Modeling the Margins: Edge-Aware E2E Driving

Kevin Selig
Stanford
kselig1@stanford.edu

Abstract

This project investigates the use of multimodal vision-language models for end-to-end (E2E) autonomous driving, with a focus on improving performance in complex, edge-case scenarios. The open-source code, OpenEMMA, is applied to the nuScenes mini dataset. OpenEMMA is an open-source E2E driving framework that integrates large vision-language models (MLLMs) with YOLO3D object detection. The framework takes as input front-camera RGB video and ego pose data to predict future vehicle trajectories and generate scene-level driving intent summaries. Additionally, a ResNet18-based regressor is fine-tuned for 3D bounding box estimation using KITTI data.

The experiments compare the performance of different MLLMs, showing that GPT-4o paired with YOLO3D achieves the lowest trajectory prediction error, albeit with higher computational cost. Attempts to retrain YOLO3D components led to marginal improvement, highlighting the challenge of optimizing spatial reasoning components with limited data. This work establishes a scalable pipeline for E2E training and evaluation, and provides insight into the trade-offs between model accuracy, interpretability, and resource efficiency. Future work includes scaling to larger datasets like the Waymo E2E dataset, augmenting training data and incorporating additional vision based modalities such as lidar or radar.

1. Introduction

Autonomous driving systems must operate safely not only in routine scenarios but also in rare, long-tail edge cases that challenge model generalization. The problem investigated in this project is how accurately a vision-based, end-to-end (E2E) autonomous driving model can predict driving behavior, especially in such edge cases. End-to-end driving refers to learning a direct mapping from sensor inputs — such as raw images or video streams — to driving outputs like vehicle trajectory, control signals, or intent. While the E2E paradigm has been explored for over a decade, recent advancements in vision-language and foundation models warrant a fresh investigation into their effectiveness, particularly under data-constrained conditions.

A key inspiration for this project is the Waymo End-to-End Driving Challenge [20]. Given the scale of the dataset and the author's relative newness to the field, this project sets simplified goals that align with and support the broader objectives of the challenge. The goals of this project are (1) to evaluate how newer multimodal vision-language architectures perform in E2E driving scenarios, and (2) to prototype a simplified training and evaluation pipeline suitable for future application to large-scale datasets.

In this project, OpenEMMA, a multi-modal vision based model [21] is applied to a simplified version of the NuScenes mini data set [2]. The primary inputs used are:

1. Front camera RGB video frames
2. Ego pose history
3. Pretrained MLLM and 3D object detection weights

Outputs of the model are:

1. Future vehicle trajectory
2. Scene images with 3D bounding boxes
3. Text based summaries with driving intent and decision logs
4. Evaluation metrics comparing trajectory accuracy

Focusing on constrained inputs allows for a clearer evaluation of model components and their contributions to overall driving behavior. This work helps lay the groundwork for scaling to larger, more diverse E2E datasets by establishing a training and evaluation pipeline. Through this prototype, trade-offs are investigated between interpretability, accuracy, and computational cost.

1.1. Related Work

A review of the latest challenges and modeling approaches in the end-to-end autonomous driving community is provided to add context to the challenge pursued in this project. After review, the main obstacles found are data quality, model generalisation, and interpretability. Li et al. present a data-centric roadmap that views closed-loop dataset integration, hardware development to process large datasets and personalization as primary areas to further improve performance [10]. Complementing this, Chen et al. catalogue 270 + E2E papers and emphasize focus areas of data quality, causal confusion, and foundation-model integration [3]. Earlier analysis by Chib & Singh [4] and Singh [19] note the historical shift from modular pipelines to imitation and reinforcement-based E2E methods. They emphasize the need for refined safety constraints and transparent decision making. Recent reviews highlight the promise of large language models (LLMs) and vision language models (VLMs) integration for improving interpretability and user alignment, while also acknowledging the added uncertainty and complexity these models bring to E2E driving.

