Multiverse Transformer: 1st Place Solution for Waymo Open Sim Agents Challenge 2023

Yu Wang Tiebiao Zhao Fan Yi
Pegasus
yuwangrpi@gmail.com

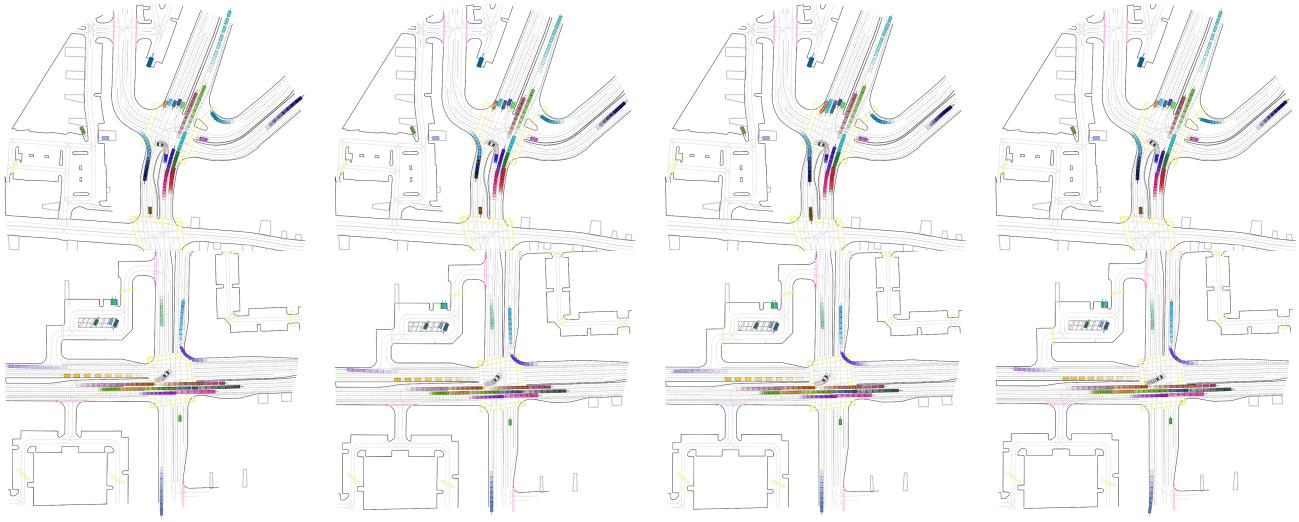


Figure 1: Simulations of agents produced by MVTA. 4 parallel universes are visualized for each of the two scenes, in which the autonomous driving vehicle (ADV) uniquely represented by a car icon for easy identification is undertaking a U-turn maneuver (top) or making a left turn (bottom). For illustrative purposes, eight timesteps are shown for each agent, and time progression is encoded in the opacity of the boxes.

Abstract

*This technical report presents our 1st place solution for the Waymo Open Sim Agents Challenge (WOSAC) 2023. Our proposed **M**ulti**V**erse **T**ransformer for **A**gent simulation (**MVTA**) effectively leverages transformer-based motion prediction approaches, and is tailored for closed-loop simulation of agents. In order to produce simulations with a high degree of realism, we design novel training and sampling methods, and implement a receding horizon prediction mechanism. In addition, we introduce a variable-length history aggregation method to mitigate the compounding error that can arise during closed-loop autoregressive execution. On the WOSAC, our **MVTA** and its enhanced version **MVTE** reach a realism meta-metric of 0.5091 and 0.5168, respectively, outperforming all the other methods on the leaderboard.*

1. Introduction

The simulation of traffic agents is an integral element for evaluating self-driving systems, facilitating rapid development and ensuring safety [17]. WOSAC [11] is the first public benchmark for the evaluation of simulation agents in the domain of autonomous driving, introducing new evaluation metrics and leveraging large-scale real-world logged data [4] with a diverse set of scenarios and agent behaviors.

Recent advancements in traffic agent simulators [17, 2, 5, 20, 8, 15, 18] have shown a notable shift towards learning from logged real-world driving data and data-driven generative models conditioned on the scene context, rather than relying on traditional heuristic-based models encoding traffic

