Adv-BMT: Bidirectional Motion Transformer for Safety-Critical Traffic Scenario Generation

Yuxin Liu[*†] **Zhenghao Peng**[*†] **Xuanhao Cui**[†] **Bolei Zhou**[†]
[†]University of California, Los Angeles

Abstract

Scenario-based testing is essential for validating the performance of autonomous driving (AD) systems. However, such testing is limited by the scarcity of long-tailed, safety-critical scenarios in existing datasets collected in the real world. To tackle the data issue, we propose the Adv-BMT framework, which augments real-world scenarios with diverse and realistic adversarial interactions. The core component of Adv-BMT is a bidirectional motion transformer (BMT) model to perform inverse traffic motion predictions, which takes agent information in the last time step of the scenario as input, and reconstruct the traffic in the inverse of chronological order until the initial time step. The Adv-BMT framework is a two-staged pipeline: it first conducts adversarial initializations and then inverse motion predictions. Different from previous work, we do not need any collision data for pretraining, and are able to generate realistic and diverse collision interactions. Our experimental results validate the quality of generated collision scenarios by Adv-BMT: training in our augmented dataset would reduce episode collision rates by 20% compared to previous work. The code will be made available at `https://metadriverse.github.io/adv-bmt/`.

1 Introduction

In recent years, autonomous driving (AD) agents have achieved unprecedented performance in urban environments [5, 9, 22]. However, handling corner traffic situations, especially collision scenarios, remains a major challenge. One cause is that safety-critical scenarios are missing from real-world driving datasets due to high costs and risks of data collections. Without enough collision training data, it is hard for the autonomous driving (AD) planners and prediction models to learn safe driving in challenging and risky scenarios. This motivates the need for simulating collision traffic that emulates real-world accidents. According to [13], useful safety-critical scenarios should be realistic and diverse, as well as challenging. Previous works [13, 24, 16, 12] leveraged learned real-world traffic priors to constraint adversarial trajectories, and optimized predictions on collision-encouraging objectives. However, from our observations and evaluations, generated behaviors from these methods have a lack of diversity and a lower collision success rates (attacking efficiency).

To improve upon previous work, we propose the Adv-BMT framework for realistic and diverse collision augmentations from real-world driving data. Adv-BMT samples various collision situations from the input driving log, and then reconstruct complete adversarial trajectories accordingly. This design enhances the diversity of predicted collision interactions, since now it augments different collision behaviors for each driving log. Besides, there is no need to include collision data in trainings. We further implement a rule-based rejection sampling mechanism to filter out candidate trajectories from unsatisfactory adversarial initializations, and maintain realistic collision interactions. In the core of Adv-BMT is the bidirectional motion transformer (BMT) model. Similar to existing multi-agent

[*]Equal contribution

Preprint.

simulators and prediction models [23, 14, 25], BMT tokenizes continuous agent trajectories into discrete actions, and performs next-token predictions in an auto-regressive manner. Different from existing models, BMT learns to predict both future and history motion for all agents, conditioning on one-step current states. This is achieved by processing two sets of motion tokens from the input scenario, with one set of tokens used for forward prediction (future motion) and one for reverse prediction (history motion).

As a summary, Adv-BMT is a two-staged pipeline: first, it initializes diverse collision frames between an new adversary agent (ADV) and ego vehicle; then, it reconstructs the adversarial trajectories via BMT's reverse predictions. In the output, ADV is added to the original scenario, maintaining realistic interactions with surrounding traffic. An overview of our framework is illustrated in Fig. 1. We summarize our contributions as follows: (1) We introduce the bidirectional motion transformer with temporally reversible motion tokenizations; (2) We develop Adv-BMT for realistic and diverse safety-critical traffic simulations; (3) We leverage Adv-BMT in a closed-loop setting to dynamically create challenging environments for reinforcement learning agents.

2 Related Work

Driving Motion Predictions and Simulations Recent work have utilized auto-regressive sequence modeling on discretized motion representations via transformer-based models. In previous works, Trajeglish [11] discretizes motion representations using a k-disk based tokenization scheme to represent position and angle differences for relative movements; but the interaction performance is sensitive to agent ordering, and agents decoded first might not be as reactive to the rest of the predicted agents. MTR++ [15] directly represents motion on continuous space in Gaussian mixture distributions. The design is efficient for multi-modal generations but prediction diversities across modes rely on the learned intention queries. MotionLM [14] and SMART [23] model motion actions that represents trajectory deltas on decoder-only transformers. BehaviorGPT [25] performs next-patch predictions with future motion chunks, instead of single-step predictions, and achieves higher inference efficiency and reduces compounding errors.

Safety-Critical Scenario Generations Previous works leverage learned traffic prior to generate realistic safety-critical interactions. STRIVE [13] enhances diversity through modeling traffic a latent space using variational autoencoders but requires computationally expensive per-scenario optimizations. CAT [24] utilizes the augmentation framework into MetaDrive [9] simulator as a closed-loop reinforcement learning environment. Upon CAT, SEAL [16] uses a skill-based adversarial policy for human-like driving guided by collision-related objectives. MixSim [17] generates realistic reactions by exploring future goals and re-simulates the inferred trajectories using a trained policy. GOOSE [12] uses a goal-conditioned reinforcement learning approach, which parameterizes trajectories and goal constraints to generate scenario-level, safety-critical interactions. AdvSim [20] directly perturbs actor trajectories using a kinematic model and optimizes via a black-box adversarial loss. Recent work [4] uses a conditional normalizing flow to model the distribution of real-world safety-critical trajectories. SafeSim [3] proposes a diffusion model with a test-time collision-sensitive guidance loss to control the collision type and adversarial agent selections; however its generation quality relies on initial collections of collision data. CrashAgent [7] and LCTGen [18] leverage free-form texts as inputs and extract embeddings to parameterize scene initializations and agent driving directions.

3 Methodology

In this section, we formulate forward and reverse motion predictions. We then introduce bidirectional motion transformer (BMT) model. Last but not the least, we demonstrate details of Adv-BMT.

3.1 Problem Formulation

Consider a traffic scenario with maximumly N agents and time horizon T. The trajectory of agent i is denoted by $\boldsymbol{\tau}^i = \{\tau_0^i, \tau_1^i, \ldots, \tau_T^i\}$, where each state $\tau_t^i \in \mathbb{R}^d$ includes position, velocity, and heading at time t. We define a prediction direction indicator $D \in \{\text{Forward}, \text{Reverse}\}$, for future or

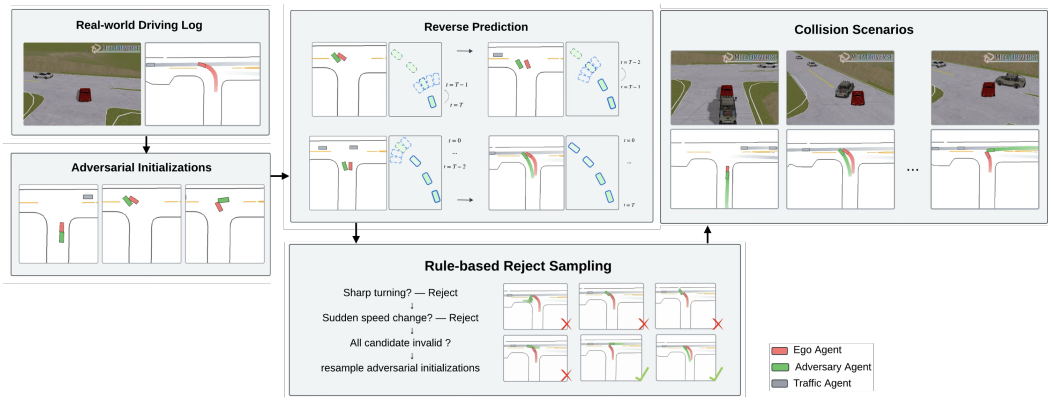

Figure 1: Overview of Adv-BMT framework. From the input scenario, Adv-BMT first samples diverse collision states between ego agent and a new adversarial agent, then reconstruct complete adversarial trajectories via BMT's reverse motion predictions. A rule-based reject sampling mechanism is used for filtering unsatisfactory collision initializations.