Recent closed and open-source autonomous driving models show a trend toward more multimodal learning frameworks. The following trends characterize the latest modeling strategies:

1. **Multimodal, language-aligned architectures.** There is a growing emphasis on framing driving as a multi-task problem where perception, planning, and reasoning are handled within a unified model. Language serves as a common interface to align diverse modalities and facilitate supervision across tasks.

2. **Vision–language fusion for contextual awareness.** Models aim to mimic human-like attention mechanisms and enhance decision-making in complex or ambiguous driving scenes by integrating textual priors or language-derived features into spatial representations (e.g., BEV maps).

3. **LLM-driven reasoning and explanation.** Large language models are increasingly used to supervise or augment policy learning, enabling systems to generate interpretable rationales alongside control outputs. This improves transparency and interpretability.

1.2. Model Architectures

A survey of autonomous driving models was performed to identify architectures that could be applied or adapted to the Waymo driving challenge. Identified models are summarized below, beginning with closed-source models followed by open-source alternatives.

EMMA leverages Gemini to cast all inputs and tasks as text, achieving high motion-planning accuracy on nuScenes and 3-D detection on the Waymo Open Dataset (WOD). Co-training across planning, perception, and mapping yields further gains. Limitations include computational expense and does not incorporate lidar or radar. [9]

VLM-E2E focuses on attention semantics and modality balance using a BEV-Text learnable weighted fusion strategy, a spatio-temporal module to ensure temporal coherence in dynamic scenes and a probabilistic future prediction module with attention guided trajectory refinement. Radar and lidar integration is planned for future development. [12]

DriveGPT4 applies Llama2 as a backbone to processes monocular video, answers human queries, and produces low-level control predictions. A bespoke tunning data set was created for training. [22]

Here are open-source alternatives:

OpenEMMA built by Texas A&M University (TAMU) implements an open-sourced version of EMMA with pre-trained MLLMs and a fine-tuned YOLO3D model for object detection, achieving generalizable and robust accuracy. The model shows work is still needed in MLLMs to bridge the gap in spatial accuracy. [21]

TransFuser introduces a multi-modal fusion transformer that combines resnet and transformer architectures to interpret single view images and lidar BEV images, reducing collision rates by 76 % in adversarial scenarios compared to geometry-based fusion [15]

The literature indicates a shift from modular model improvements toward knowledge and data-centric paradigms. Applying language models as backbones in autonomous driving architectures adds world knowledge, reasoning, and interpretability into driving policies. Unresolved issues still remain such as: scalable closed-loop evaluation, mitigating causal confusion, and aligning with safety-critical metrics [10, 3] Future E2E development appears to be focusing on rigorous data-centric pipelines with foundation-model reasoning to deliver robust and interpretable E2E driving systems.

2. Methods

The approach taken to explore end-to-end autonomous driving models involved experimenting with OpenEMMA and training a subset of inputs to better understand the framework's structure and performance. The following figure illustrates the primary components of the OpenEMMA architecture:

Main inputs to OpenEMMA are front camera images and 5 second ego pose information. As described previously, OpenEMMA applies pre-trained multi-modal large

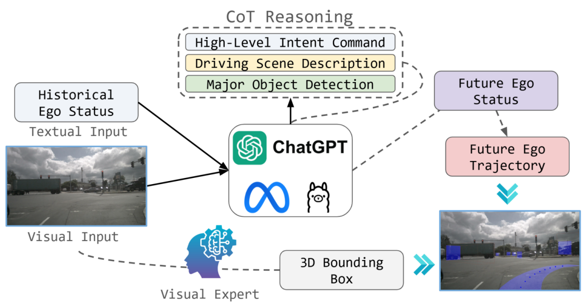

Figure 1. OpenEMMA framework [21]

language models (MLLMs). Chain of thought reasoning is employed to guide the models in generating detailed descriptions of critical objects, behavioral insights, and driving decisions. The MLLMS are prompted to generate speed and curvature vectors (vehicle turning rate). The four MLLM options that are integrated into OpenEMMA are:

