BehaviorGPT: Technical Report for 2024 Waymo Open Sim Agents Challenge

Zikang Zhou[1*] Haibo Hu[1*] Xinhong Chen[1] Jianping Wang[1] Nan Guan[1]
Kui Wu[2] Yung-Hui Li[3] Yu-Kai Huang[4] Chun Jason Xue[5]
[1]City University of Hong Kong [2]University of Victoria [3]Hon Hai Research Institute
[4]Carnegie Mellon University [5]Mohamed bin Zayed University of Artificial Intelligence

Abstract

Simulating realistic agents is key for validating autonomous driving systems. Current data-driven simulators use encoder-decoder structures, complicating models and harming data and parameter efficiency. This technical report introduces BehaviorGPT, a decoder-only autoregressive model that treats each time step as the current one for enhanced data and parameter efficiency. In addition, our Next-Patch Prediction Paradigm (NP3) allows models to reason at the patch level and capture long-range interactions. BehaviorGPT achieves state-of-the-art performance on the Waymo Open Sim Agents Benchmark with a realism score of 0.7473 *and a minADE score of* 1.4147*, using merely* $3M$ *model parameters. Our code will be released in the future.*

1. Introduction

Autonomous driving demands reliable evaluation of system safety. While on-road testing is costly and lacks safety-critical scenarios, simulation offers a low-cost alternative for large-scale offline testing. This technical report focuses on simulating traffic agents' behavior, which is crucial for validating the driving policies of autonomous vehicles (AVs).

Existing learning-based agent simulators [3, 6] often adopt an encoder-decoder architecture inspired by motion forecasting models. These models encode historical information to facilitate future state decoding, requiring manually splitting the time series into history and future segments. However, using heterogeneous encoders and decoders complicates the architecture. Moreover, we expect a sample-efficient framework to learn from all possible history-future pairs of time series. This is difficult to achieve with encoder-decoder models owing to their heterogeneous processing for historical and future time steps.

Inspired by the success of Large Language Models (LLMs), we introduce BehaviorGPT, a decoder-only autoregressive architecture for agent simulation. Our BehaviorGPT uses homogeneous Transformer blocks [5] to process the entire trajectory snippets without separating history and future, resulting in a simpler and more efficient simulator. By employing relative spacetime representations [7] to model each agent state as the current one, we require each agent state to perform future prediction, thereby maximizing data utilization. To address compounding errors and causal confusion in autoregressive modeling, we introduce the Next-Patch Prediction Paradigm (NP3). Our proposed NP3 enables models to reason at the patch level of trajectories, facilitating long-range interaction modeling and preventing models from leveraging trivial shortcuts at training time. Equipped with NP3, BehaviorGPT achieves superior performance in the Waymo Open Sim Agents Challenge (WOSAC) [2] with merely 3M model parameters, demonstrating its effectiveness for agent behavior simulation.

2. Method

This section presents our proposed BehaviorGPT for multi-agent behavior simulation, with Fig. 1 illustrating the overall framework.

2.1. Problem Formulation

In multi-agent behavior simulation, we aim to simulate agents' future behavior in dynamic environments. A scenario consists of a vector map M and the states of N_{agent} agents over T time steps. The state S_i of the i-th agent at each time step includes position, velocity, yaw angle, and bounding box size. Agent types (e.g., vehicles, pedestrians, and cyclists) are also given. We formulate this as sequential predictions over trajectory patches, where the prediction of each patch will affect the subsequent patches. An agent-level trajectory patch is defined as:

$$P_i^\tau = S_i^{(\tau-1)\times\ell+1:\tau\times\ell}, \; i \in \{1,\dots,N_{\text{agent}}\}, \; \tau \in \{1,\dots,N_{\text{patch}}\}, \tag{1}$$