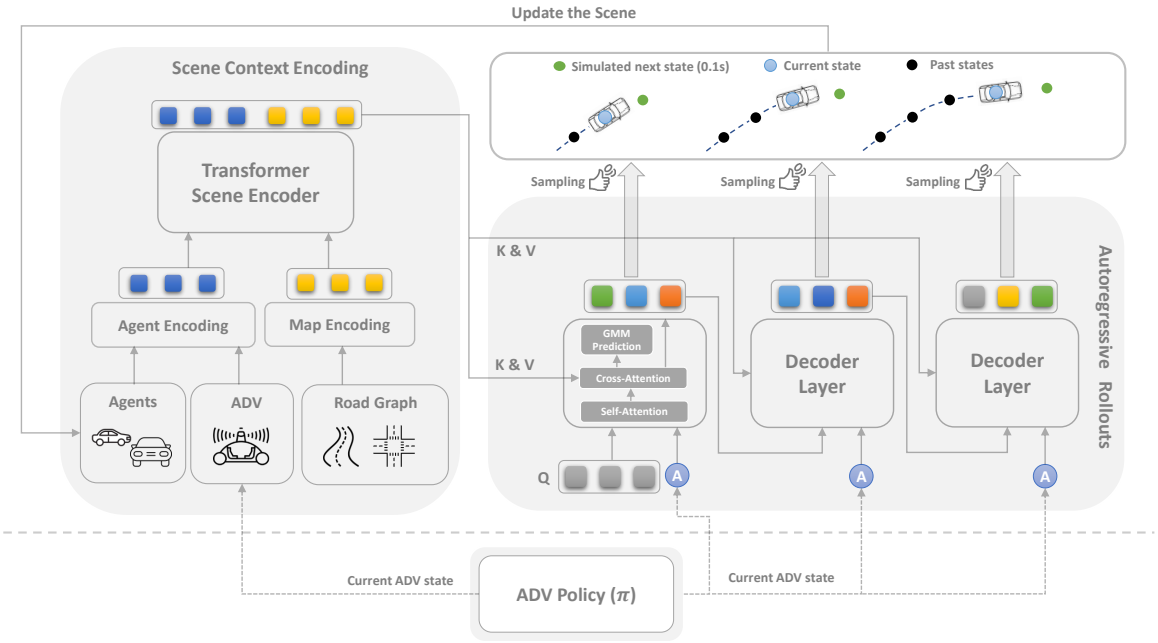


Figure 2: **Main Architecture of the MVTA**. The world agents, ADV and road graph inputs are processed by transformer-based encoding steps to generate enhanced scene context features. In the decoding step, rollout is executed in an autoregressive manner. The ADV operates at 0.1s intervals, and concurrently, each world agent decoder also simulates the forthcoming states at the same 0.1s interval. Q denotes the query content feature, while K and V stand for the keys and values, respectively. The coin flip icon indicates sampling at each timestep.

rules. Our proposed simulator also falls within the learning-based generative model paradigm. Specifically, we leverage state-of-the-art motion prediction models [16, 12] and adapt them to an autoregressive closed-loop agent simulator.

However, for autoregressive models, the error can accumulate over time, as future state of agents is heavily dependent on the immediate past. This can potentially lead to drift, where the predictions become increasingly inaccurate over time.

Inspired by TrafficSim [17] and MTR [16], we propose a novel closed-loop simulation framework based on transformer encoder and decoder, and further enhance the realism of the simulations through the development of novel training and sampling strategies, as well as the receding horizon prediction and variable-length history aggregation methods. Several example simulations generated by MVTA are depicted in Figure 1.

2. Related Work

Multi-modal motion prediction. In the motion prediction literature, there are agent-centric prediction algorithms [12, 16], as well as scene-centric and joint multi-agent [7, 13, 19, 10] prediction methods. Most recent prediction methods adopt encoder-decoder Transformer networks [12, 16, 1]. There is also a trend of employing diffusion in the literature for predicting trajectories through the denoising process [14, 9].

Most motion prediction methods are open-loop, in the sense that the whole trajectory is produced in one-shot independently. However, in this challenge, traffic agent simulation needs to be simulated in a closed-loop autoregressive manner at 0.1s intervals. Additionally, the method must differentiate the world agents and the ADV and factorize their joint distribution for conditional independence.

Traffic agent simulation. TrafficSim [17] utilizes real-world logged data to mimic a broad range of human driving behaviors, and leverages an implicit latent variable model [3] to generate socially-consistent plans for all traffic actors jointly. TrafficGen [5] places agents in the scene based on the learned distribution and simulates their future states. CTG [20] developed a conditional diffusion model that allows user to control over trajectory properties while maintaining realism.

The WOSAC requires the simulator to be agnostic to the choice of ADV policy, therefore it can be swapped with arbitrary ADV policy or planner. Both ADV and environment agent models need to obtain multiple modes in order to perform well [11].

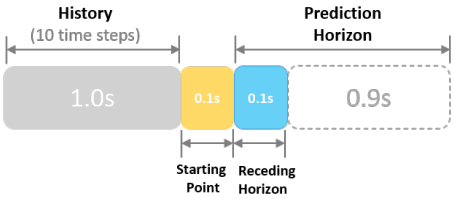


Figure 3: Illustrating the receding horizon. Even though predictions are made for the next 1s, only the waypoint of the initial 0.1s is utilized, with the remaining prediction being discarded.