- Qwen2-VL-7B-Instruct [16]
- LLaVA-1.6-Mistral-7B [11]
- LLAMA-3.2-11B-Vision-Instruct [13]
- GPT-4o [14]

Qwen, LLaVa and LLama are all vision based LLMs but the OpenEMMA group found there were limitations with spatial reasoning. To address this they integrated YOLO3D [1] with GPT-4o to improve perception capabilities. YOLO3D performs 3D object detection, creating 3D bounding boxes on identifed objects. Three different weights files are used in the YOLO3D pipeline to construct and refine the 3D bounding boxes. The purpose and name of the weight file is provided below for reference:

- 2D bounding box (yolo11n_nuimages.pt)
- Initial 3D bounding box (yolov5s.pt)
- 3D bounding box fine tuning (resnet18.pkl)

In the OpenEMMA study [21] the YOLO3D 2D bounding box identification was trained on the nuimages dataset [2], a subset of the nuScenes dataset. This created the yolo11n_nuimages.pt weights. They kept the yolov5s.pt weights which create the initial 3D bounding boxes. The yolo3D repository in OpenEMMA allows the options to train a regressor model which fine-tunes the 3D bounding box orientation and size.

2.1. Baseline Method

The baseline evaluation involves applying OpenEMMA with its integrated MLLMs and YOLO3D module using all pretrained weights. Due to configuration constraints, the Llama model was excluded from the analysis. The baseline comparisons focuses on:

- Average displacement error (L2 distance) between predicted and ground-truth trajectories,
- Model interpretability via natural language reasoning outputs,
- Computational efficiency, including runtime and cost.

2.2. Further training

Proceeding beyond the baseline configuration, a regression model is fine-tuned as a part of the YOLO3D pipeline. The goal of the model is to be able to accurately predict the 3D dimensions and orientation of objects from 2D images. The regression model training is trained on a subset of KITTI training data [7] which provides monocular RGB images, camera calibration files and annotated 3D bounding boxes.

YOLO3D provides both ResNet18 [8] and VGG11 [18] models for regression training. ResNet18 was chosen over VGG11 for the regression training due to its improved training stability, lower parameter count, and stronger generalization on small datasets like KITTI. The residual connections in ResNet18 enable more efficient gradient flow which leads to better performance under limited data conditions.

The loss function used is adapted from [1] and combines 3 components:

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{dim}} + \mathcal{L}_{\text{conf}} + w \cdot \mathcal{L}_{\text{orient}},$$

where:

- $\mathcal{L}_{\text{dim}}$ is a mean square error loss over object dimensions (height, width, length)
- $\mathcal{L}_{\text{conf}}$ is a cross-entropy loss on confidence scores
- $\mathcal{L}_{\text{orient}}$ is an orientation loss

Default weights are set to $\alpha = 0.6$ and $w = 0.4$ which were likely created to balance dimensional accuracy and orientation quality.

Training is conducted using stochastic gradient descent (SGD) with momentum. Loss metrics and model checkpoints are logged using Comet.ml [5]. After training, the model is evaluated in inference mode by projecting predicted 3D bounding boxes back onto the original images for qualitative comparison with outputs applying the pretrained regressor weights.

2.3. Code Development

This project primarily utilized the existing OpenEMMA codebase [21], with minor modifications made to be able to execute the code. Most of the development effort focused on setting up the runtime environment and constructing the data pipeline necessary for training and evaluation. Additionally, GPU Finder [6] proved useful for identifying available computing resources.

3. Dataset

One of the datasets used here is the nuScenes data set mini [2] for running and testing OpenEMMA. The nuScenes dataset containts various car driving scenarios. The dataset contains 10 difference scenes captured at 20 fps for 20 seconds. The data set consists of:

1. ego pose
2. 6 camera views
3. radar and lidar images

OpenEmma only relies on front camera views and historical ego pose for predicting the next 5 seconds. so the rest of the data set was not used. An example front camera driving scene is shown in Figure 2. Many of the scenes like this one contain maintenance or utility operation zones to avoid.

