UniMM: Technical Report for Waymo Open Sim Agents Challenge 2025

Longzhong Lin, Xuewu Lin, Kechun Xu, Haojian Lu, Lichao Huang, Rong Xiong, Yue Wang

***Abstract*—Simulation plays a crucial role in assessing autonomous driving systems, where the generation of realistic multi-agent behaviors is a key aspect. In multi-agent simulation, the primary challenges include behavioral multimodality and closed-loop distributional shifts. This technical report presents UniMM, a unified mixture model framework for generating multimodal agent behaviors. Crucially, we propose a closed-loop sample generation approach tailored for mixture models to mitigate distributional shifts. To extend the benefits of closed-loop samples across a broader range of mixture models, we further address the shortcut learning and off-policy learning issues. UniMM achieves top-tier performance in the Waymo Open Sim Agents Challenge (WOSAC) 2025. Complete details and full experimental results can be found on the project webpage: https://longzhong-lin.github.io/unimm-webpage.**

I. Introduction

Simulation facilitates the assessment of autonomous driving systems in a safe, controllable, and cost-effective manner. One key to narrowing the gap between simulated and real-world environments is the generation of human-like multi-agent behaviors. The main challenges in achieving such realism involve capturing the *multimodality* of agent behaviors and addressing *distributional shifts* in closed-loop rollouts.

Recovering the multimodality of agent behaviors has been extensively studied in motion prediction, where state-of-the-art methods [1]–[5] predominantly employ mixture models. Consequently, analogous models have been adopted for multi-agent simulation [6]–[8], typically incorporating a winner-takes-all continuous regression loss combined with a classification term. More recently, inspired by GPTs [9], [10], a growing number of studies [11]–[14] discretize agent trajectories into motion tokens and apply a next-token prediction task. We observe that GPT-like models can also be interpreted as mixture models, where each mixture component represents a discrete category. Therefore, we propose the unified mixture model (**UniMM**) framework for multimodal behavior generation, which covers the mainstream continuous mixture models and GPT-like discrete models.

In addition to behavioral multimodality, another major challenge of multi-agent simulation lies in distributional shifts during closed-loop rollouts. To address this, we propose a *closed-loop sample generation* approach for general mixture models, drawing on the philosophy of DaD [15] and TrafficSim [16]. In an effort to enable closed-loop samples to benefit a wide spectrum of mixture models, we further identify and address the *shortcut learning* and *off-policy learning* issues.

UniMM achieves a high ranking on the Waymo Open Sim Agents Challenge (WOSAC) benchmark [17]. This report focuses primarily on the technical implementation of the proposed method. A systematic investigation and comprehensive results can be found in our full paper, which extends the findings on mixture models and provides deeper empirical insights into multi-agent simulation.

Longzhong Lin, Kechun Xu, Haojian Lu, Rong Xiong, and Yue Wang are with Zhejiang University, China (e-mail: {linlongzhong2000, kcxu, luhaojian, rxiong, ywang24}@zju.edu.cn).

Xuewu Lin, and Lichao Huang are with Horizon Robotics, China (e-mail: {xuewu.lin, lichao.huang}@horizon.auto). This work was done during Longzhong Lin's internship at Horizon Robotics.

II. Unified Mixture Model (UniMM)

A. Problem Formulation

Multi-agent simulation is generally factorized into an autoregressive sequential process:

$$p(S_{0:T}|C_0) = \prod_{t\in\{0,\tau,2\tau,\dots,T-\tau\}} p(S_{t:t+\tau}|S_{0:t}, C_0), \quad (1)$$

where $S_{0:T}$ represents the simulated scenario, specifically the state sequence of all agents from time 0 to T, and C_0 denotes the initial scenario context, including map information and historical agent states before time 0. With a sufficiently small update interval τ, each agent can be considered to independently take actions [11], [18]:

$$p(S_{t:t+\tau}|S_{0:t}, C_0) = \prod_{n=1}^{N} p(S^n_{t:t+\tau}|S_{0:t}, C_0, n), \quad (2)$$

where $S^n_{t:t+\tau}$ represents the state sequence of the n-th agent from time t to $t+\tau$, and N denotes the number of agents in the scenario. In practice, we adopt the widely used setting of $\tau = 0.5s$ [13], [14], which aligns with real-world driving [19].

