] Rajeev Yasarla, Manish Kumar Singh, Hong Cai, Yunxiao Shi, Jisoo Jeong, Yinhao Zhu, Shizhong Han, Risheek Garrepalli, and Fatih Porikli. Futuredepth: Learning to predict the future improves video depth estimation. In *European Conference on Computer Vision*, pages 440–458. Springer, 2024.

[33] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[34] Jiang-Tian Zhai, Ze Feng, Jinhao Du, Yongqiang Mao, Jiang-Jiang Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *arXiv preprint arXiv:2305.10430*, 2023.

Appendix

A Overview

We present the appendix for the paper "Distilling Multi-modal Large Language Models for Autonomous Driving". In Section B, we provide details on the surrogate tasks module of DiMA . Section C has the details on the training setup and the procedure for generating additional text annotations for the nuScenes dataset. In Section D, we present additional qualitative results of DiMA's vision-based planning branch as well as visual question-answering results from the MLLM branch. We provide an additional quantitative evaluation of DiMA in Section E. In Section F, we provide a set of ablation experiments to examine the role of each surrogate task in planning performance.

B Surrogate tasks overview

An important component of training the MLLM branch of DiMA is the surrogate tasks module. In addition to being trained for planning and visual question answering, the MLLM is trained to perform the following surrogate tasks: masked reconstruction, future prediction, and scene editing. We design these tasks to enrich the **b**ird's-eye-view, **e**go, **a**gent, and **m**ap ($BEAM$) scene representations. Surrogate tasks module takes hidden token embedding of penultimate layer of the LLM model. An illustration of the module can be seen in Figure 2. Each decoder head in surrogate module is consists of 3 Linear layers with a ReLU activation layer. Below, we provide some additional details on the scene editing tasks well as overviews on the other two tasks.

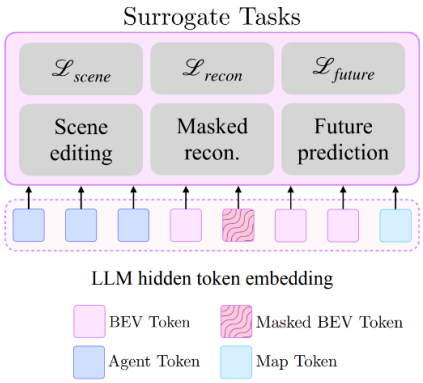

Figure 2: Overview of Surrogate tasks. Here hidden token embeddings are latent representations from the penultimate layer of LLM. These hidden token embeddings corresponding to **b**ird's-eye-view, **e**go, **a**gent, and **m**ap ($BEAM$) token embeddings are used as input as surrogate task decoder heads to perform masked reconstruction, future prediction and scene editing.

1) Masked token reconstruction: Each type of token embedding contributes to a holistic representation of the scene. In order to enrich the visual representations, we ask the network to reconstruct a masked BEV input based on the context present in the rest of the multi-modal sequence. We perform random masking after scene encoding and pass the token embeddings to the MLLM. A reconstruction head takes the latent representations from the penultimate layer of LLM and predicts reconstructed BEV token embeddings $\hat{B}$. This decoder head is supervised with the L2 loss between the prediction and the complete input,

$$\mathcal{L}_{recon} = ||\hat{B} - B||^2 \tag{3}$$

where $(\{m(B), E, A, M\})$ is input to the MLLM, and the latent MLLM representations associated with the BEV token embeddings are input to the masked reconstruction decoder head. Here, $m(.)$ denotes random masking.

2) Future BEV prediction: An important aspect of planning is anticipating future events. We introduce the surrogate task of future BEV prediction to encourage the LLM to learn spatio-temporal cues useful for planning. Given latent BEV token embeddings, we train a prediction head to predict future BEV token embeddings and supervise this task as the L2 loss between predicted and ground truth future token embeddings:

$$\mathcal{L}_{future} = ||\hat{F}_t - B_{t+1}||^2 + ||\tilde{F}_t - B_{t+2}||^2 \tag{4}$$

where $\hat{F}_t$, $\tilde{F}_t$ are the predicted future BEV token embedding at time t. Note, we predict future BEV token embeddings (B_{t+1}, B_{t+2}) of the multi-view image sequence for time steps $\{t+1, t+2\}$.

3) Scene editing: For prediction and reasoning about the ego vehicle, it is crucial to learn how surrounding agents impact the ego-vehicle's future path. We propose a novel scene editing task

in which we augment scenes by removing or adding new agents. Along with this, we construct a question-answer pair related to the edit. We show examples of this in Figure **??**. For scene addition, given the map constraints, predicted map, the ego bounding box location and the trajectories of predicted agents, we create a way-point trajectory for a new agent of category “car" or “truck". A new agent token embedding is then created using a linear layer. This new agent token embedding, the corresponding text prompt, and the rest of the $BEAM$ token embeddings are passed as input to the LLM. The hidden latent LLM features corresponding to agent token embeddings are then fed into a dedicated scene editing decoder head that performs waypoint prediction of the ego vehicle. The language prediction head performs question-answering on the new QA pair. This task thus contributes to the existing planning constraint loss and VQA loss of the MLLM.