[*]Equal contribution

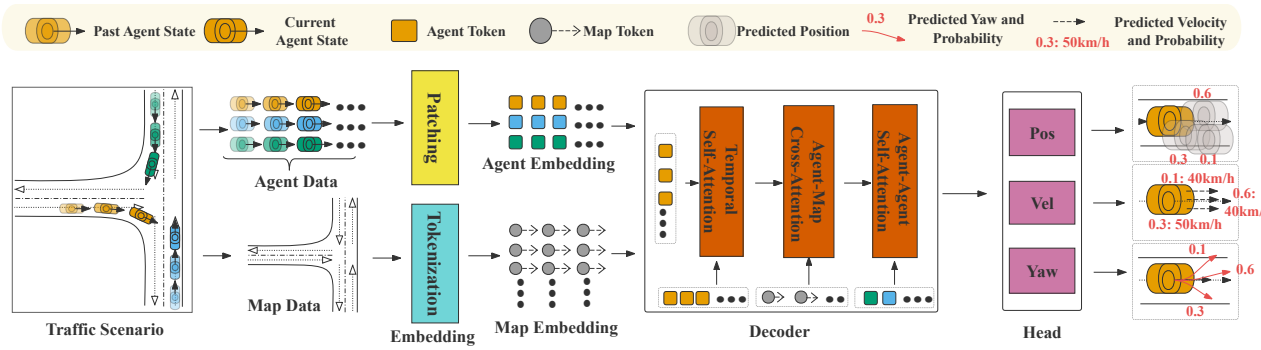

Figure 1. **Overview of BehaviorGPT.** Agent and map data are embedded and fed into a Transformer decoder for autoregressive modeling based on next-patch prediction.

where ℓ is the number of time steps per patch, $N_{\text{patch}} = T/\ell$, and P_i^τ represents the τ-th patch of the i-th agent. A scene-level patch is defined as:

$$P^\tau = S_{1:N_{\text{agent}}}^{(\tau-1)\times\ell+1:\tau\times\ell}, \tau \in \{1, \ldots, N_{\text{patch}}\}, \quad (2)$$

which includes all agents' states at the τ-th patch. We factorize the joint distribution over scene-level patches using the chain rule:

$$\begin{aligned}\Pr\left(S_{1:N_{\text{agent}}}^{1:T} \mid M\right) &= \Pr\left(P^{1:N_{\text{patch}}} \mid M\right) \\ &= \prod_{\tau=1}^{N_{\text{patch}}} \Pr\left(P^\tau \mid P^{1:\tau-1}, M\right),\end{aligned} \quad (3)$$

where $\Pr(S_{1:N_{\text{agent}}}^{1:T} \mid M)$ is the joint distribution of all agents' states over all time steps conditioned on the map M. Further, we assume that agents plan their motions independently within the horizon of a patch, factorizing the conditional distribution of each scene-level patch over agents:

$$\Pr\left(P^\tau \mid P^{1:\tau-1}, M\right) = \prod_{i=1}^{N_{\text{agent}}} \Pr\left(P_i^\tau \mid P^{1:\tau-1}, M\right). \quad (4)$$

Given the multimodality of agents' behavior, we assume $\Pr(P_i^\tau \mid P^{1:\tau-1}, M)$ to be a mixture model with N_{mode} modes:

$$\Pr\left(P_i^\tau \mid P^{1:\tau-1}, M\right) = \sum_{k=1}^{N_{\text{mode}}} \pi_{i,k}^\tau \Pr\left(P_{i,k}^\tau \mid P^{1:\tau-1}, M\right), \quad (5)$$

where $\pi_{i,k}^\tau$ is the probability of the k-th mode. Since a patch covers ℓ time steps, we further apply the chain rule to factorize the states per mode over time steps:

$$\begin{aligned}\Pr\left(P_{i,k}^\tau \mid P^{1:\tau-1}, M\right) &= \Pr\left(S_{i,k}^{(\tau-1)\times\ell+1:\tau\times\ell} \mid P^{1:\tau-1}, M\right) \\ &= \prod_{t=(\tau-1)\times\ell+1}^{\tau\times\ell} \Pr\left(S_{i,k}^t \mid S_{i,k}^{(\tau-1)\times\ell+1:t-1}, P^{1:\tau-1}, M\right).\end{aligned} \quad (6)$$

Such an autoregressive formulation can be interpreted as planning the patch-level behavior of each agent independently (Eq. (4)), fixing agents' behavior mode per ℓ time steps (Eq. (5)), and autoregressively unrolling the next state given a specific behavior mode (Eq. (6)). However, we can also increase the replan frequency at inference time to enhance agents' reactivity. For instance, although we assume that agents plan for ℓ steps, we may let them execute only one step of the planned actions and choose a new behavior mode at the next step to react to the change in environments.