To generate realistic multi-agent simulations, data-driven approaches typically learn a **behavior model**:

$$\pi_\theta(Y|X), \text{ where } \begin{cases} X := (S_{0:t}, C_0, n) \\ Y := S^n_{t:t+T_{\text{pred}}} \end{cases}. \quad (3)$$

The behavior model π_θ predicts Y, the state trajectory of the n-th agent over the prediction horizon T_{pred}, given X which includes the historical agent states, the initial scenario context, and the agent identifier. For modeling multimodal agent behaviors, **mixture models** are a suitable choice [5]:

$$\pi_\theta(Y|X) = \sum_{k=1}^{K} q_\theta(Z=k|X) m_\theta(Y|Z=k, X), \quad (4)$$

where K is the number of mixture components, and Z is the latent variable indicating the component selection. $q_\theta(Z|X)$ predicts the probabilities of selecting each component, while $m_\theta(Y|Z, X)$ represents the distribution of the future trajectory conditioned on the selected component.

Given the dataset $\mathcal{D}$ of real-world driving scenarios, the mixture model π_θ is optimized through **imitation learning**:

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\log \pi_\theta(y|x)], \text{ where } \begin{cases} x := (s_{0:t}, c_0, n) \\ y := s^n_{t:t+T_{\text{pred}}} \end{cases}, \quad (5)$$

where c_0, $s_{0:t}$, $s^n_{t:t+T_{\text{pred}}}$ are respectively the sampled values of C_0, $S_{0:t}$, and $S^n_{t:t+T_{\text{pred}}}$ from the dataset $\mathcal{D}$. Following the EM algorithm [20] and adopting a hard-assignment strategy [21], [22], the optimization objective can be reformulated to incorporate a winner-takes-all regression loss along with a classification term:

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\underbrace{\log m_\theta(y|z^*,x)}_{\text{regression}} - \underbrace{\mathbb{KL}[\hat{q}^*(z)\|q_\theta(z|x)]}_{\text{classification}}\Big], \quad (6)$$

where $\hat{q}^*(z) := \mathbb{1}(z = z^*)$.

Here, $\mathbb{1}(\cdot)$ denotes the indicator function, and z^* represents the positive component of the mixture model π_θ that best matches the ground truth (x, y).

B. Model Configurations

*1) **Positive Component Matching**:* For selecting the positive component z^* in Eq. 6, mixture models can be categorized into two primary paradigms: anchor-free and anchor-based.

In **anchor-free** methods [2], [8], the component z^*, whose predicted trajectory $\mu_{z^*}(x;\theta)$ is closest to the ground truth y, is designated as positive:

$$z^* = \arg\min_k d\big(\mu_k(x;\theta), y\big), \quad (7)$$

where $d(\cdot,\cdot)$ computes the distance between two trajectories, and $\mu_k(x;\theta) := \mathbb{E}_{Y\sim m_\theta(Y|Z=k,X=x)}[Y]$ represents the predicted trajectory corresponding to component k.

In **anchor-based** models [5], [6], [11], the state anchors $\{A_k(x)\}_{k=1}^K$ are each associated with a specific component. The positive component is determined as the one corresponding to the anchor closest to ground truth:

$$z^* = \arg\min_k d\big(A_k(x), y\big). \quad (8)$$

*2) **Continuous Regression**:* For anchor-based models, if the component distribution m_θ is chosen as:

$$m_\theta(Y|Z=k, X=x) = \mathbb{1}(Y = A_k(x)), \quad (9)$$

which means the state anchor $A_k(x)$ is directly used as the predicted trajectory for component k. Hence, the regression term in the training objective (Eq. 6) is redundant, leaving only the classification loss, which fully aligns with GPT-like models [11]–[14]. Thus, **GPT-like discrete models** are essentially anchor-based mixture models, devoid of the trainable component distribution and its corresponding continuous regression term.

*3) **Prediction Horizon**:* Training behavior models with a longer prediction horizon, T_{pred} (Eq. 3), may enhance spatio-temporal interaction reasoning, helping agents become more robust to distributional shifts and generate realistic behaviors [8], [12], [16].

