

End-to-End Driving Trajectory Prediction with Vision-Language-Action Model

Stanford CS224N Custom Project

Anze Liu

Department of Computer Science
Stanford University
anzeliu@stanford.edu

Abstract

Traditional autonomous driving involves modular software functions, including perception, prediction, and planning. More recently, end-to-end (E2E) learning-based approaches emerged to map raw sensor inputs to various driving specific outputs, including planner trajectories. Coupled with the earlier, more established Vision-Action (VA) models and recent advancements in Multimodal Large Language Models (MLLMs), learning from different modalities to generate action plans have shown promising results for E2E autonomous driving. Driven by interest in how different modalities are integrated together and how natural language can influence autonomous driving, this project proposes E2EVLA, a Vision-Language-Action (VLA) model for E2E driving trajectory prediction. The objective is to investigate if a frozen reasoning-capable Vision-Language Model (VLM) can provide action advices based on semantic scene understanding to influence a spatial-BEV-based trajectory planner, to improve the Rater Feedback Score of predicted trajectory waypoints, particularly in rare or difficult driving scenarios. The final VLA model has 5 variants that differ in how language knowledge is extracted from Cosmos Reason 2 and how it is fused into the Latent Transfuser trajectory predictor. The best performing VLA variant trained on Waymo E2E Driving subset dataset achieved Rater-Feedback Score (RFS) of 7.75 on the test set, exceeding both the simplest baseline (CNN+GRU) with RFS of 7.18 and the strongest baseline (Latent Transfuser+MLP) with RFS of 7.63. Some of the key challenges in bridging the gap between two modality spaces is also highlighted through empirical analysis.

1 Key Information

- Mentor: Luke Bailey
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Trajectory prediction forecasts the future waypoints of an agent from their observed past states and scene context. This problem remains challenging due to multimodal uncertainty, long-range temporal dependencies, and out-of-distribution scenarios. Recent studies explore LLMs as a way to offer the general knowledge or embodied reasoning capabilities. Their autoregressive nature enables modeling of data with sequential dependencies. Pretraining large-scale LLMs also enable zero- or few-shot generalization. These two main traits aligned naturally with trajectory prediction, which requires scene semantic understanding and traffic agents interactions. LLMs can be leveraged for trajectory prediction in different ways, including converting contextual data to token sequences or as part of descriptive prompts, with some studies directly employ pretrained models, some adapt them via fine-tuning, instruction tuning, or more lightweight tuning via LoRA. How language-based features,

such as reasoning or generated answer, extracted from large language model integrate with trajectory planner to improve E2E driving trajectory prediction in a VLA architecture is the core problem explored in this project. The main findings are:

- Naive integration of language reasoning does not automatically improve trajectory prediction
- Cross-modal alignment across vision, language, and action is the main bottleneck
- Action-level conditioning performed better than reasoning when using lightweight integration
- FiLM-based conditioning provides the most stable way to inject signals from language space

3 Related Work

MotionLM[1] first formulated motion forecasting as a language modeling problem. Trajectories are represented as sequences of discrete motion tokens and the goal is to model a distribution over multi-agent interactions. LMTraj[2] reformulates trajectory prediction as a question-answering task by converting numerical features into structured prompts and using a decimal-aware tokenizer to bridge the gap between continuous values and discrete tokens. Traj-LLM[3] encodes multi-modal scene context into transformer compatible sequences, fine-tunes GPT-2 backbone with LoRA, and applies lane-aware probabilistic learning. These language-based predictors embed reasoning directly while generating trajectory forecasts, but their dependency on token granularity and autoregressive decoding can result in numerical imprecision and latency bottleneck. There exists a trade-off between spatial control and interpretability: LLM-based predictors can explain intent but struggle to anchor it geometrically to produce physically feasible trajectory. Besides generating trajectories, natural language is also used as textual inputs, intermediate representations, or language-conditioning. For example, LLM-Powered [4] leverages GPT-4V to derive semantic features that are one-hot encoded as cues to the attention layers of a motion-prediction network. Related efforts highlight the emerging role of multimodal LLMs as reasoners to condition trajectory prediction. The computation overhead and dependency on prompt quality are still the main challenges. CoT-Drive [5] verbalizes intermediate steps, such as “the car approaches a stop sign, slowing down is expected”, before final trajectory generation. Generating reasoning text to influence trajectory generation improves both accuracy and interpretability but again incurs high compute cost. Also, LLM can diverge within itself: the language reasoning verbalized from its internal decision logic may not align with its own trajectory output. Evaluating model faithfulness is difficult, as methods such as counterfactual testing or rational-action consistency evaluation require carefully engineered metrics. Overall, the main challenges in LLM-trajectory integration can be summarized as 1) how to represent multimodal contextual features and trajectory inputs to be language model compatible, 2) how to prompt and generate reasoning that is coherent with trajectory outputs with reasonable latency, 3) how to evaluate predicted trajectories and LLM’s influence considering not just geometric errors but safety and physical realism.