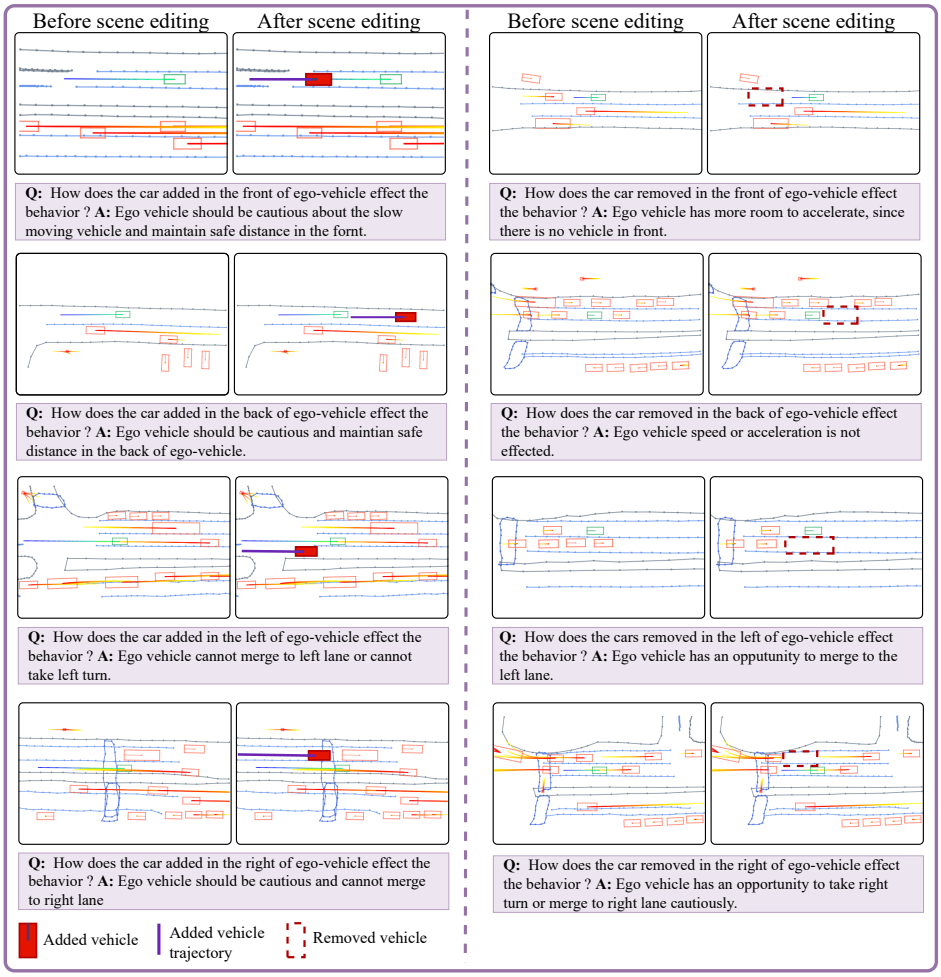

Figure 3: Examples of addition and deletion in scene editing. In the left column, a car (solid red box) is added in the premises of the ego-vehicle (green box). In the right column, a car (dashed red box) is removed from the premises of the ego-vehicle. A corresponding question-answer pair is created to characterize the edit.

We show further examples of this in Figure 3. For scene addition, given the map constraints, predicted map, the ego bounding box location and the trajectories of predicted agents obtained from perception, and prediction tasks of scene encoding (refer [12, 14], using camera meta data we project all these to two-dimensional space as shown in 3. Using this, we identify possible locations in the premises of ego-vehicle location, where a new object can be added and randomly choose the location for the new agent, which is of size maximum $2\times$ of size of ego-vehicle. Given the location for the new agent and map constraints, we create a way-point trajectory for a new agent of category “car" or “truck". A new agent token embedding is then created using a linear layer. This new agent token embedding, the corresponding text prompt, and the rest of the $BEAM$ token embeddings are passed as input to

the LLM. The hidden latent LLM features corresponding to agent token embeddings are then fed into a dedicated scene editing decoder head that performs way-point prediction of the ego vehicle. The output way-point prediction of the ego vehicle from scene editing head is supervised using $\mathcal{L}_{scene}$. $\mathcal{L}_{scene}$ is ego-agent collision constraint loss with updated agents incorporating either new added agent or removed agent in scene editing task. For ego-agent collision constraint loss refer [14].

The language prediction head performs question-answering on the new QA pair. This task thus contributes to the existing planning constraint loss and VQA loss of the MLLM. The QA pairs are generated according to a template in which the possible movements of the ego vehicle are described.

Table 2: Comparison of L2 trajectory error and collision rate on nuScenes [3] using standardized evaluation [30]. Models are evaluated on the general validation split as well as a "targeted" split of challenging samples from [30]. The performance of the DiMA model variants are in shades of purple. We summarize results by averaging over at $t = \{1, 2, 3\}s$ as well as at all time steps.

Method	Using Ego status	Traj L2 (m) ↓ 1s	2s	3s	$\text{Ave}_{1,2,3s}$	Ave_{all}	Collision (%) ↓ Ave_{all}
Full validation split							
UniAD[12]	✗	0.48	0.89	1.47	0.95	0.83	0.40
PARA-Drive[30]	✗	0.26	0.59	1.12	0.66	0.56	0.17
TOKEN[23]	✗	0.26	0.70	1.46	0.81	0.68	0.15
DiMA (UniAD)	✗	0.19	0.50	1.08	0.59	0.50	0.06
Targeted validation split							
UniAD[12]	✗	0.47	1.09	1.92	1.16	0.99	0.15
PARA-Drive[30]	✗	0.38	0.97	1.88	1.08	0.91	0.14
DiMA (UniAD)	✗	0.30	0.82	1.63	0.92	0.77	0.06
3-point turn (zero-shot)							
UniAD[12]	✗	0.68	1.55	2.90	1.71	1.43	0.00
PARA-Drive [30]	✗	0.50	1.38	2.76	1.55	1.29	5.33
TOKEN [23]	✗	0.39	1.29	2.60	1.43	1.18	4.00
DiMA (UniAD)	✗	0.28	0.94	2.16	1.13	0.90	0.00
Resume from stop							
UniAD[12]	✗	1.09	1.66	3.06	1.94	1.73	0.00
PARA-Drive	✗	0.14	0.79	2.30	1.08	0.85	0.00
TOKEN	✗	0.13	0.70	1.58	0.80	0.65	0.00
DiMA (UniAD)	✗	0.38	0.83	1.49	0.90	0.84	0.00
Overtake							
UniAD[12]	✗	0.60	1.39	2.38	1.45	1.27	0.98
PARA-Drive	✗	0.27	0.89	1.94	1.03	0.85	2.30
TOKEN	✗	0.29	0.77	1.63	0.90	0.74	0.00
DiMA (UniAD)	✗	0.28	0.75	1.55	0.86	0.78	0.41