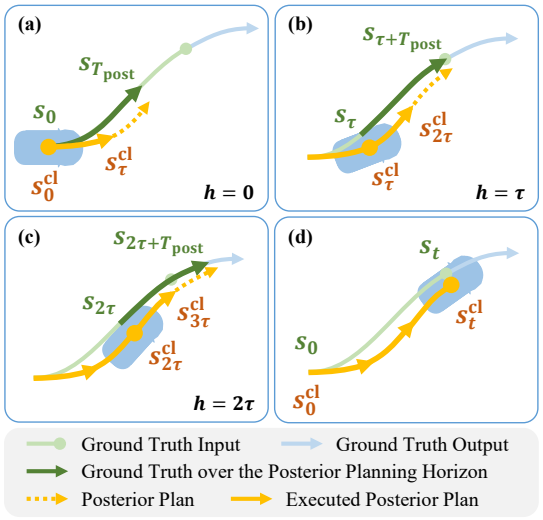

Fig. 1. The illustration of closed-loop sample generation. (a), (b), (c), and (d) sequentially represent the steps of the process.

*4) **Number of Components**:* Intuitively, the number of components, K (Eq. 4), reflects the mixture model's ability to represent complex distributions. Through the related network architecture design (Section II-D2), we enable anchor-based mixture models with continuous regression to scale up the number of components K as efficiently as GPT-like discrete models [11], [13].

C. Data Configuration

*1) **Closed-Loop Sample Generation**:* Inspired by DaD [15] and TrafficSim [16], we propose a closed-loop sample generation method tailored for mixture models to mitigate distributional shifts. Given an open-loop sample $(x = (s_{0:t}, c_0, n), y)$ from the dataset $\mathcal{D}$ (Eq. 5), the behavior model π_θ is unrolled to transform the ground truth input states $s_{0:t}$ into the closed-loop input states $s^{\text{cl}}_{0:t}$, thereby obtaining a closed-loop sample that incorporates the predicted behaviors of π_θ:

$$(x^{\text{cl}} := (s^{\text{cl}}_{0:t}, c_0, n), y) \sim \mathcal{D}^{\text{cl}}. \quad (10)$$

The specific closed-loop sample generation process is illustrated in Fig. 1. Starting from the initial ground truth states $s^{\text{cl}}_0 = s_0$, a **posterior policy** π^{post} is autoregressively applied:

$$\pi^{\text{post}}(x^{\text{cl}}_{:h}, s^n_{h:h+T_{\text{post}}};\theta) := \mu_{z^{\text{post}}}(x^{\text{cl}}_{:h};\theta), \quad (11)$$

where the posterior plan $\mu_{z^{\text{post}}}(x^{\text{cl}}_{:h};\theta)$ for time h is the predicted trajectory corresponding to the component z^{post} of π_θ, based on the previously generated closed-loop input states in $x^{\text{cl}}_{:h} = (s^{\text{cl}}_{0:h}, c_0, n)$. The posterior component z^{post} is the one that best matches the ground truth $s^n_{h:h+T_{\text{post}}}$ over the **posterior planning horizon** T_{post}:

$$z^{\text{post}} = \arg\min_k \begin{cases} d_{T_{\text{post}}}\big(\mu_k(x^{\text{cl}}_{:h};\theta), s^n_{h:h+T_{\text{post}}}\big), & \text{anchor-free} \\ d_{T_{\text{post}}}\big(A_k(x^{\text{cl}}_{:h}), s^n_{h:h+T_{\text{post}}}\big), & \text{anchor-based} \end{cases}. \quad (12)$$

Here, the subscript of $d_{T_{\text{post}}}$ highlights the distance is computed over the shared time interval T_{post}, which may be shorter than T_{pred} in subsequent discussions.