Recent works have explore Chain-of-Thought reasoning to offer interpretability when solving complex driving scenarios. However, according to AdaThinkDrive[6], the generation of extensive intermediate sequences increases inference cost and results in over-thinking that is unnecessary or sometimes irrelevant to the visual inputs. In addition, natural language may not always be the optimal medium for structured reasoning, which shifts reasoning from language to latent space. Presented by Chain-of-Continuous-Thought[7], the last hidden state of the LLM is used as a representation of the reasoning state, termed the continuous thought, instead of decoding this into a word token, it is fed back to the LLM as the subsequent input embedding directly in the continuous embedding space.

Inspired by these works, leveraging language represented via latent features and actual text as a conditional signal is explored in this project, using the following models. Cosmos Reason [8] is a multimodal LLM with Qwen3-VL [9] as the backbone architecture, specifically designed for physical AI applications, including robots and vision AI agents that interact with the physical world, post-trained on 3.7M Visual Question Answering (VQA) samples to develop physical common sense and embodied reasoning capabilities. The input video or image is projected into the LLM’s token embedding space as visual tokens by a vision encoder followed by a projector. The input text is tokenized and concatenated with the visual tokens. Together the input tokens are fed into the LLM backbone, a hybrid Mamba-MLP-Transformer architecture, which then output responses with long Chain-of-thought reasoning processes. Cosmos Reason is trained in two stages: Physical AI supervised fine-tuning and Physical AI reinforcement learning, in order to build the physical common sense and embodied reasoning. With spatial, temporal, and physics understanding, Cosmos Reason can serve as a planning model to reason about what steps an embodied agent might take next. Qwen3-

VL is composed of a Vision Transformer (ViT) encoder and a language model decoder. Its post-training is divided into non-thinking and thinking variants to address different application needs, with the latter trained on Chain-of-Thought (CoT) curated dataset and demonstrates stronger multimodal reasoning capabilities. The non-thinking mode still has outstanding performance compared to similar non-thinking or low-thinking-budget models and its inference speed is also much faster. Cosmos Reason in its two modes of thinking are adopted in the VLA architecture of this project. Navsim’s reimplementation of Latent Transfuser (LTF) [10] is used as the vision-action backbone in this project. LTF taking as inputs the latest timestep’s status, routing intent, and front-viewing camera images. LTF is pretrained on OpenScene[11], a redistribution of the public large-scale annotated nuPlan [12].

4 Approach

The proposed approach runs the pretrained LTF backbone to extract a trajectory query and runs the pretrained Cosmos Reason 2 to extract language features. To aggregate both features, fusion modules are developed to combine the information learned from vision-action space and vision-language space, followed by transforming the language-influenced new trajectory query to trajectory waypoints.

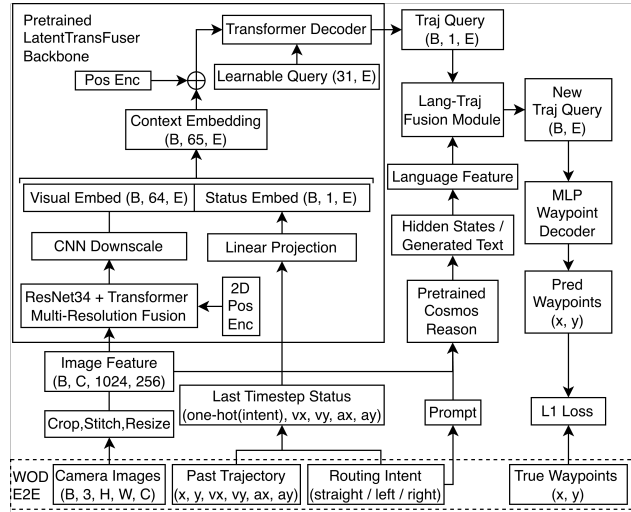


Figure 1: Architecture of the proposed approach: trajectory query obtained with LTF, language feature extracted with Cosmos Reason 2, and fusion of extracted features to predict waypoints.

4.1 Visual Features to Trajectory Query

The pretrained LTF backbone with 55.44 million parameters is adapted with new input preprocessing and waypoint decoder, serving as the baseline as well as the backbone of the final VLA model. In input preprocessing, the front-left, front, and front-right camera images are cropped and stitched horizontally to achieve a 4:1 aspect ratio, then resized to have a 1024 x 256 front-viewing image. Intent is one-hot encoded and concatenated with ego’s current velocity and acceleration, together as a status feature. LTF first applies CNNs on the image feature and on the learnable positional encoding structured the same way as a discretized LiDAR BEV grid would have. The visual features extracted are fused over several convolution stages with Transformers: layers of Resnet34, GPT backbone, and CNN fusion combinations. The step produces a 64-token long 256-dim feature vector that constitutes a compact representation of the driving scene. On the other hand, the status feature is encoded with a linear layer to map from dim-8 (4 from one-hot + 2 from velocity + 2 from acceleration) to dim-256. The dimension aligned image feature and status feature are concatenated along the token dimension. Then a Transformer decoder takes the feature sequence as key and value, along with a sequence of 31 learnable embeddings as query, of which the first query represents the trajectory query. A new MLP-based waypoint decoder is implemented to replace the original LTF’s MLP head in order to produce the 20 2D waypoints required by Waymo E2E Driving task.