C Experimental Setup

C.1 Architecture design

Vision-based planner We conduct experiments using two vision-based end-to-end planners, VAD [14] and UniAD [12] due to their performance, efficiency, and ability to model the interaction between scene components. We use two model size variants of VAD. Both planners are trained for perception, motion prediction and planning, while UniAD also performs occupancy prediction.

MLLM design The MLLM is made up of adapter layers, a language model, and a set of task-specific decoder layers. We project each $BEAM$ token embedding with its own dedicated Q-former adapter layer following [11]. After projection, the $BEAM$ token embeddings and language token embeddings lie in the same embedding space. In order to reduce memory consumption, we limit the agent token sequence length to a fraction of the total number of agents and perform sequence-wise upsampling after input to the language model. The adapter layers are randomly initialized. We use the LLM from LLaVA-v1.5-7B [17] as our language model base. The ego prediction and agent motion prediction task heads are multi-layer-perceptron networks following [14]. For the surrogate task decoder heads, we use 3 Linear layers with a ReLU activation layer.

C.2 Training details

We train DiMA-VAD-tiny, DiMA-VAD-Base, and DiMA-UniAD, using training setups adapted from [14] and [12]. Our training process follows a two-stage approach. First, we pre-train the vision planner only under the perception, prediction, and planning constraints for 60 epochs in order to learn informative latent scene representations. Second, we perform joint training of the vision planner and the MLLM for an additional 30 epochs, incorporating all proposed tasks and losses detailed in Section B. In the second stage, the language model of the MLLM is fine-tuned using LoRA [9]. Question-answer pairs from the augmented DriveLM dataset [22] are input along with the multi-view image sequence in the second stage. For each input sample, we randomly select one QA-pair from each category in every iteration to send as text prompt input to the network. This ensures diversity in questions while avoiding redundant visual inputs. For both stages, we employ the AdamW optimizer with a cosine annealing scheduler, a weight decay of 0.01, and an learning rate of 2×10^{-4}. In all our experiments we set the random masking ratio as $[0.2, 0.4]$.

C.3 Evaluation details

We evaluate planning performance based on the predicted future trajectories of the ego vehicle 3 seconds into the future, where 2 waypoints are predicted per second. We use two metrics, the L2 error (in meters) between the predicted and ground-truth waypoints as well as the collision rate (in %) between the ego-vehicle and the surrounding vehicles.

Standardized vs VAD evaluation. In [30], Weng point out inconsistencies in planning evaluation across VAD [14], UniAD [12] and AD-MLP [34], namely in the way L2 error is averaged over time and the way noisy and invalid frames are considered. The authors also point out that collision performance is considerably improved by using a more finely discretized BEV map. Accounting for these inconsistencies, they propose a "standardized" evaluation metric to maintain a fair comparison across methods. For accurate reporting and fair comparison with [30, 23], we use this standardized metric to evaluate our model. Certain existing works use the evaluation scheme followed by [14] (which we call VAD evaluation) and do not make code or models available [24, 26, 20, 10].

C.4 Generation of text annotations

We augment the existing text annotations of the Drive-LM [22] by generating question-answer (QA) pairs for samples in the nuScenes dataset [3]. First, we parse the numerical annotations of each object in the scene, such as the ego-vehicle and the surrounding objects. We denote each object as an agent and assign attributes such as the camera in which it is visible, the name of the object, and the vehicle speed. Additionally, we use rule-based algorithms to assign brief text descriptions of the future movement, the direction of movement relative to the ego-vehicle, the future speed, the type of interaction with the ego vehicle, and the probability of collision with the ego-vehicle. Using this annotation along with a few in-context examples, we prompt a Llama 3-70B model [8] to generate 5 Drive-LM-like QA pairs for each category of question. The input system prompt can be seen in Figure 6. An example of a textual description of the numerical annotations can be seen in Figure 7. Examples of generated QA pairs can be seen in Figure 8.

D Additional Qualitative Results

We present extensive qualitative results of DiMA . In Section D.1, we present a comparison of planning performance of the vision-based planner on nuScenes. In Section D.2, we provide numerous visual question-answering results on various subsets of the nuScenes dataset.

D.1 nuScenes planning

We present qualitative planning results of DiMA compared to that of VAD in Figure 5. We evaluate on difficult "targeted" samples from nuScenes. These are samples where the ego-vehicle is performing right and left turns. As seen in the figure, training with DiMA ensures safe trajectory prediction, avoiding collisions with vehicles when turning around corners (see row 1, columns 1 and 4) as well as avoiding lane departures (row 1 column 2). DiMA also results in more precise turns (see row 2).

In Fig. 1 of the main paper, we show a 3-point turn emphasizing complex maneuver, involving a sharp left turn, backward movement, and a second left turn, typically takes $\sim$1 minute to complete. In this supplemental material, we include a complete turn playable video visualization in Fig. 4 above.