past motion prediction within T time steps. For each scenario we construct a sequence of motion tokens:

$$\mathbf{Z}^i = \{z_1^i, \ldots, z_T^i\}, \quad z_k^i = \begin{cases} \phi(\tau_k^i, \tau_{k-1}^i), & D = \text{Forward} \\ \phi(\tau_{T-k}^i, \tau_{T-k+1}^i), & D = \text{Reverse} \end{cases} \tag{1}$$

where $\phi(\cdot)$ denotes the motion tokenization function which we discuss in section 3.2. The goal is to model the distribution of agent motion tokens at the k-th step, conditioned on the predicted motion token sequence, current state, and scene context $\mathcal{M}$:

$$P(z_k^1, z_k^2, \ldots, z_k^N \mid \mathbf{z}_{1:k-1}^1, \mathbf{z}_{1:k-1}^2, \ldots, \mathbf{z}_{1:k-1}^N, \mathcal{M}, D) \tag{2}$$

where $\mathbf{Z}_{0:k-1}^i$ denotes history token sequence for agent i. The decoder autoregressively selects the next action token z_{k+1}^i by attending to past tokens and scene token embedding $\mathcal{M}$. Each selected action in forward prediction maps to a position in future time-step, similar to a standard motion predictor; selected token sequence in reverse prediction can be reconstructed as history trajectories. Together, BMT is able to perform both future and past motion predictions as shown in Fig. 3.

3.2 Bi-direction Motion Tokenizations

Token space BMT's bidirectional motion tokens are derived from a simplified bicycle dynamics model. Both forward and reverse tokens are defined over the same shared token space—a set of discrete bins of acceleration and yaw rate pairs. We uniformly quantize the control space of accelerations $a \in [-a_{\max}, a_{\max}]$ and yaw rates $\delta \in [-\delta_{\max}, \delta_{\max}]$ into K bins each, yielding a total of K^2 discrete motion tokens, where $a_{\max} = 10m/s$, $\delta_{\max} = \frac{\pi}{2}$, and $K = 33$.

Control token simulation Given a motion control token, BMT simulates it and reconstructs to a new agent state using midpoint integration:

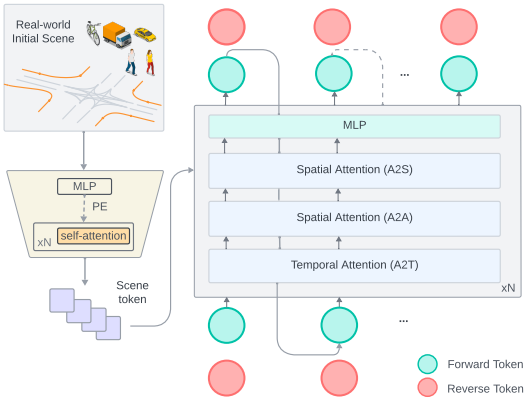

Figure 2: BMT model architecture.

$$\text{Recons}(z, \tau, \Delta t) = \big(x + \bar{v} \cdot \cos(\bar{\theta}) \cdot \Delta t,\ y + \bar{v} \cdot \sin(\bar{\theta}) \cdot \Delta t,\ \theta + \omega \cdot \Delta t,\ v + a \cdot \Delta t\big) \tag{3}$$

where $\tau = (x, y, \theta, v)$ denotes the current agent state with position (x, y), heading θ, and speed v; $z = (a, \omega)$ is the motion token containing acceleration a and yaw rate ω; Δt is the time interval; $\bar{v} = v + \frac{1}{2} \cdot a \cdot \Delta t$ is the midpoint speed; and $\bar{\theta} = \theta + \frac{1}{2} \cdot \omega \cdot \Delta t$ is the midpoint heading.

Ground-truth tokens For obtaining ground truth motion tokens during trainings, we simulate all K^2 candidate motion tokens starting from each agent state at time t, then select the token whose predicted next state $\hat{\tau}_{t+1}$ best matches the ground truth τ_{t+1}. This process maps the continuous trajectory into a sequence of discrete motion tokens $\{z_0, z_1, \ldots, z_T\}$ for each predicted agent. Forward and Reverse predictions share the same motion token space but differ in token interpretations. In reverse predictions, we invert the temporal order of the trajectory and treat the final state (e.g., collision initializations from Adv-BMT) as the initial; at each time step, we simulate each motion token (by inverting the time step, $\Delta t \to -\Delta t$), and find the token that best reconstructs the previous state τ_t from τ_{t+1}. This generates a motion token sequence in reverse time direction $\{\tilde{z}_0, \tilde{z}_1, \ldots, \tilde{z}_T\}$, where $\tilde{z}_0$ corresponds to the final state in forward time.

3.3 Token Matching with minimized Contour Error

For either reverse or forward predictions, we build motion tokens step-by-step in the order of prediction time, starting from the first prediction step and reconstructing till the last prediction step. With a 0.5-second time interval, a search is conducted from all candidate token vocabularies, and selects the best candidate that minimizes the distance between the ground-truth and reconstructed contours for the i-th candidate action for agent n:

$$a_n^* = \arg\min_i \left(\mathcal{E}_n^{(i)} = \frac{1}{P} \sum_{p=1}^{P} \left\| \mathbf{c}_n^{(i,p)} - \hat{\mathbf{c}}_n^{(p)} \right\|_2 \right) \quad \text{where } 0 \leq i < K^2 \tag{4}$$

where $\mathbf{c}_n^{(i,p)} \in \mathbb{R}^2$ and $\hat{\mathbf{c}}_n^{(p)} \in \mathbb{R}^2$ represent the p-th polygon corner of the reconstructed and ground-truth contours, each with totally P contour points; a_n^* is the index of the selected candidate token.

3.4 Bidirectional Motion Transformer (BMT)

The BMT architecture and training process is shown in Fig. 2, along with ground-truth token matching for forward predictions or reverse predictions.

Architecture BMT has a scene encoding component used to obtain embeddings for scenario contexts with separate embeddings for map polylines, traffic lights, and agent initial states. Then, we use Fourier-encoded edge features [19] to represent the spatial and directional information between these encoded entities, which are then passed to the transformer encoder with self-attention layers for the relational embeddings.

The prediction decoder predicts subsequent motion tokens in an autoregressive manner, with only initial agent information for the first frame, along with the scene embedding obtained from the Scene Encoder. The motion decoder incorporates self-attention over the initial agent token embeddings, and three relation computations separately: agent-to-agent (a2a), agent-to-time (a2t), and agent-to-scene (a2s), with each relation embedding then passed to its cross-attention layers. The output agent embeddings are concatenated and repeated a number of times. The output agent motion embeddings are mapped to the vocabulary of discretized motion tokens through MLPs.

Training In training, BMT predicts the sequence of motion tokens and use the cross-entropy objective to match the joint action distribution of the observed behaviors in the dataset during training:

$$\max_{\pi_\theta} \mathbb{E}_{\mathcal{D}} \left[\sum_{t=1}^{T} \sum_{i}^{N} \log \pi_\theta \left(z_t^i \mid \mathbf{z}_{1:k-1}^i, \mathcal{M}, D \right) \right] \tag{5}$$

During inference, BMT generate motion tokens autoregressively, and uses nucleus sampling (top-p sampling) to enhance diversity and maintain plausibility. To prevent the exposure bias during auto-regressive inference, BMT rolls out upon chosen motion tokens instead of the ground-truth token sequence.