4.2 Language Feature Extraction

As a separate stream of feature extraction, the pretrained Cosmos Reason 2 with 2.44 billion param is used as the VLM in the VLA architecture and loaded in 4-bit NF4 quantization to reduce memory consumption. The same preprocessed front-view camera image is passed to Cosmos, ensuring both backbone models extract features about the same visual content. Different from the one-hot encoded routing intent for LTF, Cosmos takes the routing intent as natural language as part of the instruction prompt. Cosmos textual output is integrated with trajectory query output from LTF to achieve language-conditioned trajectory prediction. Three modes of this language-action multimodal fusion is explored: 1) LatentReason: Cosmos with CoT reasoning enabled and reason embeddings are used to condition trajectory prediction, 2) LatentAnswer: Cosmos with CoT disabled and answer embeddings condition prediction, 3) TextAnswer: Cosmos with CoT reasoning disabled and the generated answer text condition trajectory prediction. Each variant results in different language features extracted.

VLA Mode 1 LatentReason prompt, with intent replaced with per-sample routing intent:

"Based on the front camera view and the driving intent as intent. What's the next immediate action in a few words for the ego vehicle? Reason very briefly for only and exactly one step. Answer the question using the following format: <think> Your reasoning, must be short. </think> Write your final answer immediately after the </think> tag."

VLA Mode 2 LatentAnswer and Mode 3 TextAnswer share the same prompt:

"Based on the front camera view and the driving intent as intent, describe the ego vehicle's immediate driving behavior. Select all that apply from: [keep straight, slight left, sharp left, slight right, sharp right, lane change left, lane change right, accelerate, maintain speed, decelerate, emergency brake, creep forward, stop, yield to pedestrian, yield to vehicle, follow leader, merge, overtake]. List the selected actions separated by commas, nothing else."

LatentReason builds a prompt with the <think> tag and enables Qwen thinking, so that the generated response includes CoT reasoning. The final prompt is composed of system prompt, front-view camera image, and intent-informed instruction, which is tokenized and passed to Cosmos. Cosmos performs its autoregressive generation with a max 128 tokens, considering generation latency, which means the generated response may be cut short. Therefore, instead of using the generated text, this model variant uses the latent feature, which encode important internal logic that can influence trajectory. Next, the "last token, last layer" hidden state is extracted per generation step and stacked together, which forms the reason embedding, a sequence of 2048-dim vectors, one per reasoning token. Rather than using the text output, which is discarded here, the internal hidden states are used as a dense, continuous representation of the reasoning process. Since generated responses can have variable length, potential concerns are which embeddings can actually influence trajectory, whether to use ones corresponding to reason text or final answer, and how to locate those specific embeddings. The resulting language feature is a reason embedding of shape $[B, T_{\text{reason}}, H]$, where B is batch size, T_{reason} is reason sequence length, and H is Cosmos text hidden size 2048.

Due to long latency in autoregressively generating full reasoning, LatentAnswer and TextAnswer have Cosmos thinking disabled, treating it as an VLM with no CoT. Although the accuracy of Cosmos generated answer is affected, this configuration is more time-efficient and worth experimenting. Cosmos is instructed to return all the actions that apply to the driving scene given the camera image and original driving intent, selected from the predefined list of actions. Cosmos directly returns the a set of actions in text. LatentAnswer discards the generated answer text and use the "last token, last layer" hidden state from each step of generating the actions answer, similar to LatentReason, but with a smaller max new tokens of 24. The hypothesis is that the actions answer tokens are more decisive signals to condition trajectory, at the cost of losing deliberative reasoning. Instead of using continuous hidden states from the VLM, TextAnswer performs detokenization to obtain the answer text, parses the set of actions in answer (fallback to "stop" if parsing fails), and apply multi-hot encoding to create discrete action embedding: an 18-dim binary vector.

4.3 Language-Trajectory Feature Fusion

Continuing from the 3 types of language features extracted from Cosmos, different fusion modules are developed. In Mode 1 and 2, the language features, which are reason embedding and answer embedding, respectively, are attention pooled to extract 1 embedding from the sequence-length of embeddings. Then a linear projection is applied to align with trajectory query dim-256, followed by a

gated residual to train the fusion module to decide how much language features to depend on, which prevents noisy latent features to corrupt the trajectory query. While this is an advantage of this gated approach, the unintended consequence is the lack of language influence to trajectory. Two variants of this fusion structure are experimented, one with learnable parameters as gate and another without gate to force latent language feature conditioning. For TextAnswer, two sub-variants are experimented: element-wise summation and Feature-wise Linear Modulation (FiLM)[13]. Summation approach first projects the discrete action embedding to align with trajectory query dim and then directly adds up the two. FiLM is a conditioning method that applies a channel-wise affine transformation to scale and shift trajectory features driven by external language-based signals.