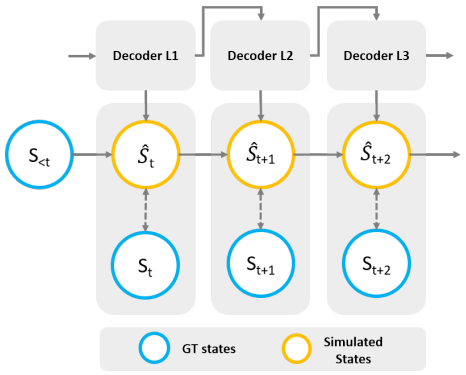


Figure 4: Training loss is calculated at each timestep. Each decoder layer produces multi-modal predictions for one timestep.

3. MultiVerse Transformer Agent Simulator (MVTA)

Problem formulation. Given the scene context, including map and past positions of the agents (*i.e.*, world agents and ADV), the goal is to simulate new states of the agents at 0.1s intervals for the upcoming $T = 80$ timesteps (*i.e.*, a 8s episode). There are two constraints: 1) the simulator must be closed-loop and run in autoregressive manner; 2) the joint distribution involving the world agents and ADV must be factorized into two conditionally independent components, to ensure that the ADV component can be replaced with any arbitrary policy or planner.

3.1. Network Architecture

Main architecture. The main architecture of MVTA is illustrated in Figure 2. Scene context features are obtained by processing the world agents, ADV, and map data through polyline and transformer-based encoding steps. The transformer decoder layer takes the scene context features and queries as input and unrolls the agent states for the next timestep. This architecture is implemented to fulfill the requirement of executing closed-loop simulations at 0.1s intervals. Current state of the ADV is also used as input to the decoder layer so the environment agents can react to it. Query content feature output by each decoder layer is used as the input for the subsequent decoder layer. The motion prediction head, based on Gaussian Mixture Model (GMM), outputs multi-modal trajectory predictions. To sample the state from the multi-modal prediction, we either pick the maximum-likelihood trajectory or randomly sample from the top-k trajectories with the highest likelihood. In our implementation, we leverage the same architecture for the ADV policy, but it can be swapped with any policy or planner.

Transformer-based scene encoder. Given the agent-centric scene inputs (*i.e.*, agents, ADV, and road graph), we utilize their vector representation [6], and adopt polyline encoders consisting of a multi-layer perceptron network (MLP) followed by maxpooling [16]. As in [16], the agent input is represented as the agent history motion state (*i.e.*, agent position, size, heading and velocity) with a one-hot category mask, while the map input consists of the position and direction of each polyline point and the polyline type. The polyline encoders produce agent features $A_{past} \in \mathbb{R}^{N_a \times D}$ and map features $M_{past} \in \mathbb{R}^{N_m \times D}$, where N_a and N_m are the number of agents and map polylines, respectively, and D is the feature dimension. A scene context transformer encoder leverages local self-attention [16] to produce enhanced scene context features that serve as inputs of the subsequent decoder network.

Autoregressive transformer decoder. The autoregressive decoding consists of a group of transformer decoder layers. Each decoder layer has a self-attention component, and a cross-attention component that attends to the scene context features, and a GMM prediction module that produces multi-modal predictions. Each Gaussian component is represented as $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ and predicted with a probability p. The motion prediction head also predicts the velocity (v_x, v_y) and heading angle $(sin(\theta), cos(\theta))$ of each agent for the next timestep. We adopt the motion query pair design in [16]. There are a total of 64 queries, corresponding to the 64 motion modes, each associated with an intention point.

Receding horizon. The next 0.1s state is simulated by sampling the multi-modal predictions output by the decoder layer. However, each decoder layer is trained to predict a 1s trajectory, and we adopt a receding horizon solution in which only the initial 0.1s is utilized, as illustrated in Figure 3. The benefits of longer prediction horizon include the promotion of multi-modal diversity, reduction of compounding error and also more flexibility in inference setup.

Scene context update. The states output by the decoder layer at each timestep are used to update the scene context features. However, the design of our decoder allows running the scene encoding at specified periodic intervals,

rather than at every simulation step. Alternatively, to keep the scene context features updated when predicting the next timestep, we implement two modifications to the network. Firstly, current position of the ADV is used as input to the decoder layer. Similar to the static intention query in [16], sinusoidal position encoding and MLP are applied, and the resulting position embedding is added to the query content. Secondly, we update the agent features with additional features encoding the current positions of the other agents $A = MLP([A_{past}, A_{current}])$.

3.2. Training

Our simulation model is trained end-to-end in a closed-loop manner similar to that used in [17]. As shown in Figure 4, supervision is provided at each decoder layer, and losses are computed for each timestep.