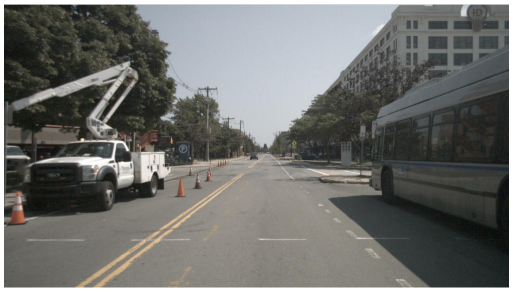
Figure 2. Example NuScenes Front Camera Input [2]

The KITTI dataset [7]was used for ResNet18 training. The KITTI data set contains 12 GB of mono RGB images (7500), camera calibrations files per image, and training labels and boundaries. An example is shown in Figure 3. Only a small subset ranging from 100 - 1000 images were used in training while another 100 were used for testing.

As the long term intent is to analyze the Waymo E2E dataset [17], the dataset is described here for reference. The full data set consists of approximately 4k segments with approximately 10 difference driving scenarios. Training and validation data spans 20 seconds and test data spans 12 seconds. Each segment contains:

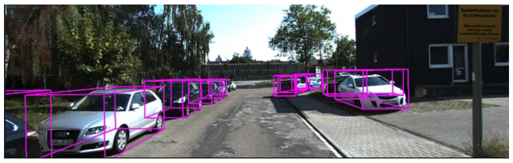
Figure 3. Example KITTI Scene with Boundaries [7]

1. vehicle trajectory
2. vehicle velocity and acceleration
3. command direction
4. 10 Hz camera images provide 360 degree coverage with 8 views
5. rater feedback score for reviewing acceptable driving scenarios

The full data set includes a total of about 600 files ranging from 2 - 4 GB each which is approximately 1.8 terabytes.

4. Experiments/Results/Discussion

The OpenEMMA model was evaluated across several different configurations using the nuScenes dataset. In addition to testing various MLLM backbones, further training was conducted on the YOLO3D model with the KITTI dataset. The results reveal how different model choices and training strategies impact trajectory prediction and at what analysis cost.

Results in this section are plotted from several different driving scenes. Example scene descriptions of those analyzed are provided here from the nuScenes data set for context. The examples are diverse yet typical of many driving scenarios. Given more time a good avenue to explore would be model performance on very end-tail scenarios.

”name”: ”scene-0103”, ”description”: ”Many peds right, wait for turning car, long bike rack left, cyclist”

”name”: ”scene-1077”, ”description”: ”Night, big street, bus stop, high speed, construction vehicle”

Within OpenEMMA three different models were run on the nuScene mini dataset that produced trajectory predictions. A comparison of L2 distance between the predicted trajectory and ground truth is shown in Table 1 for scene 103. (The table is at the top of page 6) Average displacement error is calculated as follows With $\hat{P}_t$ reresenting predicted trajectory and P_t representing ground truth.

$$\text{ADE} = \frac{1}{T}\sum_{t=1}^{T} \|\hat{\mathbf{p}}_t - \mathbf{p}_t\|_2$$

While only a single scene is not sufficient to draw conclusions about overall model accuracy, in this instance GPT-4o appears to perform the best. This supports OpenEMMA's findings that integrating YOLO3D can improve trajectory prediction relative to other MLLM backbones. It is worth noting that GPT-4o incurs a relatively high cost — approximately $2 per scene which may become significant when running large-scale simulations.

Two example scene outputs are shown with trajectories overlaid for Qwen and GPT-4o in Fig 4 5. The outputs show the Qwen trajectory veering towards a parked car. The trajectory does correct back to center but in comparison the GPT-4o + YOLO3D with bounding boxes appears superior. The heightened spatial awareness results in the trajectory down the center of the street.