	Mode 1 LatentReason	Mode 2 LatentAnswer	Mode 3 TextAnswer
Thinking	Enabled (≤ 128 tokens)	Disabled (≤ 24 tokens)	Disabled (≤ 24 tokens)
Feature	Hidden States	Hidden States	Generated Text to Multi-hot
Bottleneck	Long seq of dim-2048	Short seq of dim-2048	18-bit binary
Fusion	AttentionPool+Residual	AttentionPool+Residual	ActionEmebd+Sum/FiLM
Speed	Slowest	Medium	Fastest

Table 1: Summary comparison of the different VLA operating modes.

After fusion module, a 2-layer MLP with ReLU activation and hidden size of 2560 is applied to transform the language-conditioned trajectory query to trajectory waypoints. This specific waypoint decoder is chosen after performing preliminary baseline experiments on the LTF VA backbone model. When jointly training the fusion module and waypoint decoder, L1 loss is computed from predicted and ground truth trajectories to minimize with gradient descent.

5 Experiments

5.1 Data

Waymo Open Dataset - Vision-based End-to-End Driving (WOD-E2E)[14] is a public dataset specifically curated to represent long-tail scenarios that drivers rarely encounter, such as navigating through construction zone and maneuvering around unexpected obstacles, events that happen with a frequency of less than 0.003% in daily driving. Each sample/frame has 8 camera images, 16-timestep past trajectory states (4 seconds, 0.25 Hz), and routing intent (straight, left, or right). This project uses a small subset of WOD-E2E, where train/val/test splits are 1575/1580/1147; training set is small as the LTF backbone, though trained on OpenScene dataset, performs tasks in similar driving domain. In addition, only the 3 front-view camera images and the current timestep’s velocity and acceleration are used as input along with intent. The images have a resolution of 972 x 1079. To preserve as many pixels as possible while cropping the image to aspect ratio of 4:1 as required by the pretrained LTF, camera images are cropped by 300 at the top and 50 at the bottom. Every sample has 20-timestep (5 seconds, 0.25 Hz) future trajectory as the ground truth. All waypoints are in vehicle coordinates.



Figure 2: Examples of cropped, resized, stitched front-view camera images passed to LTF, Cosmos.

5.2 Evaluation method

Rater-Feedback Score (RFS) (Appendix A) is the primary evaluation metric, while Average Displacement Error (ADE) at $t = 3s$ and $5s$ are used as secondary evaluation metric. The average ADE across all timesteps is also reported, but the ADE at $t=3s, 5s$ entail more information on the deviation of predicted trajectory from ground truth trajectory over the course of future horizon. Not all samples in the original WOD-E2E dataset have rater specified trajectories. To ensure consistency across training and evaluation, the single ground truth trajectory in each sample is rated with score of 10, same as the evaluation method used by FROST-Drive[15]. Compared to the commonly used ADE, RFS is customized for more realistic autonomous driving scenarios as it is dependent on speed instead of position. The trust region is a form of tolerance that rewards any safe trajectory, not just the perfect imitation of the single ground truth path. RFS also decays exponentially for waypoints outside of trust regions, penalizing dangerously deviating trajectories.

5.3 Experimental details

The simplest baseline is a model composed of 1) 2-layer 1D CNN and a max pool to process past trajectory states, and 2) autoregressive waypoint decoder network composed of GRUs, referencing the original Transfuser[16]. The first stage of building the VLA model is to build a VA model. A few VA variants are experimented, including use DINOv3 or InternVL3 as the vision encoder. However, based on Table 2, it is difficult to train VLA model with the foundation vision encoder frozen on the small training subset to solve the complex, long-tail driving scenarios. This motivated the change of VA model to be the pretrained LTF adapted with a new waypoint decoder. After experimenting with different decoder configurations, as shown in Table 3, the best performing LTF based on RFS uses a 2-layer MLP with ReLU activation, mapping LTF backbone outputted trajectory query from dimension 256 to 2560 to 40, then reshaped to 20 2D waypoints. Next, Cosmos Reason 2 is experimented using different prompts to understand what it is capable of reasoning or answering (Appendix B). Then the integration of LTF and Cosmos is implemented to achieve a VLA model, illustrated in Figure 1. The experiments on VLA variants help investigate what language features extracted from Cosmos can effectively condition trajectory query output from LTF backbone. Each VLA variant is trained with the same configuration* to ensure fair comparison.

Optimizer	Batch Size	Precision	Learning Rate	Weight Decay	Epochs
AdamW	32	bfloat16	0.001	0.01	10

5.4 Results

Model	Inputs	RFS↑	ADE(all 5s)↓	ADE@3s↓	ADE@5s↓
CNN + GRU	past traj, intent	7.14	2.98	3.01	6.78
VLA-DINOv3	past traj, intent, imgs	7.24	2.54	2.60	6.33
VLA-InternVL3	past traj, intent, imgs	7.13	2.73	2.81	6.55
LTF + MLP	last status, intent, imgs	7.61	1.82	1.78	4.75

Table 2: Results of VA variants on validation set, trained with 5 epochs for preliminary study.