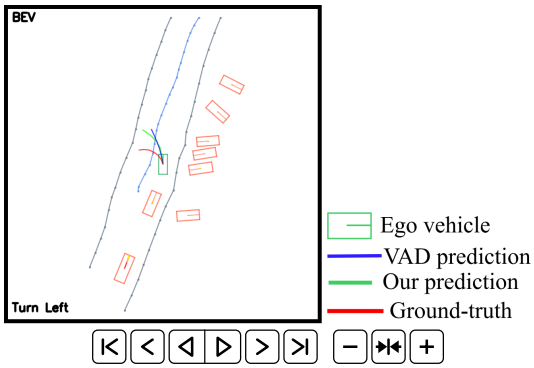

Figure 4: Visualization of three-point turn results. Press center buttons to play with Adobe Reader.

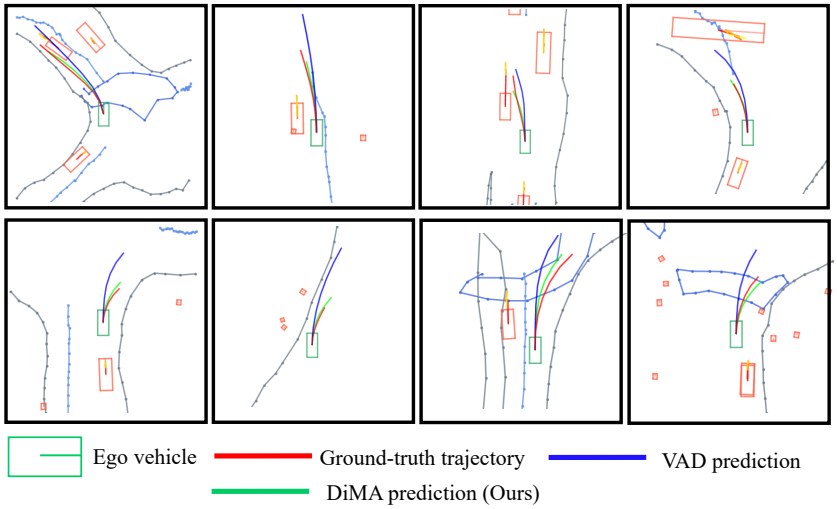

Figure 5: Visual comparison of the planning performance of DiMA (VAD-Tiny) with VAD-Tiny [14]. Samples are from the "targeted" subset of the nuScenes validation split.

D.2 Visual question-answering

We present numerous qualitative examples of planning and VQA performance of the DiMA MLLM branch. For Drive-LM test samples that have ground-truth annotations, we compare the generated text response with the ground-truth answer in Figure 9. We also plot the predicted future trajectory. In Figure 10 we show two scenarios in which DiMA provides incorrect VQA results.

For a more extensive qualitative analysis, we compare the performance of DiMA -MLLM with GPT-4 [1] in Figures 11, 12, and 13. We present common reasoning questions along with the DiMA -MLLM response and the GPT-4 response. The input to GPT-4 is the text prompt and a stitched image of the multi-view image set. We also show the planning performance plotted in the image as well as in a diagrammatic form on the right side of each row. As observed in these examples, DiMA is able to focus on objects important for navigation and planning. As can be seen in Figure 11 row 4, our model correctly predicts the future right turn to be taken by the ego-vehicle, while GPT-4 suggests the ego-vehicle should move straight. A similar problem is observed in Figure 13 row 4, where the prediction by GPT-4 is much more vague than that of DiMA .

E Additional Quantitative Results

VAD evaluation. For a fair comparison with recent works that use the evaluation strategy from VAD [14], we present an additional set of results under this strategy. [2] In Table 3, we present the evaluation of DiMA on the nuScenes validation set [3]. We significantly outperform the baseline vision-based planners VAD [14] and UniAD [12]. Notably, DiMA built on VAD-Tiny outperforms VAD-Base by 47% in L2 trajectory error while being 4 times faster. We also outperform MLLM planners like OmniDrive [26] and DriveVLM [24] without requiring an LLM at inference, making our approach more accurate as well as efficient.

We present the performance of DiMA-UniAD performance evaluated on the nuScenes dataset using standardized evaluation [30] in Table 2. We compare the performance of both DiMA-UniAD and UniAD [12] on the general validation split as well as a "targeted" split of challenging samples from [30] and on long-tail scenarios. We observe consistent improvement across all metrics, resulting in significantly reduced L2 trajectory error and collision rate. This model version also out-performs state-of-the-art methods PARA-Drive [30] and TOKEN [23] in almost all cases.

[2]We compare with reported results due to lack of published code.

QA pair generation with Llama-3-70B | System prompt

```
You are a system designed to generate high quality question answer
pairs in the scenario of autonomous driving, from the point of view
of an ego-vehicle viewing a 360 degree scene around you.
Questions-answer pairs may be of three types : Perception, Prediction,
and Planning. Perception questions relate to the nature of the
agents/objects around the ego-vehicle. Planning questions relate to
questions about the future actions of the ego-vehicle. Prediction
questions are detailed questions about the  agents/objects around the
ego vehicle. Here are some examples of each type of question.

Perception
 1. Q: 'What are objects to the front left of the ego car?'
    A: 'There is one truck and one car to the front left of the ego car.'
 2. Q: 'Are there moving pedestrians to the front right of the ego car?'
    A: 'Yes.'

Prediction
 1. Q: 'Is <c1,CAM_FRONT> a traffic sign or a road barrier?'
    A: 'No'
 2. Q: 'What object should the ego vehicle notice first when the ego
    vehicle is getting to the next possible location? .... [TRUNCATED]
    A: 'Firstly, notice <c6,CAM_FRONT,1074.8,336.5>..... [TRUNCATED]

Planning
 1. Q: 'What is the probability of colliding with <c1,CAM_FRONT> ?'
    A: 'Low'
 2. Q: 'In this scenario, what are safe actions to take ?'
    A: 'Brake gently to a stop, slightly offset to the right.'

For the given scene, generate five of each type of questions based on
the attributes provided of the ego vehicle and each agent
surrounding the ego
vehicle. Make sure that the answer is in the format

    {
        'Perception':
            {'Q':'Question','A':'Answer'},
        'Prediction':
            {'Q':'Question','A':'Answer'},
        'Planning':
            {'Q':'Question','A':'Answer'}
    }

Do not return any other text than the QA pairs in a correct python
dictionary format.
Make sure the answers are as descriptive as possible. Avoid
one word
answers or yes/no questions.
```

Figure 6: The system prompt given to Llama-3 to generate question-answer pairs.