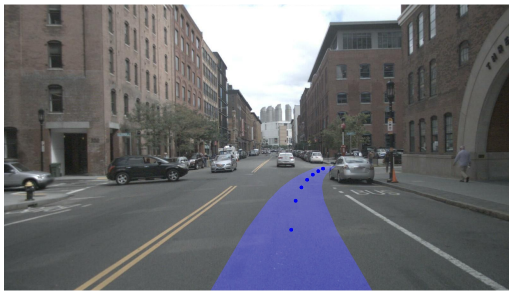

Figure 4. Qwen Scene103 FrontCam with Trajectory

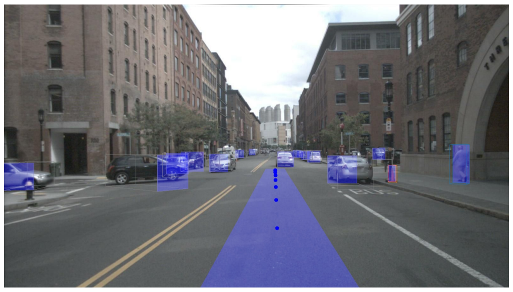

Figure 5. GPT-4o Scene103 FrontCam with Trajectory

For the above scene example, log outputs from the models are shown below which are produced by set prompts in OpenEMMA. Qwen provides an inaccurate description of the lane markings while GPT-4o identifies both the center double yellow and bus lane.

Qwen: **Lane Markings**: The ego car is still in the left lane, which is marked for a left turn. The lane markings and the direction of the car suggest that the intent to turn left remains unchanged.

GPT-4o: **Lane Markings:** - The lane markings are clearly visible. The double yellow centerlines separate the two directions of traffic, and white dashed lines guide drivers within their lanes. - A "BUS" lane is visible on the right-hand side, suggesting it is designated for bus traffic or other restricted vehicles.

4.1. YOLO3D training

As mentioned previously YOLO3D relies on 3 different weights sets in creating 3D bounding boxes. Within OpenEMMA there is an option to train the regression model weights that are meant to provide bounding box size and orientation. A baseline case was run using the default Resnet18 pretrained weights on the KITTI test data set for later qualitative comparison.

The regression model was then trained on a subset of the KITTI training data with some cases using the pretrained weights as a starting point. The training code was not set up to split the training data into testing and validation sets so as an intermediary means the test data was used as a validation set for comparison, applying the trained weights in YOLO3D inference mode. In the future the data will be more appropriately split between training/validation and test. The hyperparameters that were settled upon were as follows:

1. learning rate: 1e-4
2. batch size: 32
3. momentum: 0.9
4. epoch: 30
5. α: 0.6
6. w: 0.4

These are close to the default hyperparemters in OpenEMMA. Learning rate was adjusted but other values were found to make the training worse. Number of epochs was increased from 10 to 100 which appears to have slightly improved the loss in some cases but also largely increased training time. Training loss seemed to mostly settle by epic 30. α and w are hyperparamters specific to the YOLO3D loss function and were kept as defaults.

Example training runs are shown in Figure 6 where some were successful and others much less so, The results were from playing around with different learning rates, dataset sizes, momentum and epoch lengths.

Example training losses, on the slightly better end, are shown in Figure 7. The purple trace came from first applying pretrained weights and then training on a small KITTI data set out to 100 epochs. Training loss showed a quick improvement and then stayed at a relatively low level although never fully stabilized. The final loss was similar to non-pretrained data on a larger training set size shown in blue. The blue trace showed good initial training loss and then had a brief increase before coming back down. Given

Table 1. **Average Displacement Error (ADE) for Scene 103**

Model	L2 (m) 1s	L2 (m) 3s	L2 (m) avg	hrs/scene
LLaVA-1.6-Mistral-7B	2.39	5.11	3.64	0.25
Qwen2-VL-7B-Instruct	3.40	6.16	4.54	1.25
GPT-4o	1.02	3.83	2.47	0.5

further time, methods would be explored to improve training loss convergence such as by expanding the training set and applying image augmentation techniques.