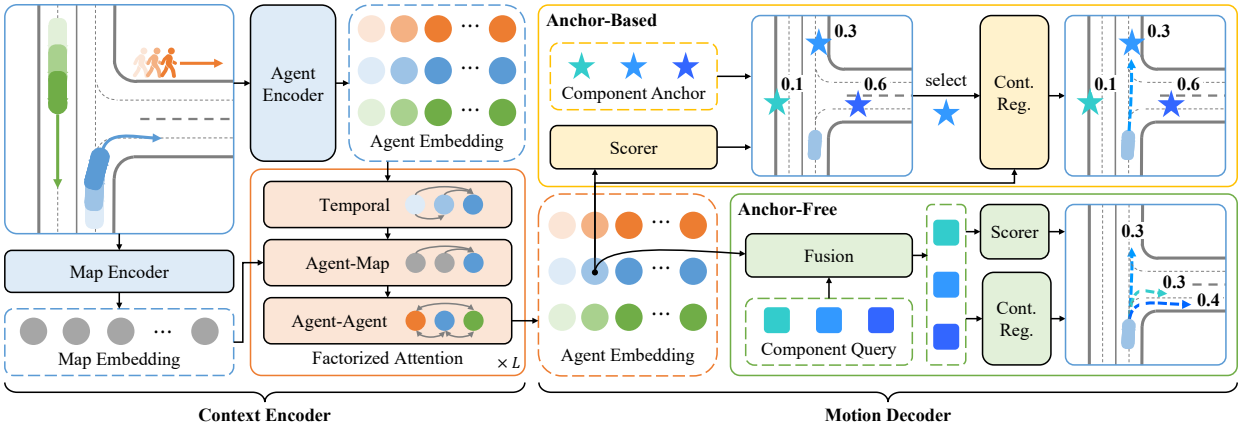

Fig. 2. The demonstration of the network architecture, including the *context encoder* and *motion decoder*. The motion decoders of anchor-based and anchor-free models are designed separately. "Cont. Reg." represents the network for continuous regression, which outputs the trajectory of continuous states for the corresponding mixture component.

The posterior plans of each agent are then executed to generate the subsequent closed-loop states, and the replanning frequency is aligned with the simulation update interval τ:

$$\begin{aligned} \hat{s}^n_{h:h+T_{\text{pred}}} &= \pi^{\text{post}}(x^{\text{cl}}_{:h}, s^n_{h:h+T_{\text{post}}}; \theta), \\ s^{\text{cl}}_{h:h+\tau} &= \{\hat{s}^n_{h:h+\tau}\}_{n=1}^{N}. \end{aligned} \tag{13}$$

At each optimization step during training, the model π_θ first generates a batch of closed-loop samples through the above process, and then updates θ based on these samples:

$$\max_\theta \mathbb{E}_{(x^{\text{cl}},y)\sim\mathcal{D}^{\text{cl}}}[\log \pi_\theta(y|x^{\text{cl}})]. \tag{14}$$

Such closed-loop samples, while attempting to stay close to the ground truth, introduce the model's generated behaviors into the input x^{cl}. As a result, the input states observed during training more closely resemble those encountered by the model in closed-loop simulation.

*2) **Shortcut Learning Issue**:* Considering the goal of making input states more consistent between training and closed-loop simulation, the posterior planning horizon T_{post} is by default set equal to the prediction horizon T_{pred}. However, when T_{pred} exceeds the update interval τ, meaning that the planning horizon of π^{post} is greater than its replanning interval ($T_{\text{post}} > \tau$), the generated closed-loop input x^{cl} will incorporate information from the ground truth output y. As shown in Fig. 1(c), the closed-loop states $s^{\text{cl}}_{2\tau:3\tau}$ are derived based on $s_{2\tau:2\tau+T_{\text{post}}}$ that overlap with the ground truth output. This may lead the model π_θ to learn a shortcut, which could hinder its ability to generate realistic behaviors in closed-loop simulation. Therefore, we attempt to **set the posterior planning horizon equal to the update interval ($T_{\text{post}} = \tau$)** for resolving the shortcut learning issue.

*3) **Off-Policy Learning Problem**:* If one seeks to leverage a longer prediction horizon ($T_{\text{pred}} > \tau$), the aforementioned alignment between the posterior planning horizon and the update interval ($T_{\text{post}} = \tau$) will introduce a misalignment between T_{post} and T_{pred}. Upon closer inspection, this misalignment is primarily reflected in the fact that the behavior model π_θ is trained to select the positive component z^* over T_{pred} (Eq. 7 and Eq. 8), while the posterior policy π^{post} selects the posterior component z^{post} over T_{post} (Eq. 12). This is analogous to the *off-policy problem* in Reinforcement Learning (RL), where a mismatch between the data collection policy and the training policy leads to distributional shifts. Here, the off-policyness is mainly manifested in the disparity between the component selection horizons of π_θ and π^{post}.