Training samples. The training samples are generated to accommodate variable lengths of past history, as opposed to adhering to a fixed length of 1.1s. Specifically, for each 9.1s training trajectory, we randomly pick a point to separate the trajectory to history and future components. This way, more training samples can be generated from each ground-truth trajectory. Moreover, this facilities the trajectory history aggregating mechanism in our inference step.

Training losses. We use $L1$ loss for regressing the agent velocity and heading angles, and the Gaussian regression loss implemented based on the negative log-likelihood loss to maximize the likelihood of ground-truth trajectory [16].

At each timestep the loss can be formulated as:

$$\mathcal{L}_{NLL} = -\log\mathcal{N}(S_x - \mu_x, \sigma_x; S_y - \mu_y, \sigma_y; \rho) \quad (1)$$

where S_x, S_y is the waypoint of the ground-truth trajectory at this timestep, and $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho)$ represent the parameters of the selected positive Gaussian component.

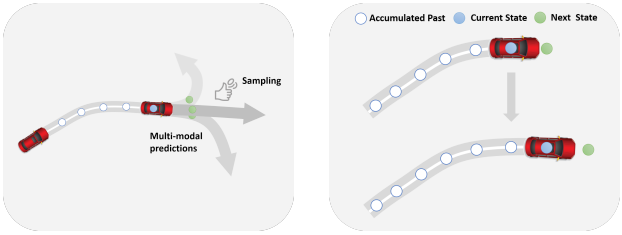
We calculate the final loss based on the weighted sum of the $\mathcal{L}_{NLL}$ loss and the $L1$ losses of velocity and heading angles.

$$\mathcal{L}_{total} = \sum_t \lambda_1 \mathcal{L}^t_{NLL} + \lambda_2 \mathcal{L}^t_{Vel} + \lambda_3 \mathcal{L}^t_{\theta} \quad (2)$$

During the challenge, we also made an attempt to implement a simplified version of the collision avoidance loss [17], however the preliminary result did not indicate any improvement in terms of realism metric, and we leave it to future studies.

3.3. Inference

Top-*k* sampling. During model inference, each simulation step produces multi-modal trajectories. There are two alternatives available for trajectory sampling. The first approach is to select the maximum-likelihood trajectory, while the second is to randomly choose among the top-k (*e.g.*, k=3) trajectories (Figure 5a) with the highest likelihood. The first one, while producing accurate trajectory, tends to yield less varied trajectories. On the other hand, opting for the top-k trajectories encourages diversity but is susceptible to the compounding error and could generate trajectories with unrealistic kinematic motions or even drift, As a result, we employ the top-k sampling at periodic intervals during the simulation steps to strike a balance between realism and diversity.



(a) (b)

Figure 5: (a) Top-k sampling. (b) Aggregating new waypoint to the past trajectory.

Variable-length history aggregation. Instead of using fixed 1.1s history, we continuously aggregate the past history as the agent state unrolls overtime (Figure 5b), and use the aggregated history trajectory for the scene context encoding in the next simulation step. The motivation is two-fold, firstly, our training process also uses variable-length history. Secondly, we aim to enhance the stability of the trajectory simulation, thereby reducing the potential for compounding error. One example showcasing the autoregressive rollout is shown in Figure 6.

4. Experimental Evaluation

4.1. Dataset and Metrics

We use the Waymo Open Motion Dataset (WOMD) [4] v1.2.0 release in our experiments. There are a total of 486,995, 44,097, and 44,920 scenarios in the training, validation, and test set, respectively. Each scenario in the training and validation sets comprises of 11 observations for history and 80 observations from 8 seconds of future data, therefore the total duration of each scenario is 9.1 seconds.

The task is to simulate up to 128 agents including the ADV, and generate 80 simulation steps (8s) for each agent in a 0.1s sampling interval, and in an autoregressive and reactive manner [11]. 32 simulations are required for each agent to be simulated.

There are three object types (vehicles, cyclists, and pedestrians), and their $x/y/z$ centroid coordinates and heading need to be simulated. There is no need to simulate the size of each agent since it stays constant. In our experiments, we keep the z value the same as the starting