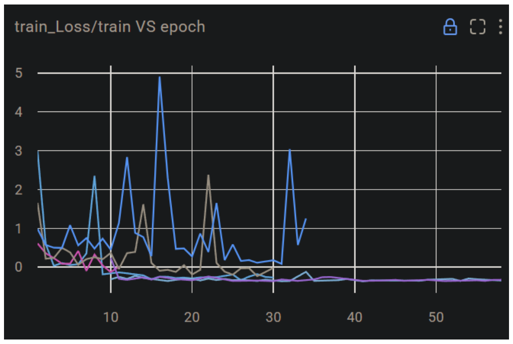

Figure 6. YOLO3D Resnet18 Training Run Loss Examples

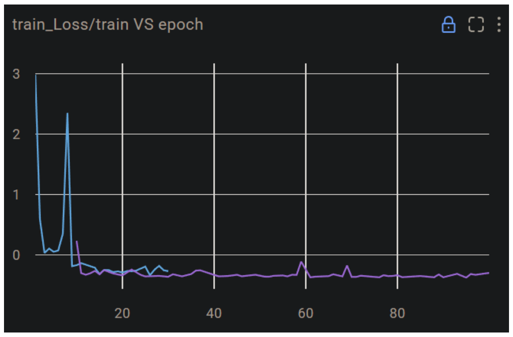

Figure 7. Slightly Better YOLO3D Resnet18 Training Run Loss Examples. The blue trace uses a larger training data set and the purple trace starts from pre-trained weights

Figure 8 shows results running YOLO3D in inference mode with the pre-trained (left) and trained (right) weights on the KITTI test set. The image on the left shows a bounding box extending beyond the vehicle's trunk while the right shows a much more centered bounding box. This was a more extreme example as throughout most of the test scenes, the bounding boxes of the pre-trained vs trained weights were very similar. Overall the training showed a very small change in bounding box shape and a deeper dive would be needed to confirm an overall improvement to the bounding accuracy. As the training did not show a strong improvment and due to time constraints a further test case with OpenEMMA and the further trained YOLO3D weights was not run.

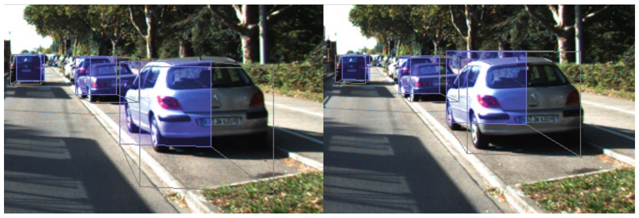
Figure 8. YOLO3D inference runs pretrained (left) and trained (right)

The results successfully demonstrate a functional pipeline for analyzing autonomous driving data and fine-tuning specific model components. Scaling this approach to a larger dataset, such as the full Waymo E2E challenge, would require careful consideration of model architecture and training strategy as these choices can significantly impact time and computational cost. While GPT-4o produced the most robust trajectory predictions, its higher per-scene cost could become prohibitive at scale.

5. Conclusion/Future Work

This project explored the performance of multimodal vision-language models applied to end-to-end autonomous driving scenarios using the OpenEMMA framework. By leveraging pretrained MLLMs and integrating YOLO3D for 3D object detection, the system was able to generate trajectory predictions, scene-level textual reasoning, and visualizations of driving intent. Among the models evaluated, GPT-4o yielded the lowest average displacement error (ADE), suggesting that the addition of spatially grounded bounding box reasoning from YOLO3D improves prediction quality. Using GPT-4o however came with an additional monetary cost per scene, indicating a trade-off between interpretability, accuracy, and scalability.

Training a ResNet18-based 3D bounding box regressor on a subset of the KITTI dataset yielded only marginal improvements over pretrained weights and training loss was also somewhat noisy. This suggests there is room to improve training through using more diverse training data, improved hyperparameter tuning, augmentation or all of the above. With the increase in accuracy from incorporating YOLO3D, results suggest careful tuning of supporting models within the OpenEMMA framework can play a critical role in enhancing overall performance.