In fact, the component selection horizon of π_θ, namely the **positive matching horizon** T_{z^*}, can differ from the prediction horizon T_{pred} by extending Eq. 7 and Eq. 8 as follows:

$$z^* = \arg\min_k \begin{cases} d_{T_{z^*}}\left(\mu_k(x^{\text{cl}};\theta), y\right), & \text{anchor-free} \\ d_{T_{z^*}}\left(A_k(x^{\text{cl}}), y\right), & \text{anchor-based} \end{cases}. \tag{15}$$

Similar to Eq. 12, $d_{T_{z^*}}$ represents the distance calculated over the first T_{z^*} time interval of the trajectories. This indicates that the positive component z^* is selected over the horizon T_{z^*}, while π_θ could still generate trajectories over a longer T_{pred}. We utilize **the alignment between T_{z^*} and T_{post}** to mitigate the off-policy learning problem.

*4) **Approximate Posterior Policy**:* Except for discrete models, closed-loop sample generation involves iterative inference with the model π_θ, which could take a relatively long time. According to previous work [4], the behavior patterns generated by anchor-based models can be reflected in their anchors. Therefore, we propose an **approximate posterior policy** for anchor-based models:

$$\pi^{\text{post}}(x^{\text{cl}}_{:h}, s^n_{h:h+T_{\text{post}}}; \theta) \approx A_{z^{\text{post}}}(x^{\text{cl}}_{:h}), \tag{16}$$

where the anchor $A_{z^{\text{post}}}(x^{\text{cl}}_{:h})$ rather than the predicted trajectory of the component z^{post} is applied. In this way, the generation of closed-loop samples involves only predefined anchors, eliminating the need for π_θ computation and significantly reducing the required time.

D. Network Architecture

*1) **Context Encoder**:* To efficiently process information from multiple agents across multiple time steps simultaneously, we adopt a symmetric scene context encoding based on query-centric attention [3], [5]. Specifically, for each scene element, such as a map polyline or an agent tracklet, the embedding is derived in its local reference frame. When

modeling interactions between scene elements, their relative positional encodings are integrated into the corresponding attention operations. Following existing works [8], [13], we apply map self-attention within the map encoder, as well as the factorized attention containing temporal, agent-map and agent-agent attention (Fig. 2), to obtain agent embeddings enriched with diverse spatio-temporal features. During closed-loop simulation, thanks to the symmetric encoding [3], we can reuse previously derived embeddings to incrementally encode newly generated agent motions for faster inference, akin to the KV cache in LLMs [10].

*2) **Motion Decoder**:* Since the agent embeddings derived from the symmetric encoder contain spatio-temporal information in their local reference frame, the motion decoder treats each embedding equivalently and outputs trajectories in the corresponding local coordinate system. Without loss of generality, focusing on the processing of an individual agent embedding, we next introduce the decoder designs for anchor-free and anchor-based models respectively.

For **anchor-free** models, similar to previous methods [8], each learnable query is linked to a specific component. Every component query is fused with the agent embedding, which is then used to generate the corresponding trajectory along with its confidence score, as depicted in Fig. 2.

For **anchor-based** models, we first generate anchor trajectories for each agent category by clustering the training data, as done in previous works [6], [11]. In this context, for a specific agent category, the anchor corresponding to each component index is actually well-defined. Additionally, the confidence scores assess the congruence between the ground truth and anchors, rather than predicted trajectories. Therefore, we can directly use the agent embedding to predict a categorical distribution over anchors, as shown in Fig. 2. When employing continuous regression, we only need to select a single anchor, either the one associated with the positive component or one sampled based on the scores, and generate its corresponding trajectory. If continuous regression is not applied, the model is equivalent to GPT-like discrete models [13]. While preserving the features inherent to anchor-based models, the above design markedly mitigates the increase in decoder computational cost as the number of components grows.