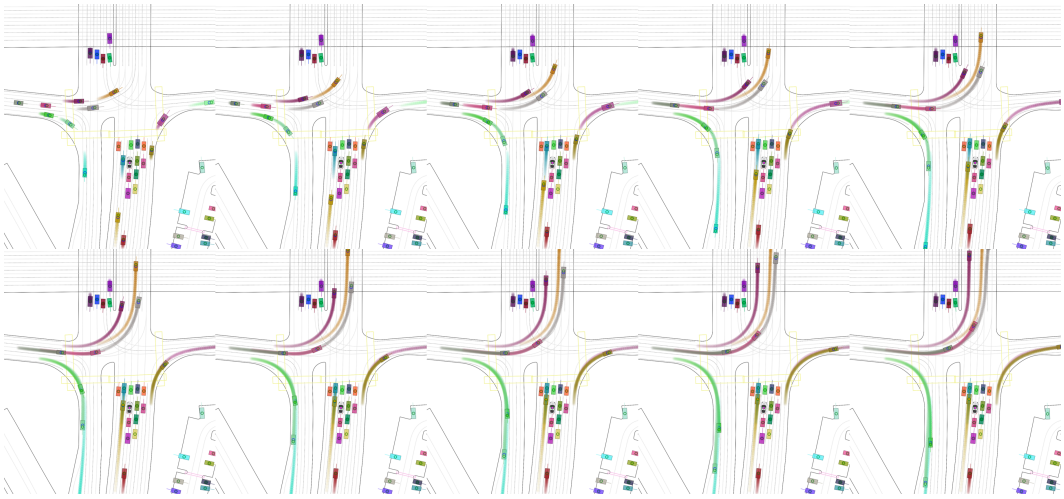


Figure 6: An example of the autoregressive rollout. The context history, current state (blue-edged circle) and next simulated waypoint (green-edged circle) are visualized for each agent.

WAYMO	META METRIC	KINEMATIC				INTERACTIVE			MAP		
LEADERBOARD	**REALISM**	LINEAR SPEED	LINEAR ACCEL.	ANG. SPEED	ANG. ACCEL.	DIST. TO OBJ.	COLLISION	TTC	DIST. TO ROAD	OFFROAD	minADE (↓)
MVTE (ours)	**0.5168**	0.4426	0.2218	**0.5353**	**0.481**	**0.382**	0.4509	**0.832**	**0.6641**	**0.6409**	1.677
MVTA (ours)	0.5091	0.4365	0.22	0.533	0.4805	0.3729	0.4359	0.8298	0.6545	0.6288	1.8698
MTR+++	0.4697	0.4119	0.1066	0.4838	0.4365	0.3457	0.4144	0.7969	0.6545	0.577	1.6817
CAD	0.4321	0.3464	**0.2526**	0.4327	0.311	0.33	0.3114	0.7893	0.6376	0.5397	2.3146
multipath	0.424	0.4318	0.2304	0.0193	0.0355	0.3493	**0.4854**	0.8111	0.6372	0.613	2.0517
sim_agents_tutorial	0.3941	0.3143	0.1738	0.4785	0.4631	0.2641	0.2671	0.7709	0.5575	0.4111	3.6198
QCNeXt	0.392	**0.4773**	0.2424	0.3252	0.1987	0.3759	0.3244	0.7569	0.6099	0.36	**1.083**
sim_agents_tutorial	0.3201	0.3826	0.0999	0.0318	0.0391	0.2909	0.336	0.7549	0.5251	0.3804	3.108
linear_extrapolation_baseline_tutorial	0.2576	0.0745	0.1659	0.0187	0.0348	0.2221	0.2211	0.7551	0.479	0.3352	7.5148

Table 1: **WOSAC Leaderboard**. Realism meta-metric is the primary metric for ranking the methods. Our simulator reached the highest meta-metric of 0.5168 among all the methods on the leaderboard.

state. The challenge does not enforce any motion model, and therefore there are no kinematic constraints.

The main evaluation metric is the realism meta-metric, aggregating a group of component metrics including kinematic, interactive and map-based metrics. For more details, please refer to [11].

4.2. Implementation and Simulation Setup

Table 2 summarizes the hyperparameters of different modules used in our implementation.

Training details. The simulation model is trained end-to-end for all three agent types, using AdamW optimizer for 30 epochs. The learning rate is set to 0.0001. We set the loss weights $\lambda_1, \lambda_2, \lambda_3$ in Equation (2) to 1.0, 0.5, 0.5, respectively. Similar to [16], we use 64 motion query pairs based on 64 intention points learned by running k-means clustering algorithm on the future 1s waypoints of the training trajectories. A set of 64 intention points is obtained for each object category.

Batch inference and optimization. There are a total of 44,920 scenes in the test set, and each scene requires running the model inference for $32 \times T$ times to generate the 32 simulations for a group of agents. As such, we implemented batch inference to speed up the simulation process. Given that our model design supports periodic updates of the scene context features, the inference speed can be further optimized by running the scene context encoder every few timesteps (*e.g.*, 0.5s) and running several decoder layers (*e.g.*, the first 5).

MVTE. We explore the design space of the MVTA model and trained 3 variants of the model by increasing the number of hidden feature dimension in the encoder (*e.g.*, 384 as opposed to 256) and decoder (*e.g.*, 768 as opposed to 512). In the enhanced MVTE solution, model is randomly sampled to generate each simulation, encouraging more diversity in the resulting simulations.