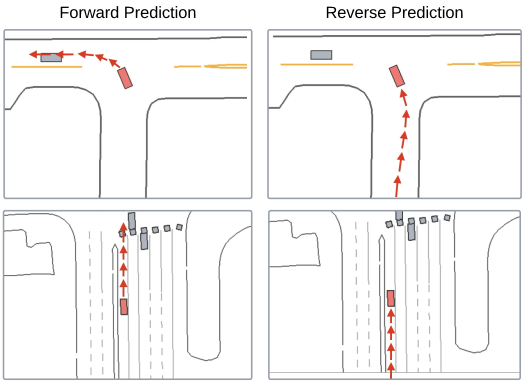

Figure 3: BMT predicts both future and history agent trajectories.

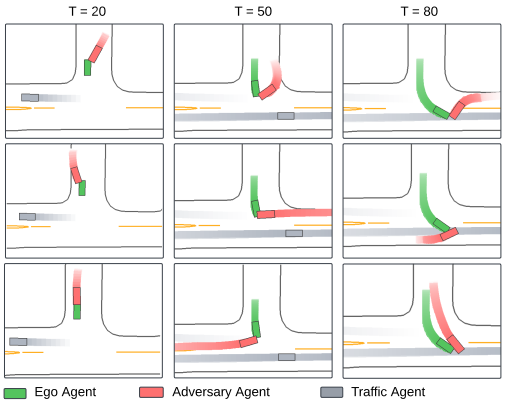

Figure 4: Adv-BMT generates diverse collision scenarios.

3.5 Adv-BMT

The overview of Adv-BMT is illustrated in Fig.1. The framework mainly consists of three steps: (1) it adds an adversarial agent (ADV) with a specified collision state into the current scenario; (2) it predicts reversely for the adversarial trajectory; (3) it performs rule-based checks and reject physically implausible ones.

Diverse Adversarial Initializations While existing works select a convenient neighbor agent and modify the behavior to attack the ego agent, Adv-BMT inserts new agents (ADV) with diverse collision initializations for different opponent interactions. We visualize an example result in 4. Formally, we define the collision state as to include the position, time, velocity, and heading at the collision step. The collision time is sampled from the first second to the last timestep of the ego trajectory length. Similarly, ADV's collision headings are randomly sampled. ADV's collision position can be calculated from collision heading and the ego vehicle's collision position, with the ADV contour intersecting with the ego vehicle's contour. Last but not least, collision velocity is also calculated from sampling a variable offset from ego vehicle's speed at collision step.

Reverse Predictions We use the sampled collision states as inputs of BMT for reverse predictions, and obtain ADV's trajectory from predictions to augment the original scene with a new adv agent. During BMT inference, we teacher-force the ground-truth token sequence of ego agent and traffic agents, to keep the originality of the scenario as much as possible (since BMT could predict agents to driving on a different lane choice, which makes it a totally different scenario). By teacher-forcing traffic vehicle agents, BMT reconstructs the adversarial agent trajectory to fit in the input scenario. Reconstructing entirely new scenarios is out of the scope of our current work, but BMT is able to realize both normal and safety-critical generations through forward and reverse predictions.

Rule-based Rejection Samplings We design Adv-BMT to have diverse initializations, which do not guarantee the realism of the collision outcomes. To address this issue, we implement a rule-based rejection mechanism for ADV candidates. We first measure the driving distance and average speed of ADV candidates; if it moves too short and mostly wanders at the designated position waiting for the ego vehicle, then it is considered invalid. Meanwhile, we check the max curvature (the rate of change for heading): given a candidate ADV trajectory, we compute the curvature constraint using:$\kappa_t = \Delta\theta_t / \Delta s_t$, where $\Delta\theta_t$ is the absolute heading change between time steps, and Δs_t is the displacement between consecutive positions. Adv-BMT rejects a prediction if $\max_t(\kappa_t) > \kappa_{\text{threshold}}$,where $\kappa_{\text{threshold}} = 0.8$ is a predefined curvature limit that we found useful. We check the curvature to ensure it does not exceed our defined threshold, and reject candidate ADVs with unrealistic changes in headings. With our straightforward rule-based rejection sampling mechanism, we are able to maintain only realistic collision events between the ADV and the ego vehicle.

Table 1: Evaluation for BMT on scenario realism on different prediction modes.

	Context	SFDE ↓	SADE ↓	minSFDE ↓	minSADE ↓	FDD ↑	ADD ↑	Collision
Forward	**1s**	**3.043**	**1.019**	1.908	**0.693**	9.241	3.240	**0.026**
Forward	0s	3.180	1.297	2.144	0.927	**10.263**	4.177	0.028
Reverse	0s	3.411	1.380	2.158	0.975	8.234	2.913	0.031
Bidirections	0s	3.059	2.111	**1.831**	1.536	7.618	**6.270**	0.035

Forward refinement for Reactive Traffic In real-world collision events, neighbor vehicles would be affected by the collision vehicles and would have reactions like sudden lane changes and brakings. With this concern, we implement Bi-directional Adv-BMT, with a forward refinement to re-simulate traffic agents upon predicted ADV trajectory. In forward refinement, all traffic agents would be predicted and modified in the output scenario, with teacher-forcing motions of collision participants (ego and ADV agents). In experiments, we refer to it as Adv-BMT (Bi) to indicate bi-directions.

4 Experiments

In section 4.1, we evaluate BMT model on motion prediction performance. In section 4.2, we compare Adv-BMT with baseline methods. In section 4.3 we train a reinforcement learning agent in different safety-critical augmented datasets. For augmentations, we use ground-truth traffic data from the Waymo Open Motion Datasets (WOMD) [6].

4.1 Realism of Adv-BMT

Settings We measure BMT on four different settings: Forward prediction with or without agent history, Reverse prediction without agent history, and bi-directional predictions. We measure on open-loop scenario generation metrics with 6 prediction modes on each scenario: scenario final displacement errors (SFDE) which measures the average displacement error at the final step on the best prediction mode, and scenario average displacement errors (SADE) which measures the averages according to all time steps on the best mode. We also measure Final Displacement Diversity (FDD) and average displacement diversity (ADD) for prediction diversities across different prediction modes: FDD measures the spread of final positions and ADD measures trajectory variations over time. Finally, we measure the overall collision rates for realistic interactions. The results are presented in Table 1.

Analysis Results from Table 1 indicate that BMT performs realistic scenario generations in both forward predictions and reverse predictions. Furthermore, by conditioning on a 1-second history, the forward prediction achieves better SADE and SFDE scores, indicating better temporal consistencies. Due to design of Adv-BMT with agent initializations at the last step, it is impossible for reverse predictions to access agent histories. Compared to single-pass prediction, bi-directional prediction exhibit lower prediction accuracy in ADE metrics, indicating a harm in the realism of generations.

In terms of prediction diversities, Forward prediction exhibits higher FDD and ADD scores than reverse prediction. This indicates that predicting histories is more constrained, but predicting future motions encourages more explorations for variant directions. This further validates the necessity of Adv-BMT's diverse collision initialization design to force different collision outcomes. Despite these differences, the overall collision rate remains reasonable and comparable for all prediction modes, and indicates that both prediction modes effectively generate realistic interactions.

4.2 Safety-critical Evaluations

In following experiments, We compare Adv-BMT with three baseline methods for safety-critical augmentations, namely STRIVE [13], CAT [24], and SEAL [16]. We evaluate the quality and efficiency of generated collisions among all methods. We measure the adversarial attack success rate (SDC-ADV collision rate), collision rates between adversary agents and traffic agents (ADV-BV Collision rate), indicating interaction realism. Additionally, we evaluate the difference between the predicted distribution and the ground-truth distribution on velocities (VEL JSD) and accelerations (ACC JSD). These two metrics also indicate the realism of generated adversarial agent motions.