Waypoint Decoder	lr	RFS↑	ADE↓	ADE@3s↓	ADE@5s↓
2-layer MLP dim 1024	5e-4	7.52	1.79	1.80	4.83
2-layer MLP dim 1536	5e-4	7.56	1.75	1.75	4.74
2-layer MLP dim 2048	5e-4	7.60	1.73	1.71	4.64
2-layer MLP dim 2048	1e-3	7.68	1.69	1.68	4.53
2-layer MLP dim 2560	5e-4	7.57	1.75	1.75	4.75
2-layer MLP dim 2560	1e-3	7.72	1.70	1.68	4.47
GRU dim 64 with 2-layer MLP pre-proj	5e-4	7.42	1.94	1.83	5.06
GRU dim 128 with 1-layer MLP pre-proj	5e-4	7.25	1.96	1.83	5.15
GRU dim 128 with 2-layer MLP pre-proj	5e-4	7.52	2.01	1.91	5.08

Table 3: Results of LTF baseline variants evaluated on validation set, trained with 10 epochs.

Method	Val				Test			
	RFS	ADE	ADE3s	ADE5s	RFS	ADE	ADE3s	ADE5s
Baseline: CNN + GRU	7.31	2.32	2.35	6.33	7.18	2.28	2.30	6.24
Latent Transfuser + MLP	7.72	1.70	1.68	4.47	7.63	1.83	1.83	4.84
VLA LatentReason(NoGate)*	7.22	2.18	2.20	5.40	7.15	2.25	2.28	5.70
VLA LatentAnswer	7.55	1.96	1.94	4.83	7.54	2.05	2.01	5.21
VLA LatentAnswer(NoGate)	7.81	1.74	1.70	4.58	7.74	1.82	1.77	4.88
VLA TextAnswer(Sum)	7.70	1.71	1.69	4.51	7.62	1.87	1.84	5.02
VLA TextAnswer(FiLM)	7.81	1.70	1.66	4.43	7.75	1.83	1.79	4.89

Table 4: Results on down-scaled validation and test dataset with the core model variants experimented.

*LatentReason is trained for 5 epochs due to expensive CoT reasoning generation, all others with 10.

As shown from Table 4, the best performing VLA model is TextAnswer with FiLM, improved RFS on validation set by 6.8% compared to the simplest baseline, while reached RFS of 7.75 on test set compared to baseline 7.18. Compared to the strong baseline of LTF+MLP, TextAnswer(FiLM) also improved RFS from 7.63 to 7.75, suggesting that adding language feature conditioning does improve the trajectory predictions overall. The second best, LatentAnswer without gated residual, follows closely. This may suggest that latent features, if fused optimally, can condition trajectories in a similar way done by the tokenized and multi-hot encoded textual answer. TextAnswer with summation first

projects the 18-class multi-hot vector into a single 256-dim action embedding before adding up with trajectory query, which discards the compositional structure of the action set. In comparison, using FiLM to shift the trajectory query to any affine subspace conditioned on the action labels helps to direct the trajectory per combination of actions more precisely.

Since the reasoning chain in LatentReason is stochastic and verbose, many tokens are linguistic fillers rather than features carrying geometrically meaningful content. Also, pooling a max of 128 tokens into a single 256-dim vector is a severe information bottleneck, especially with no auxiliary loss used, the attention pooling weights learned from scratch have limited supervision signal. Although Cosmos could arrive at a more logically sound action plan after reasoning, the simplicity of the fusion method and the late stage of where this fusion is happening is limiting the information exchange across the two backbone models and seems to be adding unhelpful noises to LTF backbone, evident from the LatentReason RFS 7.22 being lower than LTF+MLP RFS 7.61 on val set, both trained for 5-epochs.

Based on Figure 3, showing the training L1 loss for each of the model variants trained for 10 epochs, both TextAnswer approaches converged faster than LatentAnswer approaches, with TestAnswer(FiLM) reaching a final epoch loss of 0.92, the lowest of all. Although the training curve of TestAnswer(FiLM) and LTF+MLP appears similar, TestAnswer(FiLM) performs better on test set, suggesting that it may have stronger generalization capability to unseen scenarios. Nonetheless, running Cosmos without CoT results in more randomness and lack of coherence in its action answer.

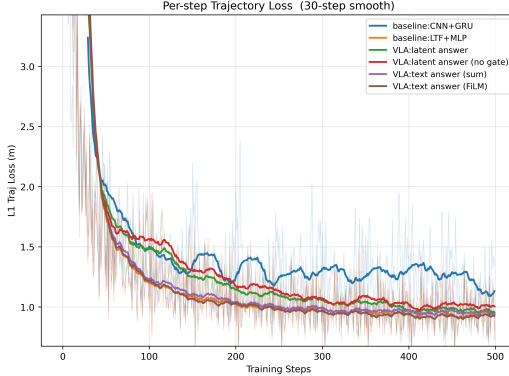


Figure 3: Training L1 loss for each of the core methods with evaluation results in Table 4.