QA pair generation with Llama-3-70B | Sample numerical annotation


```
"token": "f9878012c3f6412184c294c13ba4bac3",
"scene_description": "Car overtaking, parking lot, pedestrians,
pedestrian exiting car, objects on the ground",
"agent_attributes": {
    "c0": {
        "category": "truck",
        "speed": -0.0078431633186077 3,
        "assigned_cameras": [
            "CAM_BACK_LEFT"
        ],
        "future movement": "stopped",
        "future speed": "not moving",
        "direction": "towards from ego vehicle",
        "interaction_with_ego_vehicle_type": "none",
        "probability_of_collision_with_ego_vehicle": "low"
    },
    "c1": {
        "category": "pedestrian",
        "speed": 0.0,
        "assigned_cameras": [
            "CAM_FRONT",
            "CAM_FRONT_LEFT"
        ],
        "future movement": "stopped",
        "future speed": "not moving",
        "direction": "away from ego vehicle",
        "interaction_with_ego_vehicle_type": "none",
        "probability_of_collision_with_ego_vehicle": "low"
    },

    ... [TRUNCATED] ...

    "c24": {
        "category": "car",
        "speed": 0.0,
        "assigned_cameras": [
            "CAM_BACK"
        ],
        "future movement": "stopped",
        "future speed": "not moving",
        "direction": "away from ego vehicle",
        "interaction_with_ego_vehicle_type": "none",
        "probability_of_collision_with_ego_vehicle": "low"
    }
```


Figure 7: An example of the text-description of the numerical annotations of a scene from nuScenes. This JSON file is created using rule-based algorithms. This is appended to the system prompt.

QA pair generation with Llama-3-70B | Generated QA pairs

Perception

```
{
    "Q": "What are objects to the back right of the ego car?",
    "A": "There is one car and one bicycle to the back right of
         the ego car."
},
{
    "Q": "Are there moving agents to the right of the ego car?",
    "A": "Yes, there is one car to the right of the ego car."
},
... [TRUNCATED] ...
```

Prediction

```
{
    "Q": "What are the future movements of the agents to the back
         rightof the ego car?",
    "A": "The car will slightly steer to the right and the bicycle
         will remain stopped."
},
{
    "Q": "What are the future speeds of the agents to the back
         right of the ego car?",
    "A": "The car will be driving fast and the bicycle will not be
         moving."
},
... [TRUNCATED] ...
```

Planning

```
{
    "Q": "What is the probability of colliding with the car after
         the ego vehicle steps on the brakes?",
    "A": "Medium"
},
{
    "Q": "What actions taken by the ego vehicle can lead to a
         collision with the bicycle?",
    "A": "No action."
},
... [TRUNCATED] ...
```

Behavior

```
    "Q": "Predict the behavior of the ego vehicle.",
    "A": "The ego vehicle is slightly steering to the right.
         The ego vehicle is moving at a moderate speed."
```

Figure 8: Some examples of generated QA pairs. The perception, prediction, and planning pairs are generated with Llama-3. The behavior QA is created using the future motion of the ego vehicle.

Table 3: Comparison of L2 trajectory error and collision rate on nuScenes [3] using VAD evaluation [14]. Models are evaluated on the general validation split. The performance of the DiMA model variants are in shades of purple. The performance of the MLLM-branch DiMA model variant is in blue. We summarize results by averaging over all time steps. The best average performance in each setting is in bold. "+" indicates the use of ego-status information.

Method	Using Ego status	Traj L2 (m) ↓				Collision (%) ↓				Latency (ms)	FPS
		1s	2s	3s	Ave_{all}	1s	2s	3s	Ave_{all}		
ST-P3 [10]	✗	1.33	2.11	2.90	2.11	0.23	0.62	1.27	0.71	628.3	1.6
UniAD[12]	✗	0.48	0.96	1.65	1.03	0.05	0.17	0.71	0.31	555.6	1.8
VAD-Tiny[14]	✗	0.46	0.76	1.12	0.78	0.21	0.35	0.58	0.38	59.5	16.8
VAD-Base[14]	✗	0.41	0.70	1.05	0.72	0.07	0.17	0.41	0.22	224.3	4.5
VLP-VAD-Base[20]	✗	0.26	0.47	0.78	0.50	0.12	0.17	0.42	0.23	–	–
OmniDrive [26]	✗	0.40	0.80	1.32	0.84	0.04	0.46	2.32	0.94	–	–
DiMA (UniAD)	✗	0.15	0.30	0.56	0.34	0.06	0.08	0.22	0.12	560	1.8
DiMA (VAD-Tiny)	✗	0.18	0.36	0.61	0.38	0.07	0.10	0.27	0.15	59.5	16.8
DiMA (VAD-Base)	✗	0.13	0.27	0.47	**0.29**	0.05	0.08	0.16	**0.10**	226	4.5
VAD-Tiny	✓	0.20	0.38	0.65	0.41	0.10	0.12	0.27	0.16	59.5	16.8
VAD-Base	✓	0.17	0.34	0.60	0.37	0.07	0.10	0.24	0.14	224.3	4.5
DriveVLM [24]	✓	0.18	0.34	0.68	0.40	0.10	0.22	0.45	0.27	–	–
DriveVLM-Dual (VAD-Base) [24]	✓	0.15	0.29	0.48	0.31	0.05	0.08	0.17	0.10	–	–
DiMA + (VAD-Tiny)	✓	0.17	0.33	0.59	0.37	0.07	0.10	0.26	0.14	59.5	16.8
DiMA + (VAD-Base)	✓	0.12	0.25	0.44	**0.27**	0.04	0.06	0.15	**0.08**	226	4.5
DiMA-Dual+ (VAD-Tiny)	✓	0.14	0.27	0.46	0.29	0.05	0.07	0.15	0.09	286	3.5