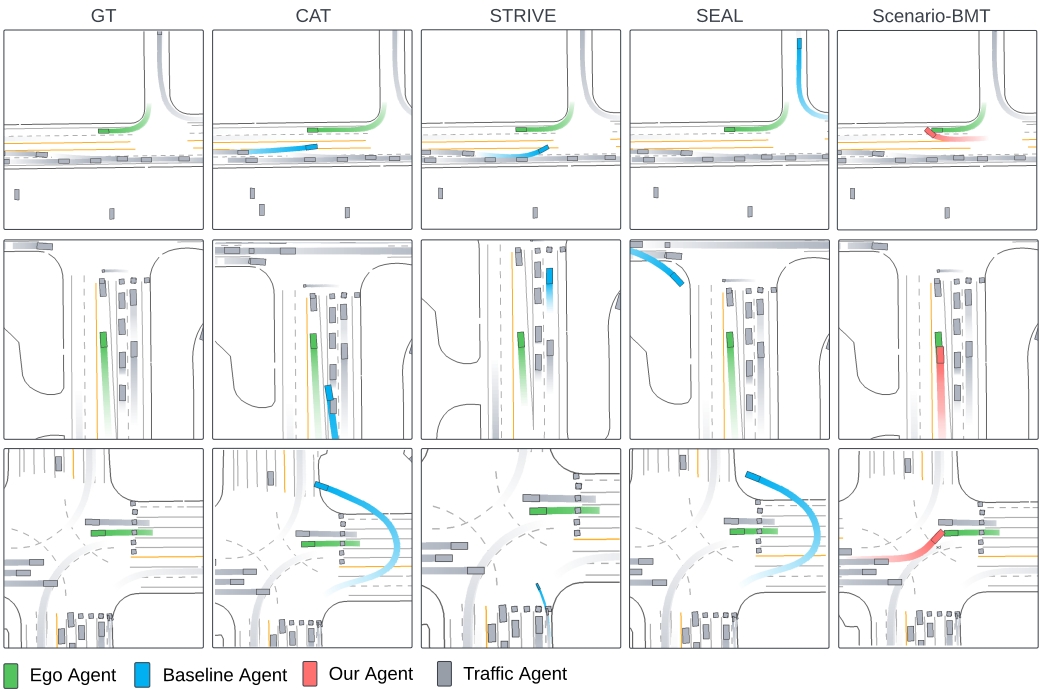

Figure 5: Comparison among adversarial behaviors of different methods.

Table 2: Evaluation for Safety-critical Scenarios

	SDC-ADV Collision ↑	**ADV-BV Collision↓**	**Velocity JSD ↓**	**Accel. JSD ↓**	**FDD↑**	**SDD↑**
CAT [24]	0.470	0.271	0.137	0.480	0.0	0.0
SEAL [16]	0.582	**0.261**	0.239	0.561	0.0	0.0
STRIVE [13]	0.534	0.358	**0.117**	**0.432**	0.004	0.001
Adv-BMT	**1.000**	0.391	0.142	0.435	0	**2.344**
Adv-BMT (Bi)	0.997	0.373	0.188	0.534	**0.038**	2.257

Settings We randomly select 500 WOMD scenarios, each as a base environment for adversarial augmentations. For each scenario, each method predicts six different adversarial trajectories. To be consistent with baseline methods in this evaluation only, we force Adv-BMT to choose an existing neighbor agent and modify its behavior as the predicted adversary.

Quantitative Results Table 2 shows the quantitative results for scenario generation metrics of all methods. Two settings of Adv-BMT outperform baselines in terms of adversarial attack success rate. This validates Adv-BMT's design of collision initialization + reverse predictions, which couldn't be achieved by other methods. While Adv-BMT generates highly adversarial behaviors, it also preserves diversity of traffic interactions compared to baselines. Note tha FDD and SDD metrics suggest that our baselines generate nearly the same adversarial trajectory on the given scene, while Adv-BMT is able to generate diverse behaviors at different prediction mode. The JSD metrics suggests slightly better realism on velocities and accelerations for baselines compared to Adv-BMT, showing the trade-off between diversity and velocity/acceleration realism. BMT model is able to achieve realistic motion predictions indicted by Waymo Open Sim Agent Challenge metrics [10], which we append in our appendix. Overall, our safety-critical evaluations indicate that Adv-BMT is an efficient framework for realistic, diverse, and safety-critical generations.

Visualizations We simulate Adv-BMT scenarios in MetaDrive [9] simulators using both BEV and 3D rendering. In Fig. 4, we show diverse Adv-BMT adversarial behaviors in the collision directions on the same ego agent in an intersection scenario from WOMD. Adversarial agents (in red) come

from different lanes and collide with the ego agent (in green) at different angles and facets. Besides, the generated adversarial agent follows the traffic rules and maintains realistic driving patterns. Apart from diverse collision directions, Adv-BMT generates collisions at diverse timings. Adv-BMT adversarial agents cover collisions with the ego before entering the intersection, during the middle of the intersection, and after turning to the main lane. With a limited number of real-world maps and logs, Adv-BMT is able to generate diverse collision behaviors and timings, making it suitable for AD testing and adversarial training. A visual comparison among Adv-BMT and baseline methods is shown in Fig. 5, where other methods fail to generate an adversarial trajectory but Adv-BMT is able to insert a realistic ADV and triggers collisions .

Generation Speeds Adv-BMT and CAT generate safety-critical scenarios significantly faster than other baselines. Adv-BMT generates one scenario in 1.02s compared to SEAL's 2.36s and STRIVE's 9.53s, achieving great efficiency gains. With details shown in Table 7 in Appendix.

4.3 Adversarial Learning

To validate the value of Adv-BMT scenarios in downstream autonomous driving (AD) tasks, we train a reinforcement learning agent within augmented scenarios with collision interactions generated by Adv-BMT and baseline methods. To determine the quality of augmented training scenarios, we measure both driving performance and safety performance of the learned AD agent compared to learning in original training set. In our experiment, we conduct two sets of trainings. (1) open-loop training: agent is trained in the fixed training set with adversarial scenarios generated based on ground-truth ego trajectories. (2) closed-loop training: an adaptive adversarial agent attacks on current ego agent based on its recent rollout trajectory records. The adaptive adversarial agent's motion is generated by Adv-BMT and baseline. Results of both trainings are shown in table 3 and table 4 respectively.

Settings The training set contains 500 real-world scenarios randomly selected from WOMD training set. Each waymo scenario is augmented with one collision scenario, so that the ratio between waymo and safety-critical scenes is 1:1. For each method, we augments the original dataset into a new training set. For the adaptive adversarial learning experiment, we implement an adaptive generator for Adv-BMT and CAT [24] (discard other methods due to lower inference time). We train a Twin Delayed DDPG (TD3) agent for 1 million steps using 8 random seeds to ensure robustness. We conduct the training in the MetaDrive simulator and use the TD3 algorithm for reinforcement learning (hyper-parameters listed in Appendix). We measure the average reward, average cost, average route completion rate (Compl.), and average episode cost (cost sum) for driving performance measure. We further conducted ablation studies with results shown in Appendix.

Quantitative Results To evaluate the impact of adversarial training using Adv-BMT-generated scenarios, we assess policy performance across two distinct validation environments: (1) 100 Waymo Validation Environments, which consist of unmodified real-world driving scenarios from WOMD validation set, and (2) 100 Adv-BMT Environments, which is the augmented collision scenarios from the 100 validation scenes. Table 3 presents the evaluation results, demonstrating how training with Adv-BMT improves policy resilience in both standard and adversarial settings. A detailed analysis of these results follows in the subsequent sections.