2.2. Relative Spacetime Representation

We adopt the relative spacetime representation introduced in QCNet [7] to model the patches symmetrically in space and time, achieving simultaneous multi-agent prediction in Eq. (4) and allowing parallel next-patch predictions as depicted in Eq. (3). Under this representation, the features of each map element and agent state are derived from coordinate-independent attributes (e.g., the semantic category of a map element and the speed of an agent state). On top of this, we effectively maintain the spatial-temporal relationships between input elements via relative positional embeddings:

$$\mathcal{R}_{j\to i} = \text{MLP}\left(\|\boldsymbol{d}_{j\to i}\|, \angle(\boldsymbol{n}_i, \boldsymbol{d}_{j\to i}), \Delta\boldsymbol{\theta}_{j\to i}, \Delta\boldsymbol{\tau}_{j\to i}\right), \quad (7)$$

where $R_{j\to i}$ is the relational embedding from j to i, $\|d_{j\to i}\|$ is the Euclidean distance between them, $\angle(n_i, d_{j\to i})$ is the angle between n_i (the orientation of i) and $d_{j\to i}$ (the displacement vector from j to i), $\Delta\theta_{j\to i}$ is the relative yaw angle from j to i, and $\Delta\tau_{j\to i}$ is the time difference.

2.3. Map Tokenization and Agent Patching

We convert raw information into neural embeddings before performing spatial-temporal relational reasoning among traffic scene elements. We embed map information by sampling points along the polylines every 5 meters and tokenizing their semantic category (e.g., lane centerlines, road edges) via learnable embeddings. The shape of the embedding table for map elements is $[17, 128]$, indicating

17 semantic categories in total and a hidden size of 128. The i-th map point's embedding is denoted as $\hat{M}_i$, which only contains semantic information and does not include any information about coordinates. For agent states, we use attention-based patching to obtain patch-level trajectory embeddings. Specifically, we transform the coordinate-independent attributes of the i-th agent's state S_i^t (i.e., the agent type, the bounding box size, the speed, and the angle of the velocity vector relative to the bounding box's yaw angle) with an MLP, obtaining state embedding $\hat{S}_i^t$. We then collect ℓ consecutive state embeddings and apply attention with relative positional embeddings:

$$\hat{P}_i^\tau = \text{MHSA}(Q = \hat{S}_i^{\tau\times\ell}, K = V = \{[\hat{S}_i^t,\, \mathcal{R}_i^{t\to\tau\times\ell}]\}, \\ t \in \{(\tau-1)\times\ell+1, \ldots, \tau\times\ell-1\})\,, \tag{8}$$

Here, $\hat{P}_i^\tau$ is the patch embedding of the i-th agent at the τ-th patch, $\text{MHSA}(\cdot)$ is multi-head self-attention, $[:, :]$ denotes concatenation, and $\mathcal{R}_i^{t\to\tau\times\ell}$ is the positional embedding of S_i^t relative to $S_i^{\tau\times\ell}$.

2.4. Triple-Attention Transformer Decoder

After obtaining map tokens and agent patch embeddings, we use a Transformer decoder with a triple-attention mechanism to model spatial-temporal interactions among scene elements. The triple-attention mechanism considers three sources of information fusion: the temporal dependencies between the trajectory patches of an agent, the map regulations on agents, and the social interactions among agents. First, we use causal self-attention in the time dimension to capture the relationships among an agent's trajectory patches, accommodating our autoregressive formulation. Second, a k-NN agent-map cross-attention is employed to model the environmental influences on agents. Third, we apply a k-NN agent-agent self-attention to capture the social interactions. All these attention modules employ relative spatial-temporal positional embeddings similar to QCNet [7]. The triple-attention mechanism can be depicted as:

$$F_{a2t,i}^\tau = \text{MHSA}(Q = \hat{P}_i^\tau, K = V = \{[\hat{P}_i^t,\, \mathcal{R}_i^{t\times\ell\to\tau\times\ell}]\}, \\ t \in \{1, \ldots, \tau-1\})\,, \tag{9}$$

$$F_{a2m,i}^\tau = \text{MHCA}(Q = F_{a2t,i}^\tau, K = V = \{[\hat{M}_j,\, \mathcal{R}_{j\to i}^{\tau\times\ell}]\}, \\ j \in \mathcal{N}(i,\tau))\,, \tag{10}$$

$$F_{a2a,i}^\tau = \text{MHSA}(Q = F_{a2m,i}^\tau, K = V = \{[F_{a2m,j}^\tau,\, \mathcal{R}_{j\to i}^{\tau\times\ell}]\}, \\ j \in \mathcal{N}(i,\tau))\,, \tag{11}$$

where $\mathcal{N}(i,\tau)$ denotes the k-nearest agent/map neighbors of the i-th agent at the τ-th patch. Our decoder is relatively efficient thanks to (1) factorizing the space and time dimensions and (2) using k-NN attention with $k = 32$ for spatial interaction modeling. Limited by computing resources for model training, we only stack two layers for each attention module with 8 attention heads and a hidden size of 128. We welcome researchers with sufficient computing resources to examine the scalability of our architecture.