Rank	Action Combination	Count	% of samples
1	keep straight + maintain speed	503	31.8%
2	keep straight + lane change right + maintain speed	358	22.7%
3	keep straight + lane change left + maintain speed	86	5.4%
4	sharp right + yield to vehicle	70	4.4%
5	keep straight + lane change left + accelerate + maintain speed	67	4.2%
6	sharp left + yield to vehicle	55	3.5%
7	keep straight + lane change left	54	3.4%
8	keep straight + maintain speed + yield to pedestrian	45	2.8%
9	keep straight + yield to pedestrian	43	2.7%
10	sharp left + yield to pedestrian	39	2.5%
11	sharp right + yield to pedestrian	37	2.3%
12	keep straight	30	1.9%
13	sharp left	29	1.8%
14	keep straight + lane change right + accelerate + maintain speed	27	1.7%
15	keep straight + maintain speed + yield to vehicle	16	1.0%
16	keep straight + lane change right	16	1.0%
17	keep straight + sharp left	11	0.7%
18	keep straight + maintain speed + stop	11	0.7%
19	keep straight + lane change right + accelerate	11	0.7%
20	keep straight + lane change right + yield to pedestrian	9	0.6%

Figure 4: Top 20 action combinations returned by TextAnswer[FiLM] Cosmos. Green rows are single action.

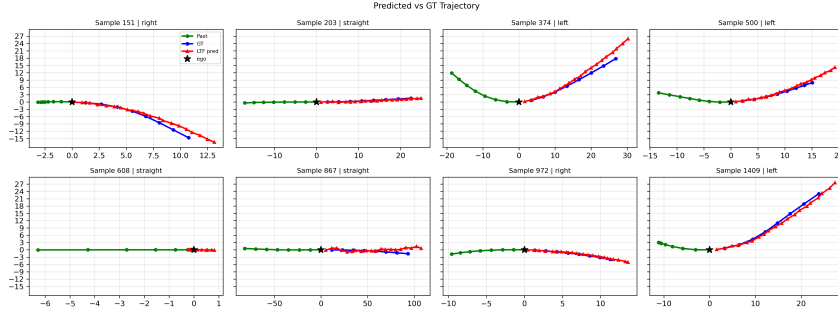


Figure 5: TextAnswer(FiLM) predicted trajectory vs ground truth on 8 different samples.

6 Analysis

The predicted trajectories shown in Figure 5 quite closely follow the ground truth. However, it is noticeable that the angle of left/right turn as well as how far to travel while turning are the challenging cases. As shown in Figure 4, Cosmos Reason 2 most of the times respond with multiple actions. Cosmos also generated self-conflicting actions, for example, "keep straight + lane change left + accelerate + maintain speed", which occurred on 67 samples in validation set. Although there are instances of hallucinated or obviously incorrect action plan, TextAnswer(FiLM) still achieved higher RFS than LTF+MLP, suggesting that this variant is able to not only amplify signal from language feature but also suppress it, an unintended positive effect.

As shown in action sensitivity plots in Figure 6, the spread of lines indicates that FiLM is taking an effect, especially noticeable in the different trajectory directions resulting from "sharp left" vs. "sharp right". However, sample (a) shows that using action embedding to direct trajectory can overshoot, while sample (b) shows that curvature is difficult to predict. In addition, for sample (a), LatentReason’s Cosmos generated answer after full CoT is "begin turning right while maintaining reduced speed to navigate the uphill incline safely", which is a much more accurate description of the driving scene compared to "sharp right, yield to pedestrian" generated without CoT. This shows that Cosmos Reason 2 is capable of creating action plan for long-tail scenarios with under 128 tokens of reasoning and answering. Future work could involve more investigation in how to use long sequence of latent feature encoding scene semantics and formulated reasoning from VLM to condition trajectories. Lastly, using MLP as waypoint decoder seems to produce unsmooth or physically unrealistic trajectory, additional post-processing methods such as spline smoothing can be applied or train the GRU waypoint decoder but for longer time to fully learn the change in waypoints to then generate waypoints autoregressively.

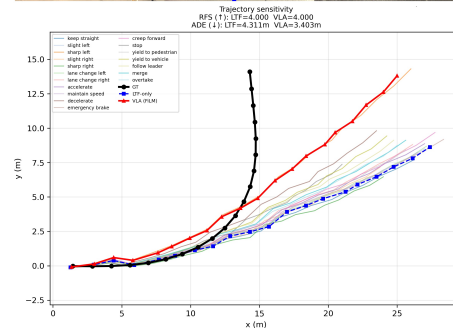
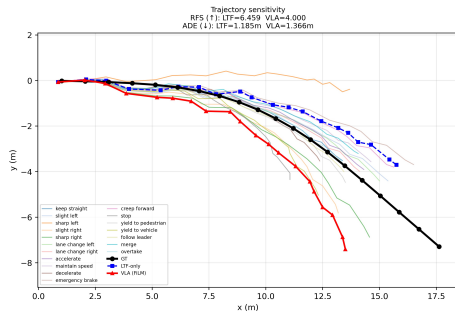


Figure 6: Failure cases of TextAnswer with FiLM conditioning. Top images: predicted (red) vs ground truth (blue) waypoints projected over image. Bottom: Each colored line in the trajectory sensitivity plots indicate the potential trajectory when activated with that single action alone.