Analysis 1) Improved Policy Robustness in Adversarial Settings. Table 3 presents the evaluation results comparing policies trained with Adv-BMT-generated scenarios against baselines in both real-world and safety-critical validation environments. Adv-BMT demonstrates superior performance in reducing collision rates while maintaining high completion rates. In the collision validation environments, Adv-BMT-trained policies achieve the lowest collision rate while preserving competitive completion rates. This highlights the effectiveness of Adv-BMT in exposing policies to diverse and high-risk scenarios, leading to improved safety performance. Furthermore, Adv-BMT achieves the lowest sum cost, suggesting that policies trained on Adv-BMT-generated data are more efficient in handling safety-critical interactions. Besides, the results from adaptive generators shows an slight improvement in performance compared to training in fixed scenario sets.

In the open-loop learning, among the baselines, STRIVE achieves the highest reward (37.72) in collision environments, but this comes at the cost of a higher collision rate (0.29). SEAL and CAT

Table 3: Open-loop learning evaluation.

(a) Evaluation in the Waymo Validation Environments

Training Scenarios	Reward ↑	Cost ↓	Compl. ↑	Coll. ↓	Cost Sum ↓
Waymo [6]	32.03 ± 4.27	0.39 ± 0.07	0.72 ± 0.05	0.14 ± 0.02	1.41 ± 0.35
CAT [24]	30.37 ± 3.89	0.39 ± 0.05	0.71 ± 0.05	0.14 ± 0.02	1.73 ± 0.39
STRIVE [13]	31.30 ± 3.59	0.40 ± 0.04	0.73 ± 0.05	0.13 ± 0.03	1.51 ± 0.40
SEAL [16]	29.94 ± 5.14	0.39 ± 0.05	0.71 ± 0.04	0.12 ± 0.02	1.63 ± 0.44
Adv-BMT	31.47 ± 3.21	0.38 ± 0.03	0.73 ± 0.04	**0.11 ± 0.02**	**1.35 ± 0.40**
Adv-BMT (Bi)	**33.22 ± 1.83**	**0.36 ± 0.03**	**0.74 ± 0.03**	0.12 ± 0.02	1.39 ± 0.22

(b) Evaluation in the Adv-BMT Environments

Training Scenarios	Reward ↑	Cost ↓	Compl. ↑	Coll. ↓	Cost Sum ↓
Waymo [6]	37.01 ± 6.16	0.64 ± 0.09	0.60 ± 0.07	0.30 ± 0.02	2.96 ± 0.63
CAT [24]	36.77 ± 4.95	0.62 ± 0.05	0.62 ± 0.05	0.29 ± 0.02	3.09 ± 0.56
STRIVE [13]	37.72 ± 5.38	0.63 ± 0.06	0.63 ± 0.06	0.29 ± 0.04	2.92 ± 0.68
SEAL [16]	35.74 ± 6.36	0.67 ± 0.08	0.60 ± 0.06	0.31 ± 0.01	2.97 ± 0.34
Adv-BMT	37.33 ± 3.57	0.62 ± 0.03	0.63 ± 0.04	**0.25 ± 0.05**	**2.41 ± 0.43**
Adv-BMT (Bi)	**39.55 ± 2.94**	**0.59 ± 0.04**	**0.65 ± 0.02**	0.27 ± 0.04	2.74 ± 0.54

Table 4: Closed-loop learning results

(a) Waymo Validation Environments

Adaptive Generator	Reward ↑	Cost ↓	Comp. ↑	Coll. ↓	Cost Sum ↓
CAT	32.15 ± 2.89	**0.38** ± 0.03	0.74 ± 0.04	0.10 ± 0.00	2.02 ± 0.24
Adv-BMT	**33.13** ± 4.11	0.39 ± 0.03	0.74 ± 0.03	**0.09** ± 0.00	**1.25** ± 0.52

(b) Adv-BMT Environments

Adaptive Generator	Reward ↑	Cost ↓	Comp. ↑	Coll. ↓	Cost Sum ↓
CAT	39.47 ± 3.88	0.62 ± 0.05	0.63 ± 0.04	0.22 ± 0.02	2.51 ± 0.45
Adv-BMT	**40.40** ± 6.39	**0.57** ± 0.04	0.63 ± 0.05	0.22 ± 0.04	**2.48** ± 0.97

exhibit higher costs and collision rates, which suggests that adversarial training leads to suboptimal behaviors. In contrast, agent training with Adv-BMT attacks results in improvement on both safety and task completion. We conduct an ablation study on Adv-BMT plus the forward refinement method, and results indicate a slightly better performance in average rewards, and average cost compared to Adv-BMT.

From closed-loop learning shown in table 4, we can see that agent learned with Adv-BMT environments has a significantly lower driving trajectory costs and higher rewards compared to baseline methods. With last step costs, collision rates, and driving completion rate remain consistent with the baseline. Comparing across tables with open-loop evaluations, we can see improvements over all metrics including rewards, completion rates, collision rates, and trajectory costs. This validates the value of diverse safety-critical scenarios which is provided in the closed-loop setting.

5 Conclusion and Limitations

Conclusion This work introduces Adv-BMT, a novel safety-critical scenario generation framework that leverages BMT model to bring diverse and realistic adversarial collisions into real-world traffic scenarios. Adv-BMT consists of three key components: (1) collision initialization, which samples a potential collision outcome as the last frame; (2) reverse prediction, which generates the adversarial pre-collision trajectories; (3) rule-based filtering, which ensures adversarial agents' motions remain realistic and feasible. We evaluate Adv-BMT across multiple benchmarks and experiments and demonstrate that training with generated adversarial scenarios improves AD agent performance in both held-out log-replay and collision test sets with a clear margin.

Limitations Following limiations of our current work will be potentially addressed in the future: (1) we only consider adversarial behaviors among vehicle agents, but not among other agent types like pedestrains and bicycles. (2) we only evaluate our adversarial scenarios in RL policies, but adversarial scenarios can also be leveraged in human-in-the-loop learning.

References

[1] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving, 2020.

[2] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. Nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles, 2022.

[3] Wei-Jer Chang, Francesco Pittaluga, Masayoshi Tomizuka, Wei Zhan, and Manmohan Chandraker. Safe-sim: Safety-critical closed-loop traffic simulation with diffusion-controllable adversaries, 2024.

[4] Wenhao Ding, Baiming Chen, Bo Li, Kim Ji Eun, and Ding Zhao. Multimodal safety-critical scenarios generation for decision-making algorithms evaluation, 2020.

[5] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[6] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset, 2021.

[7] Miao Li, Wenhao Ding, Haohong Lin, Yiqi Lyu, Yihang Yao, Yuyou Zhang, and Ding Zhao. Crashagent: Crash scenario generation via multi-modal reasoning, 2025.

[8] Quanyi Li, Zhenghao Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling, 2023.

[9] Quanyi Li, Zhenghao Peng, Lan Feng, Qihang Zhang, Zhenghai Xue, and Bolei Zhou. Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning, 2022.

[10] 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, 2023.

[11] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Traffic modeling as next-token prediction, 2024.

[12] Joshua Ransiek, Johannes Plaum, Jacob Langner, and Eric Sax. Goose: Goal-conditioned reinforcement learning for safety-critical scenario generation, 2024.

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

[14] Ari Seff, Brian Cera, Dian Chen, Mason Ng, Aurick Zhou, Nigamaa Nayakanti, Khaled S. Refaat, Rami Al-Rfou, and Benjamin Sapp. Motionlm: Multi-agent motion forecasting as language modeling, 2023.

[15] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Mtr++: Multi-agent motion prediction with symmetric scene modeling and guided intention querying, 2024.

[16] Benjamin Stoler, Ingrid Navarro, Jonathan Francis, and Jean Oh. Seal: Towards safe autonomous driving via skill-enabled adversary learning for closed-loop scenario generation, 2025.