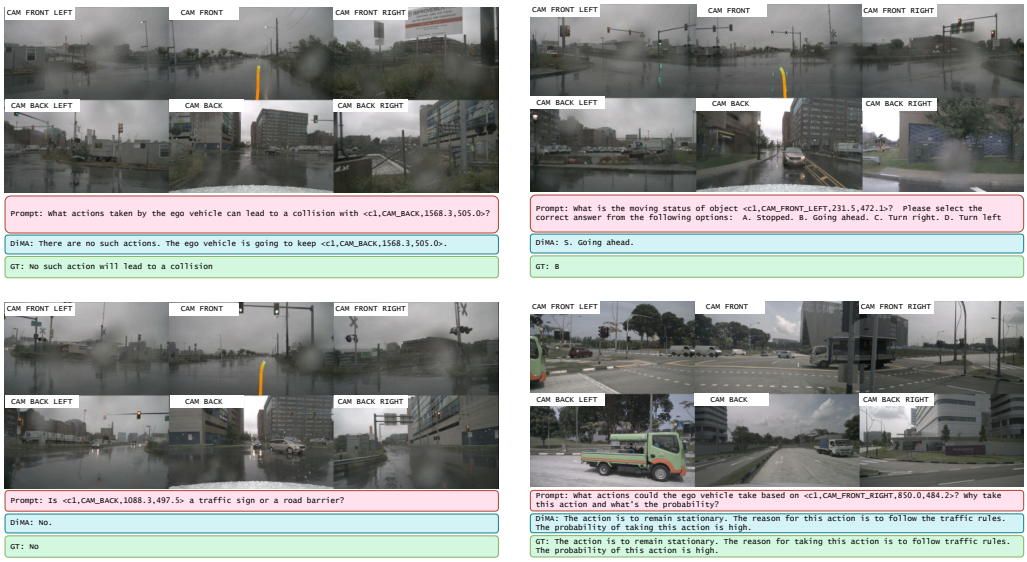

Figure 9: Visualization of VQA and planning prediction by the MLLM branch of DiMA-VAD-Tiny. We plot the predicted trajectory (orange-yellow) and show an example response of the language model branch to a question from the DriveLM test dataset [22].

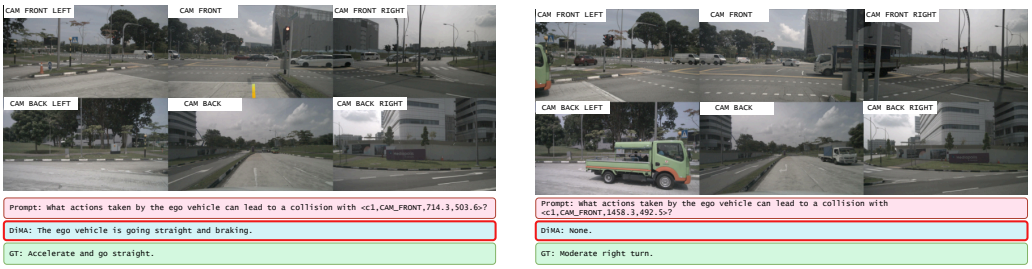

Figure 10: Visualization of failure cases of DiMA-VAD-Tiny. We plot the predicted trajectory (orange-yellow) and show an example response of the language model branch to a question from the DriveLM test dataset [22].