This project demonstrates how language features can be extracted to influence trajectory prediction as one way of bridging VLMs with VA models. The VLA variant TestAnswer(FiLM), using multi-hot encoded Cosmos actions as signals to condition trajectory query from LTF backbone, achieved the highest RFS. Yet, the challenge of such multi-modal alignment and cross-model integration is again emphasized: it is difficult to transform high-level semantic hidden states in language space to executable and physically consistent trajectories. If to take full advantage of CoT to arrive at a more reasonable answer, the autoregressive generation of such reasoning significantly slows down the training and inference time. Even if the full reasoning is generated, which latent features to extract to condition trajectory is another challenge: from which tokens and from which transformer layer(s). In addition, LTF extract BEV features while Cosmos extract image patch tokens, which adds integration complexity even if they use the same image. Future work can involve building a common vision encoder followed by two branches, one for vision-action agreement and another for vision-language, then apply conditioning techniques.

References

- [1] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S. Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling, 2023.
- [2] Inhwon Bae, Junoh Lee, and Hae-Gon Jeon. Can language beat numerical regression? language-based multimodal trajectory prediction, 2024.
- [3] Zhengxing Lan, Hongbo Li, Lingshan Liu, Bo Fan, Yisheng Lv, Yilong Ren, and Zhiyong Cui. Traj-llm: A new exploration for empowering trajectory prediction with pre-trained large language models, 2024.
- [4] Xiaoji Zheng, Lixiu Wu, Zhijie Yan, Yuanrong Tang, Hao Zhao, Chen Zhong, Bokui Chen, and Jiangtao Gong. Large language models powered context-aware motion prediction in autonomous driving, 2024.
- [5] Haicheng Liao, Hanlin Kong, Bonan Wang, Chengyue Wang, Wang Ye, Zhengbing He, Chengzhong Xu, and Zhenning Li. Cot-drive: Efficient motion forecasting for autonomous driving with llms and chain-of-thought prompting, 2025.
- [6] Yuechen Luo, Fang Li, Shaoqing Xu, Zhiyi Lai, Lei Yang, Qimao Chen, Ziang Luo, Zixun Xie, Shengyin Jiang, Jiaxin Liu, Long Chen, Bing Wang, and Zhi xin Yang. Adathinkdrive: Adaptive thinking via reinforcement learning for autonomous driving, 2025.
- [7] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025.
- [8] NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, Daniel Dworakowski, Jiaojiao Fan, Michele Fenzi, Francesco Ferroni, Sanja Fidler, Dieter Fox, Songwei Ge, Yunhao Ge, Jinwei Gu, Siddharth Gururani, Ethan He, Jiahui Huang, Jacob Huffman, Pooya Jannaty, Jingyi Jin, Seung Wook Kim, Gergely Klár, Grace Lam, Shiyi Lan, Laura Leal-Taixe, Anqi Li, Zhaoshuo Li, Chen-Hsuan Lin, Tsung-Yi Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Arsalan Mousavian, Seungjun Nah, Sriharsha Niverty, David Page, Despoina Paschalidou, Zeeshan Patel, Lindsey Pavao, Morteza Ramezani, Fitsum Reda, Xiaowei Ren, Vasanth Rao Naik Sabavat, Ed Schmerling, Stella Shi, Bartosz Stefaniak, Shitao Tang, Lyne Tchammi, Przemek Tredak, Wei-Cheng Tseng, Jibin Varghese, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Xinyue Wei, Jay Zhangjie Wu, Jiashu Xu, Wei Yang, Lin Yen-Chen, Xiaohui Zeng, Yu Zeng, Jing Zhang, Qinsheng Zhang, Yuxuan Zhang, Qingqing Zhao, and Artur Zolkowski. Cosmos world foundation model platform for physical ai, 2025.
- [9] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibao Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025.
- [10] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking, 2024.
- [11] OpenScene Contributors. Openscene: The largest up-to-date 3d occupancy prediction benchmark in autonomous driving. <https://github.com/OpenDriveLab/OpenScene>, 2023.

- [12] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles, 2022.
- [13] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer, 2017.
- [14] Runsheng Xu, Hubert Lin, Wonseok Jeon, Hao Feng, Yuliang Zou, Liting Sun, John Gorman, Ekaterina Tolstaya, Sarah Tang, Brandyn White, Ben Sapp, Mingxing Tan, Jyh-Jing Hwang, and Dragomir Anguelov. Wod-e2e: Waymo open dataset for end-to-end driving in challenging long-tail scenarios, 2025.
- [15] Zeyu Dong, Yimin Zhu, Yu Wu, and Yu Sun. Frost-drive: Scalable and efficient end-to-end driving with a frozen vision encoder, 2026.
- [16] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving, 2022.