[17] Simon Suo, Kelvin Wong, Justin Xu, James Tu, Alexander Cui, Sergio Casas, and Raquel Urtasun. Mixsim: A hierarchical framework for mixed reality traffic simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9622–9631, June 2023.

[18] Shuhan Tan, Boris Ivanovic, Xinshuo Weng, Marco Pavone, and Philipp Kraehenbuehl. Language conditioned traffic generation, 2023.

[19] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020.

[20] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles, 2023.

[21] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting, 2023.

[22] Wayne Wu, Honglin He, Jack He, Yiran Wang, Chenda Duan, Zhizheng Liu, Quanyi Li, and Bolei Zhou. Metaurban: An embodied ai simulation platform for urban micromobility, 2024.

[23] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction, 2024.

[24] Linrui Zhang, Zhenghao Peng, Quanyi Li, and Bolei Zhou. Cat: Closed-loop adversarial training for safe end-to-end driving, 2023.

[25] Zikang Zhou, Haibo Hu, Xinhong Chen, Jianping Wang, Nan Guan, Kui Wu, Yung-Hui Li, Yu-Kai Huang, and Chun Jason Xue. Behaviorgpt: Smart agent simulation for autonomous driving with next-patch prediction, 2024.

In the appendix, we include additional technical details and results upon the main paper due to limit of page length. Appendix A gives details on the model architecture and trainings, Appendix B gives details for experiments, and Appendix C and D provides additional results to supplement those in the main paper. In Appendix E, we claim the broader impacts of our methodology.

A Model Details

A.1 BMT Architecture

Scenario Format and Preprocessing The model takes real-world scenarios with the format from ScenarioNet [8], which unifies real-world datasets including [1] [2] [21].

The model performs a series of preprocessing on both static map features and dynamic agent trajectories. We compute the global boundary to extract a consistent map center and heading. Each map feature is decomposed into a sequence of vectorized segments, where each vector is represented by a start and end point in 3D coordinates. These vectors are then augmented with directional relative positions, segment headings, and lengths. Semantic attributes are binary indicators that encode feature types. The map feature types (lane, crosswalk, broken line, yellow line, stop sign, etc.) are encoded for semantic information. Features are then centralized to map center. Traffic light states are also encoded and aligned with the steps. Similarly, agent trajectories are also centralized for agent feature encode, with information of (position, heading, velocity, shape, and type) extracted across time for temporal sequences. Besides, there is reordering for ego agent and agents to be predicted. We extract a 16-dimensional state vector at each timestep. The preprocessing ensures all trajectories are represented in a consistent spatial frame and translated to an ego-centric coordinate system.

Tokenization We formulate motion prediction as an auto-regressive next-token prediction problem, where each motion token corresponds to a discretized control input for a fixed time interval. Tokenization process maps continuous agent motion (acceleration and yaw rate) into discrete 2D bins. The tokenization process considers candidate tokens sampled from a fixed grid of bins, and simulates the resulting motion over a short duration, and then selects the best-matching action by minimizing the contour alignment error between the predicted agent shape and the ground-truth position and heading. We adopt a simplified version of the bicycle model to parameterize agent motion using longitudinal acceleration and yaw rate within predefined bounds: acceleration is limited to the range of $[-10, 10]$ m/s^2, and the yaw rate is constrained to $[-\pi/2, \pi/2]$ rad/s. With predicted motion token sequences, the trajectory can be reconstructed by mapping the token back to the acceleration and yaw angle change.

In both forward and reverse directions, motion tokens are decoded into continuous trajectories using the same tokenizer. Given an initial state $\tau_t = (x_t, y_t, \theta_t, v_t)$, the forward decoding process simulates the agent's next state τ_{t+1} by applying a tokenized control action $z_t = (a_t, \delta_t) \in \mathcal{A}$, where $\mathcal{A}$ denotes the discrete token space. This is repeated autoregressively over the sequence of predicted tokens to reconstruct the full trajectory $\boldsymbol{\tau}_{t:t+T}$. In contrast, reverse decoding starts from a known future state τ_{t+1}, the model evaluates all possible token candidates $z_t \in \mathcal{A}$, simulates their inverse dynamics using $\Delta t \to -\Delta t$, and selects the token that best reconstructs the preceding state τ_t. The ability to operate in both directions is a key distinction of our approach: forward prediction enables open-loop simulation of future behaviors, while reverse prediction allows us to trace back from a desired outcome (e.g., a collision state) to plausible initiating actions.

Decoding Architecture. The decoder follows a GPT-like structure composed of stacked cross-attention layers. Each layer integrates three structured attention modules: agent-to-agent (A2A), agent-to-temporal (A2T), and agent-to-scene (A2S). These modules attend over dynamically constructed graphs defined by spatial or temporal adjacency. Relational information across modalities is captured via multiple embeddings: Agent-to-Agent Relation Embedding, Agent-to-Time Relation Embedding, and Agent-to-Scene Relation Embedding, each encoding context-specific spatial information. For token construction, several embeddings are used, including the Agent Type Embedding, Agent Shape Embedding, Agent ID Embedding, and the Motion Token Embedding which maps discrete control tokens. A Continuous Motion Feature Embedding is applied to embed acceleration and yaw rate attributes. Auxiliary embeddings include the Special Token Embedding for indicating sequence boundaries, and the Backward Prediction Indicator Embedding to distinguish between forward and

backward prediction modes. For each attention edge, a relative relation embedding is computed using a Fourier encoder and added to the key and/or value vectors. The input agent token $\mathbf{X} \in \mathbb{R}^{B\times T\times N\times d}$ (batch size B, time steps T, number of agents N, hidden dimension d) is progressively updated across layers by aggregating contextual features from other agents, their temporal history, and relevant map elements.

Motion Token Embedding Specifically, each input agent token to the BMT motion decoder is constructed by summing of embeddings of:

a. the motion token from the previous step (representing discretized acceleration and yaw rate),
b. agent shape (length, width, height),
c. agent type (e.g., vehicle, pedestrian),
d. agent identifier (embedded optionally),
e. special token type (e.g., `<start>`, `<end>`, or padding),

as well as a continuous motion delta feature embedded via a Fourier encoder. These components are projected into the same hidden dimension and summed to form the input motion token embedding.

Prediction Heads and Outputs After processing through all decoding layers, the final hidden state for each valid token is passed through a two-layer MLP head to produce logits over the motion token space:

$$\texttt{MLP}(\mathbf{h}) = W_2 \cdot \phi(W_1 \cdot \mathbf{h}) \in \mathbb{R}^{K^2},$$

where $\phi(\cdot)$ denotes the GELU activation and K^2 is the number of discrete motion tokens (from a $K \times K$ acceleration–yaw bin grid).

The resulting logits are used to predict the next motion token at each time step. During inference, we generate motion tokens using nucleus (top-p) sampling.