4.3. Experimental Results

WOSAC 2023 leaderboard. On the WOSAC leaderboard[1], the realism meta-metric is the official primary met-

[1] https://waymo.com/open/challenges/2023/sim-agents/

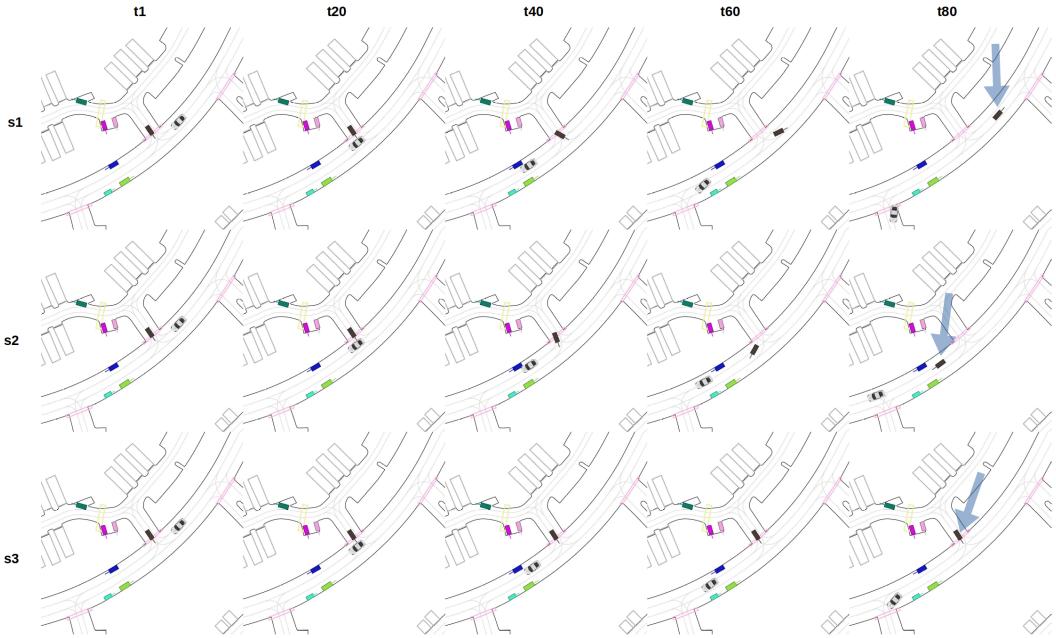


Figure 7: Three simulations of a scene, in which a vehicle waiting to get onto the main road. The vehicle turns left or right, or keeps waiting for the right time to go. Additionally, the ADV also demonstrates multi-modal behavior, either proceeding straight or making a left turn. Five timesteps are rendered for each simulation.

Module	Hyperparameters	Values
Scene MLP	No. Channels-Agent	256
	No. Layers-Agent	3
	No. Channels-Map	64
	No. Layers-Map	5
Encoder	Hidden Feature Dim.	256/384
	No. Encoder Layers	6
	No. Attention Head	8
Decoder	Hidden Feature Dim.-Agent	512/768
	Hidden Feature Dim.-Map	256/384
	No. Decoder Layers	10
	No. Attention Head	8
	No. Motion Modes	64
Training	Learning rate	0.0001
	No. Epochs	30
	Loss weights	1.0, 0.5, 0.5

Table 2: Hyperparameters of different modules in MVTA.

ric used for ranking the methods. The minADE metric is also calculated but it is primarily used for evaluating motion prediction methods. The official baseline extrapolates the trajectory of an agent using the last heading and speed logged in the provided history [11]. For more baselines based on Wayformer [12] on the validation set, please refer to [11].

The leaderboard is shown in Table 1. On the test set, our MVTA reaches a realism meta-metric of 0.5091 and our MVTE further improves the meta-metric to 0.5168, ranking the 1st place in the challenge. Notably, it also has the highest scores in component metrics except the linear and collision metrics.

Note in Table 1 that minADE does not always correlate with the ranking, as the method achieving the lowest minADE has lower realism meta metric compared to other methods.

Qualitative results. In Figure 7, we present a scenario demonstrating the multi-modal behavior of an agent. Figure 8 features five simulated scenarios showcasing reactive environment agents. These agents exhibit a wide variety of behaviors including yielding, overtaking, pausing for unprotected left turns, and engaging with the ADV. Qualitative simulation results of several intersection scenes with agents undertaking a wide variety of maneuvers are provided in Figure 9. Due to the complexity of these scenes, it is impossible for heuristic-based models that encode traffic rules to simulate these realistic agents.

5. Conclusion

In this technical report, we have presented the Multiverse Transformer (MVTA) framework which produces parallel universes for the application of traffic agents simulation. It achieved state-of-the-art performance and ranks the 1st place in the Waymo Open Sim Agents Challenge 2023. We hope our work inspires further research in the area of simulation agents. In our upcoming research, we intend to investigate scene-centric simulation approaches for improving the degree of realism of simulations, and also explore the possibility of using diffusion/denoising-based approaches.