Given more time, resources, and compute, future work would focus on three main directions: (1) scaling the pipeline to larger, more realistic datasets like the Waymo E2E Challenge to better assess generalization, (2) experimenting with advanced augmentation techniques to improve the regressor's robustness, and (3) incorporating lidar, radar or BEV representations that could enhance spatial reasoning in more occluded or cluttered environments.

6. Contributions Acknowledgments

The following codes were used in development of this project:

- OpenEMMA [21] https://github.com/taco-group/OpenEMMA
- gpu-finder [6] https://github.com/doitintl/gpu-finder

OpenEmma was the foundational code used in this project. GPU finder was applied as a very useful tool in searching for compute resources on Google Cloud Platform.

References

[1] W. Ali, S. Abdelkarim, M. Zahran, M. Zidan, and A. E. Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. *arXiv preprint arXiv:1808.02350*, 2018.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

[3] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li. End-to-end autonomous driving: Challenges and frontiers, 2024.

[4] P. S. Chib and P. Singh. Recent advancements in end-to-end autonomous driving using deep learning: A survey. *arXiv preprint*, 2023. arXiv:2301.xxxxx.

[5] Comet. Comet.ml: Experiment tracking for machine learning. `https://www.comet.com`, 2024. Accessed: 2025-06-04.

[6] doitintl. Gpu finder. `https://github.com/doitintl/gpu-finder`, 2024. Accessed: 2025-06-04.

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[9] J.-J. Hwang, R. Xu, H. Lin, W.-C. Hung, J. Ji, K. Choi, D. Huang, T. He, P. Covington, B. Sapp, Y. Zhou, J. Guo, D. Anguelov, and M. Tan. Emma: End-to-end multimodal model for autonomous driving, 2024.

[10] L. Li, W. Shao, W. Dong, Y. Tian, Q. Zhang, K. Yang, and W. Zhang. Data-centric evolution in autonomous driving: A comprehensive survey of big data system, data mining, and closed-loop technologies, 2024.

[11] H. Liu, C. Li, Q. Wu, Y. Lin, Y. J. Lee, and J. Gao. Llava: Large language and vision assistant, 2023.

[12] P. Liu, H. Liu, H. Liu, X. Liu, J. Ni, and J. Ma. Vlm-e2e: Enhancing end-to-end autonomous driving with multimodal driver attention fusion. 2025.

[13] Meta AI. Llama 3: Open foundation and instruction-tuned language models. `https://ai.meta.com/llama/`, 2024. Accessed: 2025-06-04.

[14] OpenAI. Gpt-4o technical report. `https://openai.com/index/gpt-4o`, 2024. Accessed: 2025-06-04.

[15] A. Prakash, K. Chitta, and A. Geiger. Multi-modal fusion transformer for end-to-end autonomous driving, 2021.

[16] Qwen Team, Alibaba DAMO Academy. Qwen-vl: A versatile vision-language model with in-context learning. `https://huggingface.co/Qwen/Qwen-VL`, 2024. Accessed: 2025-06-04.

[17] W. Research. Waymo open dataset: End-to-end driving. `https://waymo.com/open/data/e2e`, 2025.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1409.1556.

[19] A. Singh. End-to-end autonomous driving using deep learning: A systematic review. *arXiv preprint*, 2023. arXiv:2302.xxxxx.

[20] Waymo Team. Waymo e2e challenge (2025), 2025. `https://waymo.com/open/challenges/2025/e2e-driving/`.

[21] S. Xing, C. Qian, Y. Wang, H. Hua, K. Tian, Y. Zhou, and Z. Tu. Openemma: Open-source multimodal model for end-to-end autonomous driving, 2025.

[22] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K.-Y. K. Wong, Z. Li, and H. Zhao. Drivegpt4: Interpretable end-to-end autonomous driving via large language models. *arXiv preprint*, 2023. arXiv:2304.xxxxx.