Trajectory Reconstruction Our model makes a prediction in the interval of 5 time steps (0.5 seconds). To simulate the effect of a motion token over a fixed time step $\Delta t = 0.5\,\mathrm{s}$, we adopt midpoint integration based on a simplified bicycle model. In forward predictions, given a current state $\mathbf{s}_t = (x_t, y_t, \theta_t, v_t)$, the model computes the next speed and heading as $v_{t+1} = v_t + a \cdot \Delta t$ and $\theta_{t+1} = \theta_t + \omega \cdot \Delta t$. The average speed and heading are then defined as $\bar{v} = \frac{v_t + v_{t+1}}{2}$ and $\bar{\theta} = \left(\frac{\theta_t + \theta_{t+1}}{2}\right)$. In reverse prediction, the process is reverted. In the backward direction, the process is inverted. Given a future state $\mathbf{s}_{t+1}$, the model enumerates all possible token candidates and inverts the dynamics: $v_t = v_{t+1} - a \cdot \Delta t$ and $\theta_t = \theta_{t+1} - \omega \cdot \Delta t$. The average quantities $\bar{v}$ and $\bar{\theta}$ are computed similarly and used to derive the previous position:

$$x_t = x_{t+1} - \bar{v} \cdot \cos(\bar{\theta}) \cdot \Delta t, \quad y_t = y_{t+1} - \bar{v} \cdot \sin(\bar{\theta}) \cdot \Delta t.$$

Decoder Training Loss The decoder produces a logit tensor $\hat{\mathbf{z}} \in \mathbb{R}^{B\times T\times N\times |\mathcal{A}|}$, where $|\mathcal{A}|$ is the number of motion tokens (i.e., discretized acceleration–yaw pairs). The supervision target is the ground-truth token sequence $\mathbf{z}^* \in \mathbb{N}^{B\times T\times N}$, derived by tokenizing agent trajectories. A binary mask $\mathbf{m} \in \{0,1\}^{B\times T\times N}$ specifies which tokens are valid and should contribute to the training loss. The training objective is computed over all valid entries using the cross-entropy loss:

$$\mathcal{L}_{\text{main}} = \frac{1}{\sum_{b,t,n} m_{b,t,n}} \sum_{b,t,n} m_{b,t,n} \cdot \text{CE}(\hat{z}_{b,t,n}, z^*_{b,t,n}),$$

where CE denotes the standard cross-entropy loss between the predicted logits and the ground-truth discrete token.

Reverse Prediction During reverse prediction, the model measures metrics separately for forward and reverse token predictions. Let $\mathbf{b} \in \{0,1\}^{B\times T\times N}$ be a binary indicator for whether each token comes from reverse prediction. Then we compute separate metrics:

$$\text{Accuracy}^{\text{reverse}} = \frac{\sum m_{b,t,n} \cdot b_{b,t,n} \cdot \mathbf{1}[\hat{z}_{b,t,n} = z^*_{b,t,n}]}{\sum m_{b,t,n} \cdot b_{b,t,n}},$$

$$\text{Entropy}^{\text{reverse}} = \frac{1}{\sum m \cdot b} \sum m_{b,t,n} \cdot b_{b,t,n} \cdot \mathcal{H}(\hat{z}_{b,t,n}),$$

Table 5: BMT Model Parameters.

Component	Parameters	Size (MB)
Scene Encoder	902,080	3.44
Map Polyline Encoder	22,656	0.09
Traffic Light Embedding MLP	1,024	0.00
Scene Relation Embedding	117,184	0.45
Scene Transformer Encoder	744,448	2.84
Scene Encoder Output Projection	16,512	0.06
Scene Output Pre-Normalization	256	0.00
Motion Decoder	4,385,025	16.73
Multi-Cross Attention Decoder	2,881,536	10.99
Motion Prediction Head	157,121	0.60
Motion Prediction Pre-Normalization	256	0.00
Agent-to-Agent Relation Embedding	418,432	1.60
Agent-to-Time Relation Embedding	418,432	1.60
Agent-to-Scene Relation Embedding	117,184	0.45
Agent Type Embedding	640	0.00
Motion Token Embedding	139,520	0.53
Agent Shape Embedding	17,152	0.07
Agent ID Embedding	16,384	0.06
Continuous Motion Feature Embedding	217,600	0.83
Special Token Embedding	512	0.00
Reverse Prediction Indicator Embedding	256	0.00
Total	5,287,105	20.17

with analogous expressions for forward prediction (i.e., for $1 - b_{b,t,n}$).

Metrics To measure the quality and diversity of the model's predictions during training, we track the perplexity:

$$\text{Perplexity} = \exp\left(-\sum_{a\in\mathcal{A}} \bar{p}_a \log(\bar{p}_a + \epsilon)\right), \quad \text{where} \quad \bar{p}_a = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}[\hat{z}_i = a],$$

and M is the number of valid tokens. We also track the number of distinct tokens used by both predictions and ground truth:

$$\text{Cluster} = \sum_{a\in\mathcal{A}} \mathbf{1}[\bar{p}_a > 0].$$

Total Loss The total loss is the sum of all enabled components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{main}} + \lambda_{\text{map}}\mathcal{L}_{\text{map}} + \lambda_{\text{tg}}\mathcal{L}_{\text{tg-total}},$$

with default weights $\lambda_{\text{map}} = \lambda_{\text{tg}} = 1$.

A.2 Training Details

Our model has 5.2 million trainable parameters, with detailed break downs indicatd in Table 5. We trained BMT model on the training set of Waymo Open Motion Datasets [6]. WOMD contains 480K real-world traffic with each scenario of length 9 seconds; traffic are composed by agents of vehicle, pedestrian, and bicycle; Each scenario comes with a high-fidelity road map. During training, we use 8 NVIDIA RTX A6000 GPUs for our model training and fine-tunings. We trained BMT in two stages, each with hyper-parameters indicated in Table 6. In the first stage, we pre-trained BMT for forward prediction only with 1 million steps. Then, we fine-tuned BMT with reverse motion prediction in fine-tuning with totally 1.5 million steps. We use AdamW optimizer for learning rate scheduling.

Table 6: BMT Training settigs.

Forward Prediction	
Hyper-parameter	Value
Training steps	10E6
Batch sizes	2
Training Time (h)	185
Sampling Topp	0.95
Sampling temperature	1.0
Learning Rates	3E-4

Reverse Prediction	
Hyper-parameter	Value
Training steps	15E6
Batch size	2
Training Time (h)	310
Sampling Topp	0.95
Sampling temperature	1.0
Learning Rates	3E-4

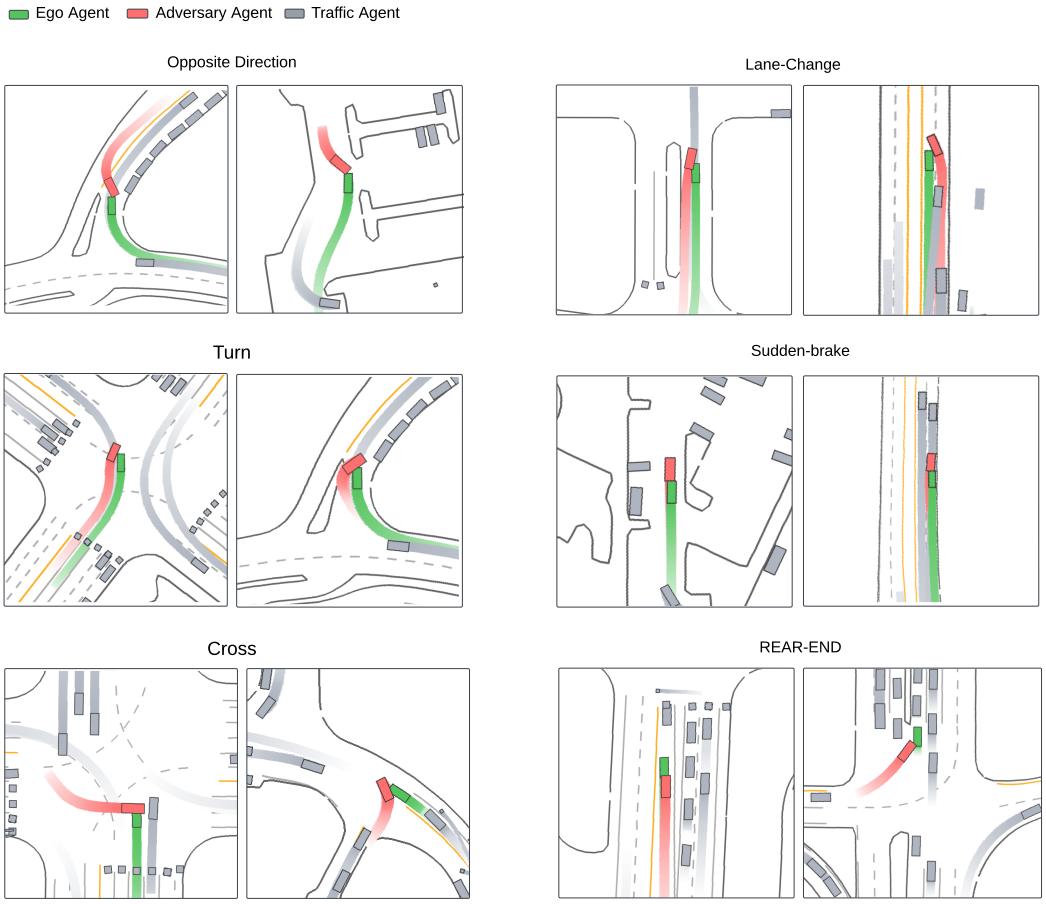

Figure 6: Diverse adversarial behaviors generated by Adv-BMT.