E. Implementation Details

To balance granularity and efficiency, we downsample the HD map so that the point distance is around 2.5 meters and divide it into polylines with intervals of about 5 meters. The agent trajectories are segmented into multiple tracklets, each with a duration equal to the simulation update interval $\tau = 0.5s$. The embeddings at the map polyline or agent tracklet level are derived through attention-based aggregation, with the dimensions set to 128. For modeling interactions among scene elements, we apply 1 layer of map self-attention and stack 2 layers of factorized attention. The temporal attention has a time window of 3 seconds, and the numbers of neighbors for map-map, agent-map, and agent-agent attention are 16, 64, and 32, respectively. The scorer and continuous regression networks in the decoder are implemented using 2-layer MLPs. In anchor-free models, component queries and agent embeddings are fused by concatenation. In anchor-based models, anchors are generated through the k-means clustering on the training set. Before entering the continuous regression network, the selected anchor is encoded by a 2-layer MLP and then concatenated with the agent embedding. Ultimately, the models in our experiments generally have 4M parameters.

The agent state in the model's predicted trajectory includes 2D position and heading. Following previous works [1], [3], we treat each coordinate and time step in the trajectory as independent. In the component distribution, we use Laplace distributions for position and von Mises distribution for heading. At inference, the output trajectory for a specific component utilizes the expected value of its corresponding component distribution. During closed-loop simulation, we sample the components based on the original output probabilities.

F. Training Details

The models evaluated on the WOSAC [17] benchmark are trained on the full WOMD [23] training set, using the AdamW optimizer with a weight decay of 0.0001. Training is performed with a batch size of 32 scenes for 30 epochs on 8 GPUs (NVIDIA RTX 4090). The learning rate is decayed from 0.0005 to 0 using a cosine annealing scheduler. Consistent with simulation, the scene's $1s$ history is included in the initial context C_0, and all valid agents at the current time are used. Historical states in C_0 are processed by the encoder like other states, except that their agent embeddings are not passed to the decoder for prediction. Using the $8s$ future from the training data, the model predicts in parallel based on corresponding agent embeddings at intervals of $\tau = 0.5s$.

III. RESULTS

Building on the exploratory experiments (see our full paper), we select the best-performing configuration within the UniMM framework for evaluation on the WOSAC [17] benchmark:

- **UniMM (Anchor-Based-4s):** The anchor-based model *with* continuous regression, using $K = 2048$ anchors and a $T_{\text{pred}} = \mathbf{4s}$ prediction horizon, trained with *closed-loop* samples based on the *approximate* posterior policy with a $T_{\text{post}} = 0.5s$ planning horizon. The horizon for positive component matching, $T_{z^*} = 0.5s$, aligns with the posterior planning horizon T_{post}.

As shown in Table I, **UniMM (Anchor-Based-4s)** is competitive with current state-of-the-art methods across all metrics, particularly in terms of minADE, highlighting the advantages of continuous modeling in multi-agent simulation.

IV. CONCLUSION

This technical report introduces the unified mixture model (UniMM) framework for multi-agent simulation, seeking to unify mainstream methods derived from different inspirations. We propose a closed-loop sample generation approach applicable to general mixture models, and further address the shortcut learning and off-policy learning issues to extend its benefits across a broader class of mixture models. As a result, UniMM ranks among the top-performing methods in the Waymo Open Sim Agents Challenge (WOSAC) 2025.

TABLE I
WOSAC 2025 BENCHMARK RESULTS

Method	**Realism Meta** ↑	Kinematic ↑	Interactive ↑	Map-based ↑	minADE ↓
SMART-R1	**0.7855**	0.4940	0.8109	0.9194	1.2990
TrajTok	0.7852	0.4887	0.8116	**0.9207**	1.3179
unimotion	0.7851	**0.4943**	0.8105	0.9187	1.3036
SMART-tiny-CLSFT [24]	0.7846	0.4931	0.8106	0.9177	1.3065
SMART-tiny-RLFTSim	0.7844	0.4893	**0.8128**	0.9164	1.3470
comBOT	0.7837	0.4899	0.8102	0.9175	1.3687
AgentFormer	0.7836	0.4906	0.8103	0.9167	1.3422
UniMM (Anchor-Based-4s)	0.7829	0.4914	0.8089	0.9161	**1.2949**

REFERENCES

[1] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Motion transformer with global intention localization and local movement refinement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6531–6543, 2022.