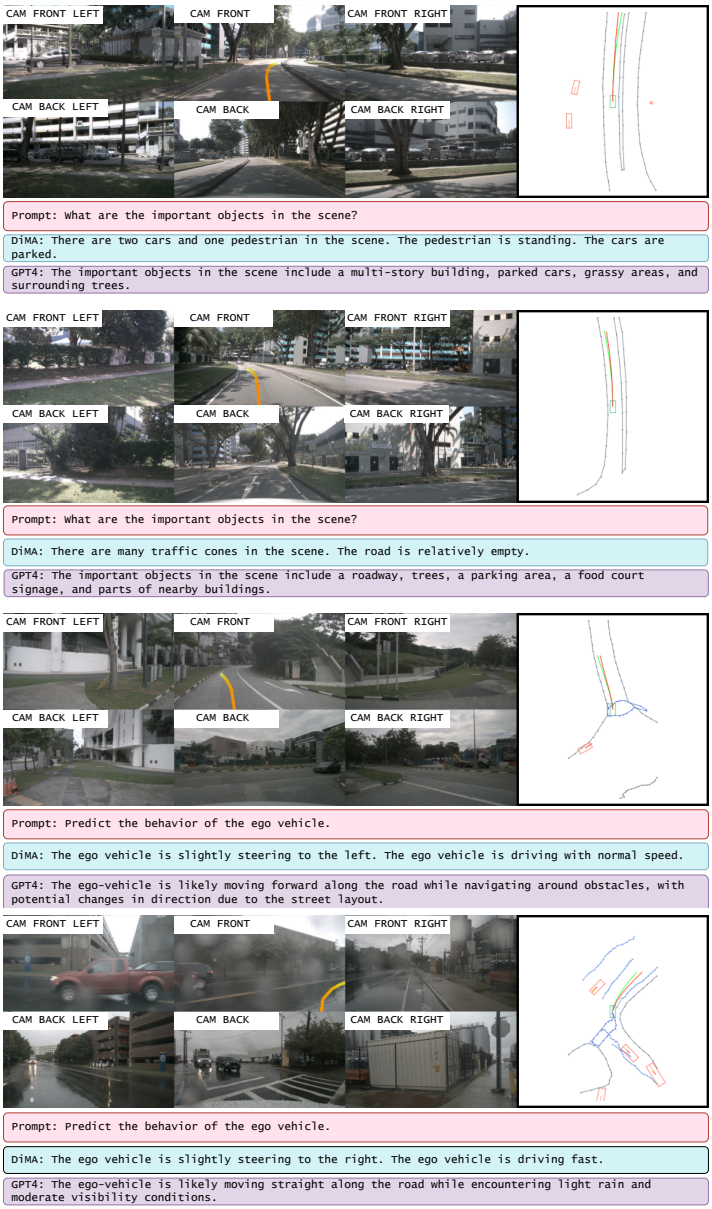

Figure 11: Visualization of visual question-answering on the targeted subset of the nuScenes dataset. On the image, we plot the predicted trajectory (orange-yellow) The red line is the ground-truth trajectory. In the diagram, the green line is the predicted trajectory.

F Ablation study

We examine the role of different design choices and proposed tasks in the DiMA framework through a comprehensive ablation study in Table 4. In order to do so, we build our final frame-work step-by-step. We begin with VAD-Tiny as our baseline model in the first row, denoted by ID-1. In ID-2, we observe inconsistent gains and drops in performance by naively training an LLM with the VAD-Tiny under a the planning and VQA objectives while using only BEV features. By comparing ID-3 and ID-4, we show that that providing more scene context to the MLLM by using all the $BEAM$ token embeddings to train both LLM and vision-based planner benefits the performance, resulting a significant improvement in trajectory estimation and collision rate. We show in ID-5 that explicitly distilling the information from the penultimate layers of the MLLM to the planning transformer gives another boost in performance. In ID-6, ID-7 and ID-8 we demonstrate the importance of surrogate

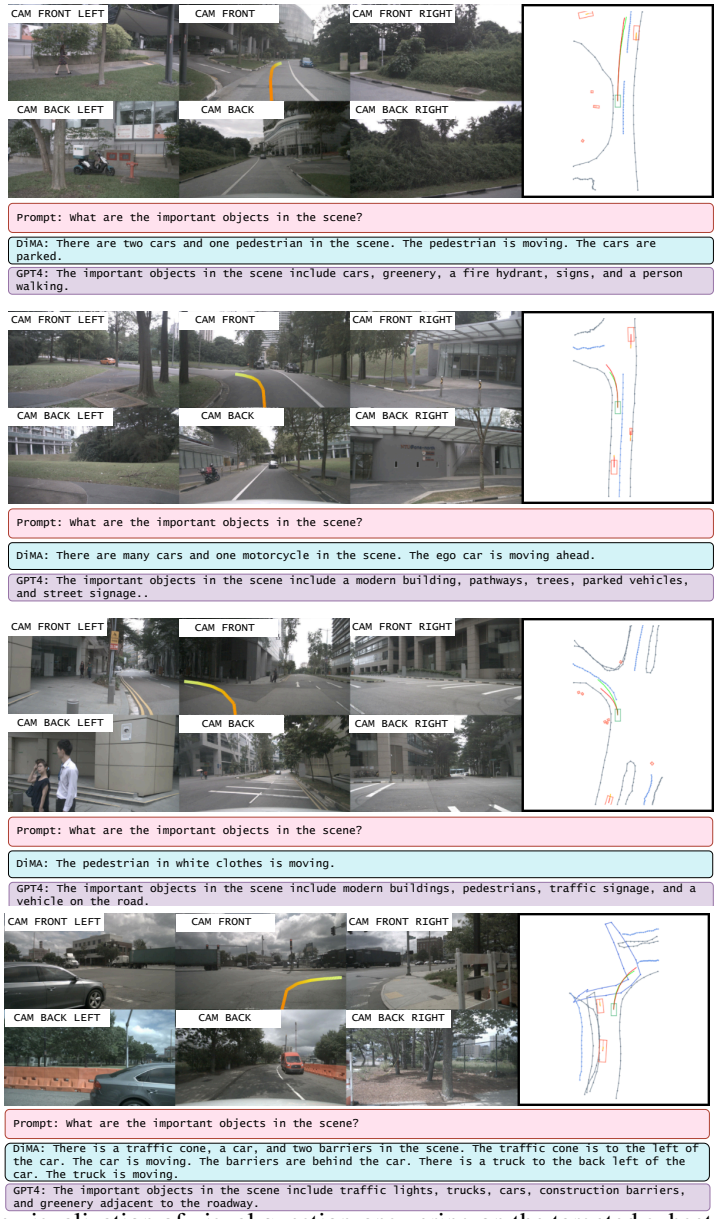

Figure 12: More visualization of visual question-answering on the targeted subset of the nuScenes dataset. On the image, we plot the predicted trajectory (orange-yellow) The red line is the ground-truth trajectory. In the diagram, the green line is the predicted trajectory.

tasks like masked-reconstruction, future prediction and scene editing in helping the scene encoder to learn richer representations useful for planning. We include additional ablation studies to analyze the effect of surrogate tasks across targeted and long-tail scenarios in nuScenes in Tab. 5 of this appendix. Our experiments demonstrate that closer the objective of the surrogate task is to the planning task, the more it boosts performance. For example, masked reconstruction enhances the visual representation. We observe a stronger boost in planning by training for future prediction, which encourages temporal consistency. The task of scene editing is most beneficial, as it encodes agent behaviors and reasoning patterns, further improving planning performance.