B Experiment Details

Training Environment We conduct our reinforcement learning experiments using the MetaDrive ScenarioEnv [9], which provides standardized driving environments for training and evaluating autonomous agents. Each environment encodes sensor observations including LiDAR-based surroundings and physical dynamics. Specifically, the observation space consists of three key components: (i) Ego state, which contains the ego vehicle's current physical state such as speed, heading, and steering; and (ii) surroundings, which encodes nearby traffic objects.

Actions are continuous and correspond to low-level vehicle control commands. The agent outputs a 2D normalized vector, which is then mapped to steering angle, throttle (acceleration), and brake signal. The environment includes a compositional reward structure combining driving progress, collision

Table 7: RL training settings.

Adv-BMT	
Hyper-parameter	Value
Scenario Horizon	9s
History Horizon	0s
Collision Step	1s–9s
Prediction Mode	8
Policy Training Steps	10E6

TD3	
Hyper-parameter	Value
Discounted Factor	0.99
Train Batch Size	1024
Learning Rate	1E-4
Policy Delay	200
Target Network	0.005

Generation Time	
Method	Time
CAT [24]	0.80s
SEAL [16]	2.36s
STRIVE [13]	9.53s
Adv-BMT	0.74s

Table 8: WOSAC Evaluation results of BMT.

Metrics	**Reverse**	**Forward**
Linear speed	0.375	0.393
Linear acceleration	0.394	0.405
Angular speed	0.441	0.428
Angular acceleration	0.594	0.593
Distance to nearest object	0.405	0.388
Collision	0.521	0.951
Time to Collide	0.840	0.840
Distance to road edge	0.675	0.683
Offroad	0.564	0.934
Realism	0.554	0.753
Kinematic	0.451	0.455
Interactive	0.566	0.801
Map	0.596	0.862
minADE	2.148	1.344
Metametric	0.554	0.753

penalties, and road boundary violations. Driving reward is measured by forward lane progress, while penalties are applied for collisions with other vehicles or drifting off-road.

Hyper-parameters The settings of our open-loop and closed-loop adversarial RL experiments are shown in table 7. Note that in our closed-loop learning, Adv-BMT takes one frame of agent information as input for adversarial generations, whereas all baseline methods take one second agent history.

C Quantitative Results

C.1 Waymo Open Sim Agents Challenge

We evaluated BMT resutls on 400 WOMD validation scenarios using Waymo Open Sim Agents Challenge (WOSAC) 2024 [10]. Evaluation results are summarized in Table 8. Within the metrics, for minADE metrics, smaller values indicate better more accurate perdictions, and for the rest metrics, the larger score means better performance. Within the results, we can see that forward prediction achieves much better performance compared to reverse prediction across all metrics, except similar performance in Angular speed, angular acceleration, distance to nearest object, and TTC. In these

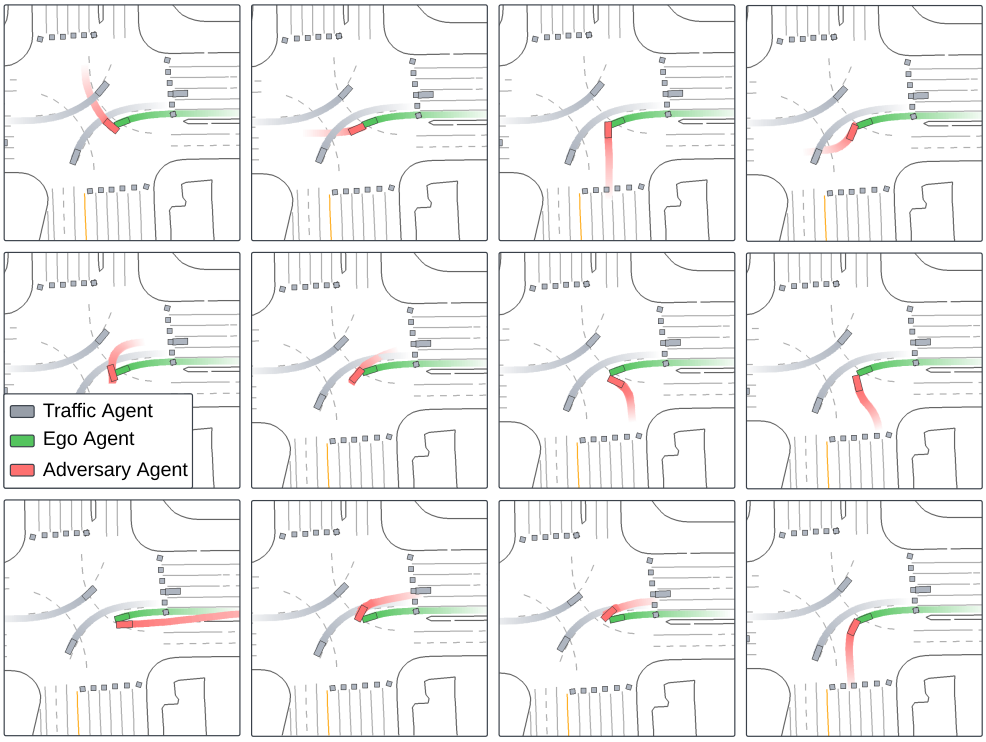

Figure 7: Diverse adversarial behaviors generated on one driving log by Adv-BMT.

mentioned metrics. Note that the training time for forward prediction and reverse prediction are similar (10E6 and 15E6 steps respectively). The WOSAC results indicate that our BMT model is better at predicting future motion than predicting history motion. We will submit to the official WOSAC learderboard for further reference in the future.

D Qualitative Results

Visualizations In Fig. 6, we demonstrate six pairs of qualitative results by Adv-BMT. In different scenarios, our opponent behavior exihibits in different category of safety-critical driving patterns. This means that our augmented opponent agents have plausible interactions with realistic driving trajectories. Besides, in Fig. 7, we demonstrate diverse collisions from just one single scenario. This is the power of Adv-BMT compares to other baseline work, which generates the same or similar adversarial behavior with the same driving log.

Demo Video We submit a video within our supplementary materials. Here we put visualizations with case studies through animated visualizations of Adv-BMT scenarios, which simulates different types of vehicles, pedestrians and bicycle agents.

E Broader Impacts

Our work introduces a novel model for generating safety-critical traffic scenarios, aiming to improve the safety reliability and driving robustness of AD systems. By modeling both forward and reverse motion trajectories, our framework enables controllable and diverse simulation of rare and high-risk traffic events. Our framework, Adv-BMT, benefits the development and testing of safer autonomous agents by exposing failure cases under challenging interactions. However, generating adversarial scenarios may potentially raise concerns about potential misuse, such as crafting unrealistic or malicious simulations. To address this, our approach is designed for research and evaluation within closed simulation environments. We encourage responsible usages of our model and encourage integrating them into safety validation pipelines with appropriate regulations.