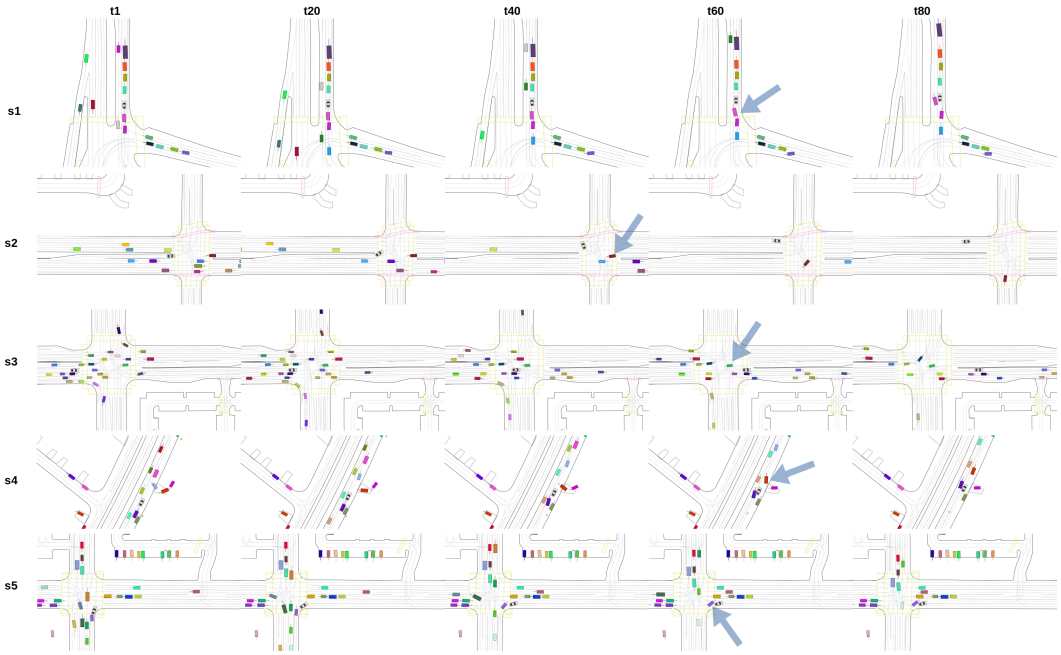


Figure 8: Five simulated scenarios featuring reactive environment agents, each with five timesteps rendered. Scenario 1 depicts a congested right lane where an agent, indicated by the arrow, attempts to overtake the ADV. Scenario 2 and 3 showcase vehicles at intersections, waiting for the oncoming traffic to clear before making an unprotected left turn. Scenario 4 shows the ADV slows down to yield to a car merging onto the main road from a driveway. The car behind the ADV then changes to the left lane and overtakes the autonomous vehicle. Lastly, scenario 5 demonstrates the ADV executing a slow right-turn, resulting in the agent behind it having to slow down or stop.

References

[1] Lina Achaji, Thierno Barry, Thibault Fouqueray, Julien Moreau, Francois Aioun, and Francois Charpillet. Pretr: Spatio-temporal non-autoregressive trajectory prediction transformer. arXiv:2203.09293, 2022.

[2] Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Blazej Osinski, Hugo Grimmett, and Peter Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *ICRA*, 2021.

[3] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *ECCV*, 2020.

[4] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Pei Sun, and Dragomir Anguelov et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *ICCV*, 2021.

[5] Lan Feng, Quanyi Li, Zhenghao Peng, Shuhan Tan, and Bolei Zhou. Trafficgen: Learning to generate diverse and realistic traffic scenarios. In *ICRA*, 2023.

[6] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation, 2020.

[7] Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *ICLR*, 2022.

[8] Maximilian Igl, Daewoo Kim, Alex Kuefler, Paul Mougin, Punit Shah, Kyriacos Shiarlis, Dragomir Anguelov, Mark Palatucci, Brandyn White, and Shimon Whiteson. Symphony: Learning realistic and diverse agents for autonomous driving simulation. In *ICRA*, 2022.

[9] Chiyu Max Jiang, Andre Cornman, Cheolho Park, Ben Sapp, Yin Zhou, and Dragomir Anguelov. Motiondiffuser: Controllable multi-agent motion prediction using diffusion. In *CVPR*, 2023.

[10] Wenjie Luo, Cheolho Park, Andre Cornman, Benjamin Sapp, and Dragomir Anguelov. Jfp: Joint future prediction with interactive multi-agent modeling for autonomous driving. In *PMLR*, 2022.