A Rater Feedback Score

Rater Feedback Score (RFS) is the primary evaluation metrics, the higher the score the better performance of the model, with a max of 10. In the WOD-E2E dataset, at specific moments within each driving segment, three distinct 5-second future trajectories are rated by experts on a scale of 0 to 10, where 0 represents the worst trajectory and 10 the best trajectory. At least one of the three rated trajectory has a score higher than 6. It is important to note that not every frame has a expert rated trajectories. With the given preferred rater specified trajectories and model predicted trajectory, a trust region is defined to be the region within a given lateral and longitudinal threshold of the rater specified trajectory at a given time T ($T = 3s, 5s$). If the predicted trajectory is within the trust region, it is given the rater’s score of the corresponding rater specified trajectory. Both the time step of waypoint and the velocity at the time step in lateral and longitudinal directions affect the size of this trust region.

Time-based longitudinal and lateral thresholds:

$$t = 3s : \hat{\tau}_{lat} = 1.0, \hat{\tau}_{lng} = 4.0$$

$$t = 5s : \hat{\tau}_{lat} = 1.8, \hat{\tau}_{lng} = 7.2$$

Speed-based scaling, where the speed v is the initial speed of the rater specified trajectory:

$$\text{scale}(v) = \begin{cases} 0.5 & \text{if } v \leq 1.4m/s \\ 0.5 + 0.5 \times \frac{v-1.4}{11-1.4} & \text{if } 1.4m/s \leq v < 11m/s \\ 1 & \text{if } v \geq 11m/s \end{cases} \quad (1)$$

Final threshold in terms of both time and initial speed:

$$\hat{\tau}_{lat}(T, v) = \text{scale}(v) \times \hat{\tau}_{lat}(T) \quad (2)$$

$$\hat{\tau}_{lng}(T, v) = \text{scale}(v) \times \hat{\tau}_{lng}(T) \quad (3)$$

If the predicted waypoint $\hat{w}_t$ is inside the trust region of actual waypoint w_t , the full score 10 is assigned. Otherwise, a score exponentially lower is assigned, calculated as follows:

$$\text{RFS}(w_t, \hat{w}_t) = \begin{cases} \bar{r} & \text{if } \Delta \leq 1 \\ \bar{r} \times 0.1^{\Delta-1} & \text{otherwise} \end{cases} \quad (4)$$

$\Delta = \max \left\{ \frac{\Delta_{lat,t}}{\tau_{lat,t}}, \frac{\Delta_{lng,t}}{\tau_{lng,t}} \right\}$, where $\Delta_{lat,t}$ is the lateral distance error and $\Delta_{lng,t}$ the longitudinal distance error between the predicted waypoint and corresponding waypoint in the rater specified trajectory at time t . $\bar{r}$ is the score of the rater specified trajectory closest to predicted trajectory.

B Cosmos Reason 2 Reasoning Capability Demonstration

This section shows the reasoning capability of Cosmos Reason 2 given the front-view camera images and an instruction prompt.

Prompt:

```
Analyze the driving scene from three front-view camera images: front-  
left, front, and front-right (given in this order).  
Navigation intent: **{intent}**  
  
Reason through:  
1. Critical objects: identify a maximum of three unique objects that  
   affect driving, such as vehicles, pedestrians, cyclists, and  
   construction objects.  
2. Scene: road type, road structure, number of lanes, markings,  
   lighting  
3. Maneuver: what action, speed adjustment, steering direction matches  
   the intent and is safe given the critical objects and scene  
  
Return answer in the following json format:  
{  
  "critical_objects": "<a list of maximum three unique critical  
    objects>",  
  "scene_description": "<one short sentence>",  
  "maneuver_plan": "<once short sentence>"  
}
```

Example: an accurate depiction of the front-view camera image on the left in Figure 2:

```
{  
  "critical_objects": [  
    "building",  
    "pole",  
    "vehicle"  
  ],  
  "scene_description": "A two-lane street with dashed yellow lines  
    and sidewalks, bordered by palm trees and lampposts, featuring  
    a solitary vehicle in the distance.",  
  "maneuver_plan": "Slow down and steer left to navigate the turn  
    while maintaining a safe distance from the pole and vehicle."  
}
```

Example: an unsafe maneuver plan, given the front-view camera image on the right in Figure 2:

```
{  
  "critical_objects": [  
    "traffic light",  
    "construction barricades",  
    "traffic cones"  
  ],  
  "scene_description": "At night, the street is deserted with no  
    vehicles or pedestrians, featuring a single lane in each  
    direction, marked lanes, and illuminated by streetlights.  
    Traffic cones and barricades are strategically placed along  
    the roadway.",  
  "maneuver_plan": "Drive straight at a steady pace while remaining  
    vigilant for any sudden developments."  
}
```

The initial attempt of building the VLA is to use Gemma 3 270M text-only language model with LoRA fine-tuning to predict the Cosmos Reason 2 generated reasoning annotation for each of the samples. However, the training was very unstable, likely due to the difficulty to align not only language-action modalities but also language-vision modalities. Therefore, this approach was discarded and project pivoted to using Cosmos Reason directly inside the VLA model.