Table 1. Comparison of the replan frequency on the test set.

Replan Frequency	minADE (↓)	Realism	Collision	Offroad
1 Hz	1.5405	0.7414	0.9520	0.9308
2 Hz	**1.4147**	**0.7473**	**0.9537**	**0.9349**
5 Hz	1.5693	0.7342	0.9429	0.9089

Given the patch embeddings updated by the Transformer decoder, we develop a next-patch prediction head to model the marginal multimodal distribution of agent trajectories. For the τ-th patch of the i-th agent, given the attention output $F_i^\tau \in \mathbb{R}^{128}$, we estimate the next patch's mixture model parameters pre-defined with $N_{\text{mode}} = 16$ modes. These modes are automatically learned without relying on any anchors. An MLP transforms F_i^τ into mixing coefficients $\pi_i^{\tau+1} \in \mathbb{R}^{N_{\text{mode}}}$. For each mode, the conditional distribution of the next agent state, as depicted in Eq. (6), is considered a multivariate marginal distribution with position and velocity components as Laplace distributions and the yaw angle as a von Mises distribution. Under this formulation, a GRU-based autoregressive RNN unrolls the states of the next patch step by step, with each step being conditioned on the last agent state. The RNN's hidden state $h_{i,k}^{\tau,t} \in \mathbb{R}^{128}$ is initialized with F_i^τ at $t = 1$ for $\forall k \in \{1, \ldots, N_{\text{modes}}\}$. At each step, an MLP estimates the location and scale parameters of the next agent state's position and velocity based on the hidden state. The MLP also estimates the location and concentration parameters of the next agent state's yaw angle. The location parameters of the newly predicted state, including the 3D positions, the 2D velocities, and the yaw angle, are used to update the RNN's hidden state directly without relying on the predicted scale/concentration parameters for sampling. The whole process is summarized as:

$$\begin{aligned} \pi_{i,k}^{\tau+1} &= \text{MLP}([F_i^\tau, Z_k])\,, \\ h_{i,k}^{\tau,1} &= F_i^\tau\,, \\ \mu_{i,k}^{\tau\times\ell+t},\, b_{i,k}^{\tau\times\ell+t},\, \kappa_{i,k}^{\tau\times\ell+t} &= \text{MLP}([h_{i,k}^{\tau,t}, Z_k])\,, \\ h_{i,k}^{\tau,t+1} &= \text{RNN}(h_{i,k}^{\tau,t},\, \text{MLP}(\mu_{i,k}^{\tau\times\ell+t}))\,. \end{aligned} \tag{12}$$

Here, $\{\mu_{i,k}^{\tau\times\ell+t} \in \mathbb{R}^6\}_{t\in\{1,\ldots,\ell\}}$, $\{b_{i,k}^{\tau\times\ell+t} \in \mathbb{R}^5\}_{t\in\{1,\ldots,\ell\}}$, and $\{\kappa_{i,k}^{\tau\times\ell+t} \in \mathbb{R}\}_{t\in\{1,\ldots,\ell\}}$ are location, scale, and concentration parameters, $Z_k \in \mathbb{R}^{128}$ is the k-th learnable mode embedding.

3. Experiments

We use the Waymo Open Motion Dataset (WOMD) [1] to evaluate the effectiveness of our BehaviorGPT.

Table 2. Performance evaluation of different models on the Waymo Open Sim Agents Benchmark.

Method Name	minADE (↓)	Realism Meta metric	Kinematic metrics	Interactive metrics	Map-based metrics
BehaviorGPT	**1.4147**	0.7473	0.4333	0.7997	**0.8593**
VBD	1.4743	0.7200	0.4169	0.7819	0.8137
SMART	1.5435	0.7457	0.4168	**0.8053**	0.8571
SMART	1.5447	**0.7511**	**0.4445**	0.8050	0.8571
GUMP	1.6041	0.7431	0.4780	0.7887	0.8359
MVTE	1.6770	0.7302	0.4503	0.7706	0.8381
TrafficBotsV1.5	1.8825	0.6988	0.4304	0.7114	0.8360
cogniBOT v1.5	1.8832	0.6288	0.3293	0.7129	0.6918

Table 3. Comparison of model parameters.