[11] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nick Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, Brandyn White, and Dragomir Anguelov. The waymo open sim agents challenge. arXiv:2305.12032, 2023.

[12] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple and efficient attention networks. arXiv:2207.05844, 2022.

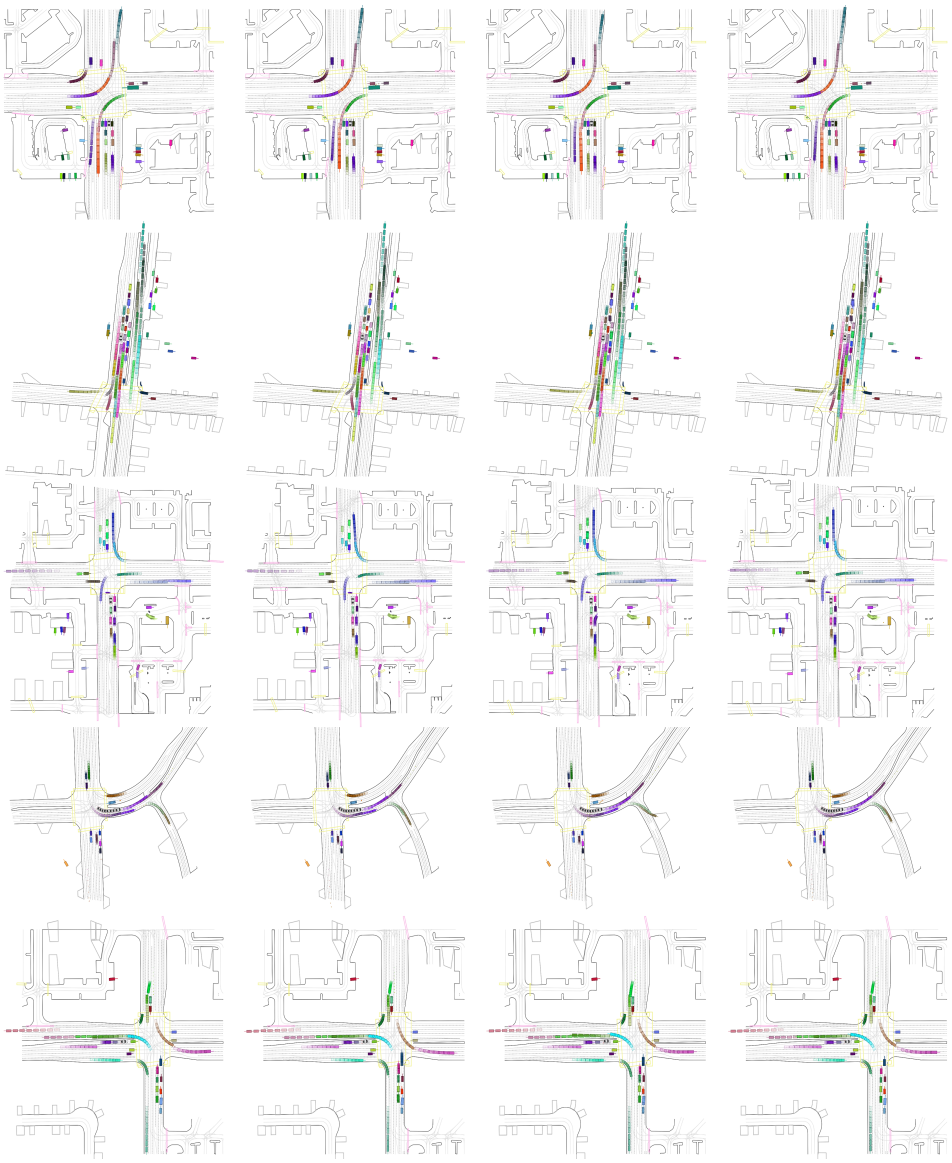


Figure 9: Qualitative simulation results of several complex intersection scenarios. Each row shows 4 out of the 32 simulations for a scenario.

[13] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, and et al. Chenxi Liu. Scene transformer: A unified architecture for predicting multiple agent trajectories. In *ICLR*, 2022.

[14] Davis Rempe, Zhengyi Luo, Xue Bin Peng, Ye Yuan, Kris Kitani, Karsten Kreis, Sanja Fidler, and Or Litany. Trace and pace: Controllable pedestrian animation via guided trajectory diffusion. In *CVPR*, 2023.

[15] Davis Rempe, Jonah Philion, Leonidas J. Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *CVPR*, 2022.

[16] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion transformer with global intention localization and local movement refinement. In *NeurIPS*, 2022.

[17] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *CVPR*, 2021.

[18] Danfei Xu, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Bits: Bi-level imitation for traffic simulation. arXiv:2208.12403, 2022.

[19] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *ICCV*, 2021.

[20] Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *ICRA*, 2023.