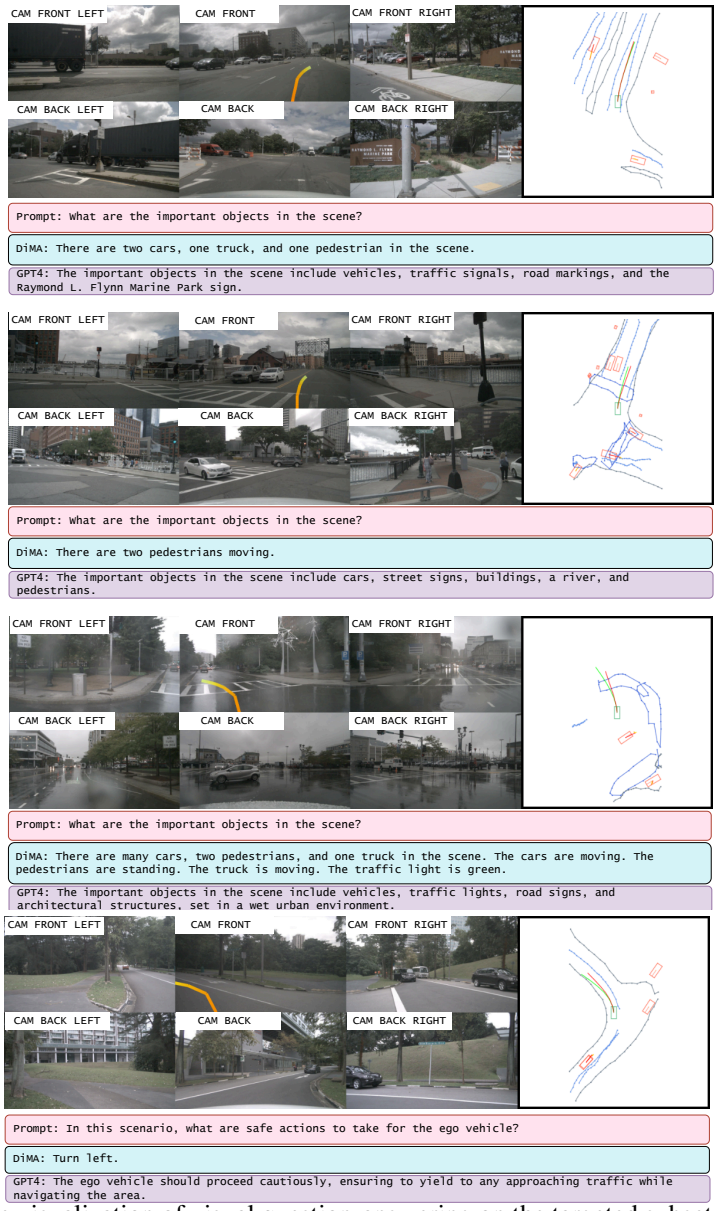

Figure 13: More visualization of visual question-answering on the targeted subset of the nuScenes dataset. On the image, we plot the predicted trajectory (orange-yellow) The red line is the ground-truth trajectory. In the diagram, the green line is the predicted trajectory.

Table 4: Results of ablation experiments of DiMA-VAD-Tiny on nuScenes under VAD evaluation. The baseline model is VAD-Tiny. We report L2 trajectory error and collision rate averaged over all time steps.

ID	VQA	Scene tokens	Distillation	LLM Planning	Surrogate tasks			Traj L2 (m) ↓				Collision (%) ↓			
					Masked recon.	Future pred.	Scene editing	1s	2s	3s	Ave_{all}	1s	2s	3s	Ave_{all}
1	✗	✗	✗	✗	✗	✗	✗	0.33	0.58	0.89	0.60	0.12	0.19	0.55	0.29
2	✓	BEV	✗	✓	✗	✗	✗	0.36	0.60	0.91	0.62	0.10	0.17	0.52	0.26
3	✓	BEV, Map	✗	✓	✗	✗	✗	0.28	0.52	0.87	0.56	0.10	0.17	0.49	0.25
4	✓	All	✗	✓	✗	✗	✗	0.26	0.51	0.83	0.52	0.10	0.16	0.38	0.21
5	✓	All	✓	✓	✗	✗	✗	0.22	0.44	0.77	0.48	0.09	0.14	0.34	0.19
6	✓	All	✓	✓	✓	✗	✗	0.19	0.39	0.68	0.42	0.09	0.13	0.32	0.18
7	✓	All	✓	✓	✓	✓	✗	0.18	0.37	0.63	0.39	0.07	0.11	0.29	0.16
8	✓	All	✓	✓	✓	✓	✓	0.18	0.36	0.61	0.38	0.07	0.10	0.27	0.15

Table 5: $Avg_{all}(\downarrow)$ L2 trajectory error on nuScenes dataset using standardized evaluation ParaDrive

Method	Surrogate			targeted	long-tail		
	Masked recon.	Future pred.	Scene editing		3point	overtake	resume
VAD-Tiny	✗	✗	✗	1.37	1.83	1.75	1.32
DiMA (VAD-Tiny)	✓	✓	✓	0.97	1.43	1.23	0.83
	✓	✗	✗	1.22	1.67	1.51	1.07
	✗	✓	✗	1.16	1.56	1.43	0.98
	✗	✗	✓	1.09	1.51	1.39	0.96