[2] N. Nayakanti, R. Al-Rfou, A. Zhou, K. Goel, K. S. Refaat, and B. Sapp, "Wayformer: Motion forecasting via simple & efficient attention networks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2980–2987.

[3] Z. Zhou, J. Wang, Y.-H. Li, and Y.-K. Huang, "Query-centric trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 863–17 873.

[4] L. Lin, X. Lin, T. Lin, L. Huang, R. Xiong, and Y. Wang, "Eda: Evolving and distinct anchors for multimodal motion prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 4, 2024, pp. 3432–3440.

[5] S. Shi, L. Jiang, D. Dai, and B. Schiele, "Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[6] Y. Wang, T. Zhao, and F. Yi, "Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023," *arXiv preprint arXiv:2306.11868*, 2023.

[7] C. Qian, D. Xiu, and M. Tian, "The 2nd place solution for 2023 waymo open sim agents challenge," *arXiv preprint arXiv:2306.15914*, 2023.

[8] Z. Zhou, H. Hu, X. Chen, J. Wang, N. Guan, K. Wu, Y.-H. Li, Y.-K. Huang, and C. J. Xue, "Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction," *arXiv preprint arXiv:2405.17372*, 2024.

[9] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, pp. 681–694, 2020.

[10] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[11] J. Philion, X. B. Peng, and S. Fidler, "Trajeglish: Traffic modeling as next-token prediction," in *The Twelfth International Conference on Learning Representations*, 2024.

[12] Y. Hu, S. Chai, Z. Yang, J. Qian, K. Li, W. Shao, H. Zhang, W. Xu, and Q. Liu, "Solving motion planning tasks with a scalable generative model," in *European Conference on Computer Vision*. Springer, 2025, pp. 386–404.

[13] W. Wu, X. Feng, Z. Gao, and Y. Kan, "Smart: Scalable multi-agent real-time simulation via next-token prediction," *arXiv preprint arXiv:2405.15677*, 2024.

[14] J. Zhao, J. Zhuang, Q. Zhou, T. Ban, Z. Xu, H. Zhou, J. Wang, G. Wang, Z. Li, and B. Li, "Kigras: Kinematic-driven generative model for realistic agent simulation," *arXiv preprint arXiv:2407.12940*, 2024.

[15] A. Venkatraman, M. Hebert, and J. Bagnell, "Improving multi-step prediction of learned time series models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015.

[16] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.

[17] N. Montali, J. Lambert, P. Mougin, A. Kuefler, N. Rhinehart, M. Li, C. Gulino, T. Emrich, Z. Yang, S. Whiteson *et al.*, "The waymo open sim agents challenge," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[18] A. Seff, B. Cera, D. Chen, M. Ng, A. Zhou, N. Nayakanti, K. S. Refaat, R. Al-Rfou, and B. Sapp, "Motionlm: Multi-agent motion forecasting as language modeling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8579–8590.

[19] J. Engström, S.-Y. Liu, A. DinparastDjadid, and C. Simoiu, "Modeling road user response timing in naturalistic traffic conflicts: a surprise-based framework," *Accident Analysis & Prevention*, vol. 198, p. 107460, 2024.

[20] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[21] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv preprint arXiv:1910.05449*, 2019.

[22] B. Varadarajan, A. Hefny, A. Srivastava, K. S. Refaat, N. Nayakanti, A. Cornman, K. Chen, B. Douillard, C. P. Lam, D. Anguelov *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7814–7821.

[23] S. Ettinger, S. Cheng, B. Caine, C. Liu, H. Zhao, S. Pradhan, Y. Chai, B. Sapp, C. R. Qi, Y. Zhou *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9710–9719.

[24] Z. Zhang, P. Karkus, M. Igl, W. Ding, Y. Chen, B. Ivanovic, and M. Pavone, "Closed-loop supervised fine-tuning of tokenized traffic models," *arXiv preprint arXiv:2412.05334*, 2024.