BehaviorGPT	SMART	Trajeglish [3]	MTR++ [4]	MTR [4]
3M	8M	35.6M	86.6M	65.8M

3.1. Implementation Details

The optimal patch size we experimented with is 10, corresponding to 1 second. All hidden sizes in the model are set to 128. All attention layers have 8 attention heads with 16 dimensions per head. Each attention layer is enhanced by residual connections, a feed-forward network activated by the ReLU function, and Layer Normalization in a pre-norm manner. We apply the same architecture to make predictions for all agents. To differentiate the policies of surrounding vehicles and AVs, we use an AV-specific learnable embedding to incorporate the agent type information when embedding AVs' states. We can also easily swap with any other AV policy, as we have properly factorized the multi-agent distribution according to Eq. (4).

Based on the training objective of the negative likelihood of the factorized $\Pr(S^{1:T}_{1:N_{\text{agent}}} \mid M)$, we train our models for 30 epochs with a batch size of 24 using the AdamW optimizer. The weight decay rate and the dropout rate are both 0.1. Using the cosine annealing scheduler, we decay the learning rate from 5×10^{-4} to 0. We apply teacher forcing at the patch level when performing next-patch prediction. In contrast, we feed the predicted states to the RNN when conducting next-state prediction within each patch.

Our results are produced by a single model with 3M parameters. To obtain 32 replicas of 8-second rollouts, we independently sample a behavior mode from each agent's next-patch distribution. After mode sampling, we can fix the behavior modes of agents for at most 10 time steps, utilizing the 10-step next-state prediction results produced by the autoregressive RNN. This can lower the inference latency at the cost of lower reactivity. To enhance the reactivity of agents, we tried conducting next-patch prediction every 0.5 seconds. As shown in Tab. 1, increasing the frequency of next-patch prediction to 2 Hz leads to much better results across all metrics. However, the 5-Hz results become worse.

3.2. Main Results

We evaluated our model on the Waymo Open Sim Agents Benchmark [2], with results demonstrated in Tab. 2. Our model surpasses state-of-the-art solutions regarding minADE and achieves the best performance on the map-based metrics. Besides the benchmarking results, we further compare the parameter count of BehaviorGPT against several baselines. Table 3 shows that BehaviorGPT, with only 3M model parameters, significantly reduces the parameter count compared to models like MTR [4] and Trajeglish [3]. These results highlight the parameter efficiency of the decoder-only architecture. Our model also has low inference latency, consuming about 10 ms for a single simulation step on an NVIDIA L20 GPU.

Table 4. Comparison of patch sizes on a subset of the val split.

Patch Size	Replan Frequency	minADE (↓)	Realism	Offroad
1	10 Hz	2.3752	0.6783	0.8432
5	2 Hz	1.5598	0.7272	0.9077
10	1 Hz	1.5203	0.7334	0.9131

In Tab. 4, we compare the simulation results using different patch sizes. As shown in the first row of the table, the simulation results are very bad without any patching operations. Using a patch size of 10 yields better results than using a patch size of 5.

4. Conclusion

This technical report introduced BehaviorGPT, a decoder-only autoregressive architecture for agent simulation in autonomous driving. By applying homogeneous Transformer blocks to the entire trajectory snippets and using relative spacetime representations, BehaviorGPT simplifies the modeling process and maximizes data utilization. We also developed the Next-Patch Prediction Paradigm to task models with predicting trajectory patches instead of single-step agent states, enabling high-level scene understanding and long-range interaction reasoning. Our Waymo Open Sim Agents Challenge results show that BehaviorGPT achieves superior performance with just 3M model parameters, highlighting its potential to further improve simulation realism with more data and computation.

References

[1] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin 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*, pages 9710–9719, 2021. 3

[2] Nico Montali, John Lambert, Paul Mougin, Alex Kuefler, Nicholas Rhinehart, Michelle Li, Cole Gulino, Tristan Emrich, Zoey Yang, Shimon Whiteson, et al. The waymo open sim agents challenge. In *Advances in Neural Information Processing Systems*, 2024. 1, 4

[3] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 4

[4] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 4

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017. 1

[6] Yu Wang, Tiebiao Zhao, and Fan Yi. Multiverse transformer: 1st place solution for waymo open sim agents challenge 2023. *arXiv preprint arXiv:2306.11868*, 2023. 1

[7] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023. 1, 2, 3